ACCUPOWER®

HOCHSCHULE MITTWEIDA
University of Applied Sciences

DIPLOMARBEIT

Herr Ing.
Maximian Geyer

AccuControlCenter

Automatisierung der Endmontage

Mittweida, Mai 2023

Dieses Dokument wurde mit Hilfe von KOMA-Script und LaTeX gesetzt.

Fakultät **Wirtschaftsingenieurwesen**

DIPLOMARBEIT

AccuControlCenter

Automatisierung der Endmontage

Autor:
Maximian Geyer

Studiengang:
Elektrotechnik

Seminargruppe:
KE19wWA

Erstprüfer:
Prof. Dr.-Ing. Swen Schmeißer

Zweitprüfer:
Jan Roloff, M.Sc.

Einreichung:
Mittweida, 22.05.2023

Verteidigung/Bewertung:
Mittweida, 13.06.2023

Faculty of **Industrial Engineering**

DIPLOMA THESIS

AccuControlCenter

Author:
Maximian Geyer

Course of Study:
Electrical engineering

Seminar Group:
KE19wWA

First Examiner:
Prof. Dr.-Ing. Swen Schmeißer

Second Examiner:
Jan Roloff, M.Sc.

Submission:
Mittweida, 22.05.2023

Defense/Evaluation:
Mittweida, 13.06.2023

Bibliografische Beschreibung:

Geyer, Maximian:
AccuControlCenter, *Automatisierung der Endmontage*. – 2023. – 68 S.
Mittweida, Hochschule Mittweida – University of Applied Sciences, Fakultät Wirtschaftsingenieurwesen, Diplomarbeit, 2023.

Referat:

Das AccuControlCenter ist eine in C# geschriebene Software. Diese steuert, überwacht und dokumentiert Prozesse der Endmontage und reduziert die Fehleranfälligkeit durch Eingriffe des Menschen, erhöht den Arbeitsschutz durch Automatisierung und reduziert Kosten. Das Kernelement des ACC ist das Beschreiben und Auslesen von µC, welche sich auf den intelligenten Akkupacks befinden.

Inhaltsverzeichnis

Ergänzende Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

µC . . . . . Mikrocontroller

ACC . . . . . AccuControlCenter

ACK . . . . . Acknowledgement

API . . . . . Application Programming Interface, deutsch Anwendungsprogrammierschnittstelle

ASCII . . . . . American Standard Code for Information Interchange

BMS . . . . . Battery Management System

bqStudio ..... Battery Management Studio

CI ..... Corporate Identity

CRUD ..... Create, Read, Update, Delete

DB ..... Datenbank

DBMS ..... Datenbankmanagementsystem

DHCP ..... Dynamic Host Configuration Protocol

EPU ..... Ein-Personen-Unternehmen

ERP ..... Enterprise Ressource Planning

GUI ..... Graphical User Interface

I2C ..... Inter-Integrated Circuit

IEEE ..... Institute of Electrical and Electronics Engineers

IoT ..... Internet of Things

IT ..... Informationstechnologie

MB ..... MegaByte

ms ..... Millisekunde

MS SQL Server .. Microsoft Structered Query Language Server

NAK ..... Not Acknowledged

OT ..... Operational Technology

PaaS ..... Platform as a Service

px ..... Pixel

SaaS ..... Software as a Service

SCPI ..... Standard Commands for Programmable Instruments

SMB ..... System Management Bus

SoC ..... Sytem on a Chip

TI ..... Texas Instruments

VB ..... Visual Basic

WPF ..... Windows Presentation Foundation

XAML ..... Extensible Application Markup Language

XML ..... Extensible Markup Language

Vorwort

An einem lauen Hochsommertag im August, genauer gesagt am 23.08.2022, wurde ich von meinem ehemaligen Arbeitskollegen Patrick Otipka, von der Firma AccuPower, kontaktiert, ob ich denn Zeit hätte, das von mir vor 2 Jahren entwickelte AccuControlCenter, kurz ACC, weiter auszubauen. Erfreut über diese Anfrage machten wir uns einen Termin mit allen beteiligten Personen am 30.08.2022 aus, um weitere Einzelheiten zu besprechen. AccuPower ist spezialisiert auf die Fertigung von Sonderlösungen im Bereich der mobilen Energieversorgung. Zu den Kunden gehören neben AVL, Audi, Opel, Porsche und VW auch das österreichische Bundesheer [1] [2].

Abbildung 1: AccuPower Logo
[3]

Durchgeführt wird die Diplomarbeit von der Firma Webgeyer, welche im Februar 2013 von mir gegründet wurde und somit einen freien Handlungsspielraum für beide Parteien auf mehreren Ebenen ermöglicht. Das erste Treffen wurde mittels Zoom durchgeführt, ein mittlerweile alltäglicher Begleiter seit der Coronapandemie [4].

Abbildung 2: Webgeyer Logo
[5]

Während des Gespräches wurden Sorgen, Probleme und Aufgaben erörtert und mögliche Lösungen skizziert, abschließend stand fest, dass wir einen kommenden physischen Vororttermin bei AccuPower benötigen werden, um letzte Feinschliffe anzulegen und einander physisch kennenzulernen, denn mittlerweile hatte die Geschäftsführung von AccuPower gewechselt und der physische Raum hat doch Vorteile gegenüber dem Cyberraum, was menschliche Gepflogenheiten betrifft, jedoch war der Same für das Entstehen einer erfolgreichen Diplomarbeit bereits gelegt worden und das Wachstum nicht mehr aufzuhalten [6].

Danksagung

Bedanken möchte ich mich bei meiner Freundin Antonia Kraft, welche mir immer die notwendige Zuversicht und Zeit zugestand, welche erforderlich war, um diese Arbeit zu bewältigen, ohne ihre Geduld und ihr Verständnis wäre das erfolgreiche abschließen dieser Arbeit nicht möglich gewesen.

Meinen Eltern Birgit Kühhas und Robert Geyer ist es zu verdanken, dass in mir ein solides Fundament der Beständigkeit und Umsetzungswilligkeit zur Lösung von Problemen jedweder Art ruht und diesem durch so manchen Quellcode Ausdruck verliehen wurde. Thomas Geyer, mein Bruder, ermöglichte mir ein gutes Vorankommen in der Dauer der Diplomarbeitserstellung durch ein temporäres Pausieren meiner anderweitigen Pflichten. Bescheidenheit und Disziplin waren weitere notwendige Säulen, um die Komplexität und Kompliziertheit gewisser Thematiken zu meistern und diese konnte ich mir seit meiner frühestens Kindheit durch das Beobachten meiner Großeltern Ingrid und Josef Kühhas aneignen.

Herrn Prof. Dr.-Ing. Swen Schmeißer möchte ich für die hilfreichen Hinweise zur Verbesserung der Diplomarbeit danken und auch Herrn Jan Roloff, M.Sc. für die Zeit zur Durchsicht und Korrektur.

Meine Begeisterung für die Elektronik und Elektrotechnik wurde von meinem alten Chef Herrn Ing. Issam Al-Abassy geweckt und somit wäre diese Arbeit wohl kaum jemals ohne seine belehrenden Worte zustande gekommen.

Bei Moritz Minarik möchte ich mich für das Vertrauen und die Möglichkeit zur Umsetzung dieser Diplomarbeit bedanken, die hoffentlich zum weiteren wirtschaftlichen Erfolg von AccuPower beiträgt.

Viel Ausdauer war von Nöten, um die unzähligen Testdurchläufe zu planen und durchzuführen, dafür möchte ich mich ganz herzlich bei Ing. Tomislav Renjo bedanken. Patrick Otipka danke ich für die vielen hilfreichen Informationen, um Besonderheiten der Prozessoren schneller zu verstehen. Bei David Gradwohl möchte ich mich für unkomplizierte und schnelle Reaktionen bzgl. des bürokratischen Aktes bedanken und das im Auge behalten der Metaebene.

Georg Stolberg und Sebastian Alecu danke ich für ihre Geduld, mir bei Fernzugriffen ein physisches Instrumentarium samt Konfiguration zur Seite zu stellen, um alle Anforderungen gut möglichst umzusetzen und testen zu können und auch den Abbau dieses.

Den wohl meist betroffenen Shamil Muradov, Daniel Ivanov und Bayram Karajic, danke ich für ihre Geduld, unzählige Tests durchzuführen und die Rückmeldungen kleinerer oder größerer Fehler um die Software somit stetig zu verbessern.

Abschließend gilt mein Dank wohl dem offensichtlichsten und doch meist Vergessenem: der Schöpfung und ihres Strahls, ohne welchem all dies nicht möglich wäre.

1 Einleitung

Diese Diplomarbeit hat zum Ziel, eine Software zu entwickeln, zu designen und zu implementieren, welche (Teil-)Prozesse automatisiert, Fehleranfälligkeit reduziert und die Effizienz und Durchlaufgeschwindigkeit erhöht und entsprechend Kosten senkt. Dazu ist es notwendig, alle benötigten (Teil-)Schritte der Fertigung zu erheben und entsprechend zu atomarisieren, um Lösungen zu erarbeiten und das Ziel vor Komplexität nicht aus den Augen zu verlieren.

Dieses Ziel wird erreicht mithilfe unterschiedlichster Technologien, begonnen bei primitiven Bits und Bytes, welche auf dem I2C-Bus zu dem µC übertragen werden, zu höherwertigen SCPI-Befehlen, um Messungen oder Belastungen durchzuführen und um abschließend die erhobenen Ergebnisse in einer Datenbank zu speichern und für spätere Auswertung nutzbar zu machen.

Die Notwendigkeit besteht in der wachsenden Auftragslage und der Lokalisierung des Engpasses Mensch. Bei AccuPower wird jeder einzelne Akkupack, bevor er die Versandzone erreicht, einer Qualitätskontrolle unterzogen und nach Wünschen des Kunden konfektioniert. Die Konfektion wird aktuell zum Großteil händisch durchgeführt, was beim Abruf von Hunderten Akkupacks zu einer monotonen und fehleranfälligen Arbeit ausufert. Da wiederkehrend die exakt selben Abläufe wiederholt werden müssen, ist dies ein Paradebeispiel für die Automatisierung.

Begonnen wird bei den Grundlagen der eingesetzten Technologien, anschließend wird die Planung und die Anforderungen an die Software beleuchtet und abschließend werden praktische Umsetzungen der Prozesse in Quellcode behandelt.

Eine konkrete Anleitung zum 1:1 Nachbau dieser Software findet sich nicht in dieser Diplomarbeit. Einzelne Prozesse sind von der Firma AccuPower geschützt und dürfen nicht veröffentlicht werden, daher werden diese ausgelassen, des Weiteren würden diese den Rahmen der Diplomarbeit sprengen. Es finden sich allerdings alle notwendigen Instrumentarien in einer allgemeinen Art, um beschriebene Problemstellungen erfolgreich zu lösen.

Die Essenz des ACC liegt in seiner Robustheit, Einfachheit und Funktionalität - es wurde der Ansatz des Leitsatzes „Form folgt Funktion“ gewählt. Dem Leitsatz entsprechend ist diese Software für die Fertigung erschaffen worden und befindet sich ohne übermäßige kosmetische Eingriffe im Produktivbetrieb.

Nach Abschluss der Lektüre dieser Diplomarbeit sollte es für den Leser kein Problem mehr darstellen, eine Software gedanklich grundlegend aufzubauen, die Daten in einer Datenbank speichert, die automatisierte Belastungen und Messungen an Hardware vornimmt, die Daten von µC ausliest und diese beschreiben kann und die dem Anwender der Software einen reibungslosen und unterstützenden Arbeitsfluss ermöglicht. Dem Leser sind mithilfe aktueller Technologien Möglichkeiten einer gegenwärtigen Programmierung aufgezeigt, um eigene Softwareprojekte umzusetzen und somit dem Menschen helfend beiseitezustehen.

1.1 Einbettung und Kontext der Arbeit

Die Schlüssel zur Fehlerfreiheit, Erhöhung des Arbeitsschutzes, Gewährleistung der Nachverfolgbarkeit und die Reduktion der Durchlaufzeit und somit Kostenreduktion liegen in der Automatisierung folgender Kernelemente:

- Starten und Beenden der Last, diese belastet Akkupacks entsprechenden der konfigurierten Parameter (Ampere, Dauer).
- Messen der Spannung anhand parametrisierter Zeitpunkte, beispielsweise messe nach 5 Sekunden Belastung die Spannung.
- Messen der Ampere zu definierten Zeitpunkten während der Belastung, beispielsweise sekündlich.
- Auslesen unterschiedlicher µC-Register, u. a. Spannung, Strom, Seriennummer, Status, relativer Ladestatus (rSoC), Zyklenanzahl und Gesundheitszustand (SoH).
- Protokollierung der erhobenen Werte in einer Datenbank zur späteren Analyse und Nachverfolgbarkeit.
- Vergleich von Ist-Werten mit vordefinierten Sollwerten und entsprechende Aussortierung der Akkumulatoren, sollten diese über- oder unterschritten worden sein.

Vor der Einführung des ACC wurden weite Teile dieser Elemente manuell und nach Gutdünken abgehandelt. Beispielsweise wurde das Belasten der Akkupacks nach Gefühl für die Dauer durchgeführt und entsprechend waren keine Standards zur Überwachung einer gleichbleibenden Qualität gewährleistet. Die Protokollierung erfolgte in unterschiedlichen Excellisten mehr oder weniger gründlich. Register des µC wurden nur teilweise ausgelesen und wenn händisch durch die Nutzung einer nicht für die Fertigung ausgelegten Softwarelösung des µC-Herstellers. Schlussendlich wurde durch die Implementierung des ACC ein Ordnungsprozess gestartet, der sämtliche Ecken der Technik des Unternehmens erfasst.

Somit darf bei der Implementierung eines Softwareprojektes niemals die Kommunikation außer Acht gelassen werden, denn mit dieser fällt und gelingt das Projekt. Es gilt die entsprechenden Verantwortlichen zu finden und ihnen bewusst zu machen, was sie an der künftigen Softwarelösung an Vorteilen gewinnen und warum deren Mitwirken für alle Parteien fruchtbar ist. Durch diese einfache Maßnahme lassen sich geschickt Fallstricke umwinden und die Evaluierungsphase der Software beschleunigen.

Da das ACC ausschließlich in der Fertigung des Unternehmens einen Einsatzort findet, gilt es diesen und dessen Abläufe genauestens zu untersuchen und entsprechend die betroffenen Menschen in den Prozess der Softwareentwicklung einzubinden. Durch die Einbindung der Belegschaft werden Synergien für ein erfolgreiches Projekt genutzt und eine Erreichung der Ziele unumgänglich.

Es gilt sich in Erinnerung zu rufen, was der Existenzgrund für die Entstehung der Software ist, und dieser liegt in der Unterstützung des Menschen bei seinen durchzuführenden Prozessen. Entsprechend dieser Logik folgend lösen sich viele Probleme von selbst und die Anwender nehmen die Software an.

1.2 Zielsetzung

Die Firma AccuPower gab zu Beginn des Projektes folgende Ziele bzw. Arbeitspakete vor, um aus dem Programm ACC einen Nutzen zu generieren:

1. Arbeitspaket „Prozessor auslesen und schreiben in die Datenbank“
   Über den I2C- oder SPI-Bus sollen automatisiert Register aus dem jeweiligen Akkuprozessor ausgelesen werden. Zu den Prozessoren gehören der bq78350-R1, bq78350-R2, bq34z100-G1, bq40z50-r1 und der bq40z80. Werte aus den entsprechenden Registern sind u. a. Spannung, Strom, Zyklenanzahl, relativer Ladezustand und die Seriennummer. Diese werden nach dem Auslesen in die Datenbank übertragen. Zusätzlich muss ein Seriennummernkreisverwaltungssystem integriert werden, da jeder Kunde entsprechend des erstandenen Akkumodells einen eigenen Seriennummernummernkreis erhält.

2. Arbeitspaket „Prozessor beschreiben“
   Das ACC soll in der Lage sein, automatisiert die genannten Prozessoren mit einem neuen Goldenimage zu beschreiben, des Weiteren ist es dazu notwendig, die Akkus zu entsperren und wieder zu sperren, dieser Vorgang entspricht einem Firmwareupdate und anschließender Konfektion für den Produktiveinsatz beim Kunden vor Ort.

3. Arbeitspaket „Embedded System - IoT Entwicklung“
   Das ACC soll in Python übersetzt werden und entsprechend auf einer Linuxdistribution zum Einsatz kommen. Dadurch wird es in weiterer Folge möglich, eine Blackbox zu bauen, die Behandlungsarten automatisiert durchführt und eine Bedienung durch den Menschen überflüssig zu machen. Im Optimalfall werden zusätzlich nach der Übertragung des Goldenimages automatisierte Spannungs- und Stromkalibrierungen durchgeführt.

4. Arbeitspaket „Integration in das ERP-System“
   Das ACC soll durch ein Odoomodul abgelöst werden, um direkt in das ERP-System integriert zu werden. Dadurch ist es möglich, Fertigungsaufträge direkt aus dem ERP-System zu planen und durchzuführen [6].

Diese 4 Ziele bzw. 4 Arbeitspakete sprengen den Umfang von der Diplomarbeit, daher wird Arbeitspaket 1 abgehandelt und Arbeitspaket 2 angedeutet , und aus diesen lässt sich folgende Forschungsfrage ermitteln:

„Wie kann das in C# programmierte AccuControlCenter dahingehend erweitert werden, um Daten aus den Registern des Akkumulatorenprozessors auszulesen, diese in einer Datenbank zu speichern, für den Menschen leserlich aufzubereiten und automatisierte Belastungstests durchzuführen?“

Im anschließenden Kapitel wird eine Übersicht über die Arbeit gegeben und wie diese die vorangegangene Forschungsfrage beantworten wird.

1.3 Aufbau der Arbeit

Die Diplomarbeit gliedert sich in 6 Hauptkapitel, diese werden wie folgt zusammengefasst und beschrieben:

1. Einleitung
   In der Einleitung wird ein grober Überblick über die Arbeit gegeben und was den Leser erwartet. Abgedeckt werden u. a. die Zielsetzung, der Kontext, in dem die Arbeit erfolgt und eine Beschreibung dessen, was die Arbeit nicht ist.

2. Grundlagen
   Innerhalb der Grundlagen wird das theoretische Wissen, welches zur Bewältigung der Forschungsfrage von Nöten ist und entsprechend zur Programmierung des ACC geliefert, ebenfalls findet sich ein Überblick über die Software und dessen Graphical User Interface, kurz GUI. Technische Grundlagen umfassen Wissen über die Thematiken: Datenbanken, C#, Netzwerk- und Busprotokolle.

3. Planung
   Die Planung umreißt das notwendige Management der Aufgabenverwaltung, der Kommunikation mit den am Projekt beteiligten Mitarbeitern, als auch Anforderungen an die Software und einen Überblick über die Systemarchitektur.

4. Implementierung
   Im Kapitel Implementierung wird eine Einführung in die Bedienung des ACC geliefert, um anschließend Prozesse anhand von Prozessleitbildern zu erklären. Nach dem Verständnis des umzusetzenden Prozesses wird der dafür benötigte Quellcode beschrieben und die Einbettung innerhalb des ACC aufgeführt. Dieses Kapitel enthält jene Teile, die zur praktischen Umsetzung des ACC notwendig sind, des Weiteren finden sich Hinweise auf verwendete APIs und zusätzliche Hardware.

5. Deploymentprozess
   Das Veröffentlichen, Installieren und Testen ist Teil dieses Kapitels und bildet die Grundlage für eine für den Produktivbetrieb geeignete Software. Hier finden sich auch Hinweise über die Versionierung der Software und eine dementsprechende korrekte Zuordnung beim Debuggen.

6. Fazit
   Eine Reflexion über die Arbeit, deren Zusammenfassung und einen Ausblick in die zukünftigen Erweiterungen findet sich im Kapitel Fazit. Dies betrifft u. a. die Arbeitspakete 2, 3 und 4, welche im vorangegangenen Unterkapitel Zielsetzung genannt sind.

2 Grundlagen

Das Kapitel Grundlagen wird die Software ACC anhand ihrer GUI vorstellen und diese aus der Perspektive des Anwenders überblicksmäßig beleuchten und erklären. Im Unterkapitel Technische Grundlagen wird anschließend der benötigte Technologiestack erörtert und dessen Funktion erklärt.

2.1 Überblick über die Software

Das Graphical User Interface des AccuControlCenters basiert auf WPF. WPF bedeutet Windows Presentation Foundation und ist in Kombination mit XAML, einer XML-ähnlichen Darstellungssprache, der moderne Weg, um C# Applikationen grafisch aufzubereiten. Seit 2014 hat Microsoft bekannt gegeben, keine Erweiterungen mehr für WinForms zu entwickeln, was ein weiterer Grund war, auf WPF zu setzen [7] [8].

Im nachfolgenden Bild wird der Tab „Behandlung“ des ACC dargestellt:

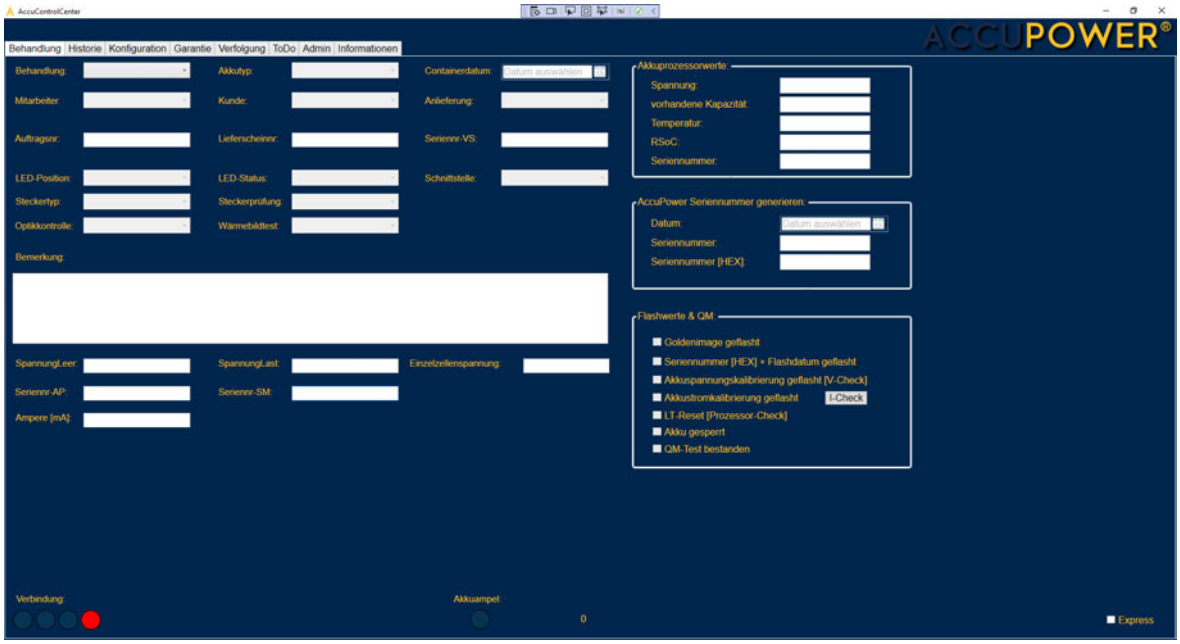


Abbildung 2.1: ACC: Abbildung der GUI vom Build 2.5.2.30928

Das ACC besteht aus 8 Tabs oder Registerkarten, diese haben im groben Überblick folgende Funktion:

- Behandlung
  Der zusehende Tab „Behandlung“ entspricht dem Kernelement des ACC, hier werden die Akkus getestet, konfektioniert und im System erfasst. Die meiste Zeit verbringt der Bediener des ACC in diesem Tab.

- Historie
  Im Tab „Historie“ wird eine Datenbankabfrage gestartet, diese stellt nach erfolgreich empfangender Antwort des Datenbankservers die letzten im System eingetragenen Akkus dar. Dieser Tab dient der Kontrolle, ob alle getesteten Akkus im System erfasst wurden.

- Konfiguration
  Innerhalb des Tabs „Konfiguration“ lassen sich die Verbindungen zu den benötigten Geräten herstellen, dazu gehören das Voltmeter, das Amperemeter und die Last. Für das Auslesen des I2C-Buses wird eine Verbindung zum Aardvark benötigt, des Weiteren ist es möglich, hier Datenexports als Exceldatei zu generieren.

- Garantie
  Sollte ein Kunde anfragen, ob auf seinen Akku noch Garantie- oder Gewährleistungsansprüche bestehen, kann der Verkauf den Tab „Garantie“ verwenden. Dieser liefert anhand der vergebenen Seriennummer entsprechend Auskunft über etwaige noch bestehende Ansprüche des Kunden.

- Verfolgung
  Der Verfolgungstab dient der Nachverfolgbarkeit - hier können sämtliche im Unternehmen durchlaufene Stationen des Akkupacks angezeigt werden. Die Notwendigkeit dieses Tabs besteht darin, dass Akkupacks 2 unterschiedliche Seriennummern innehaben und dementsprechend eine etwas komplexere Abfrage notwendig ist, um wirklich sämtlich durchlaufenen Stationen bzw. Bearbeitungsschritte angezeigt zu bekommen.

- ToDo
  Da bei Akkupacks die Gefahr der Tiefentladung besteht und diverse andere Thematiken ist es möglich, im Tab „ToDo“ eine Übersicht über gefährdete Akkupacks im Lager zu erhalten, um diese beispielsweise entsprechend nachzuladen. Es finden sich hier auch sämtliche Akkupacks, die gewisse Grenzspannungen unter- oder überschritten haben und somit nicht für den Produktiveinsatz einsetzbar sind.

- Admin
  Der Admintab kann nur über die korrekte Eingabe eines Passworts im Konfigurationstab angezeigt werden. Die Funktionalität besteht darin, Tabellen auf dem Datenbankserver direkt aus dem ACC zu bearbeiten, beispielsweise können Kunden und Mitarbeiter angelegt, gelöscht oder aktualisiert werden. Auch kritischere Daten können bearbeitet werden, beispielsweise Nummernkreise und Passwörter zum Sperren und Entsperren des Akkupacks.

- Informationen
  Die Versionsnummer bzw. Buildnummer des ACC und weitere Informationen finden sich im Tab „Informationen“.

Eine vertiefende Beschreibung und Erläuterung der angezeigten Elemente findet sich im Kapitel Design und dessen Unterkapitel.

2.2 Technische Grundlagen

Im Kapitel Technische Grundlagen wird eine überblicksmäßige Darstellung aller relevanten Technologien präsentiert, die für die Implementierung des ACC erforderlich sind. Dabei werden die historische Entwicklung, die Funktionsweise und die Anwendungsgebiete dieser Technologien im Kontext der Software untersucht.

2.2.1 C#

C# wurde im Jahr 2000 im Rahmen der .NET-Strategie veröffentlicht und orientierte sich stark an der Programmiersprache Java. Ziel war es, eine einfache, moderne und objektorientierte Programmiersprache zu schaffen. Bis zum Jahr 2023 wurden 11 Versionen von C# veröffentlicht und jede Version brachte einige Neuerungen mit sich. Bemerkenswert ist u. a., dass der Compiler „Roslyn“ veröffentlicht wurde [9] [10].

Weitere Merkmale von C# sind:

- Asynchrone Programmierung und Multithreading
- Strenge Typisierung
- Lambda Ausdrücke und LINQ (Language Integrated Query)

Software, welche mittels C# programmiert wird, wird für den Computer und dessen Prozessorarchitektur zu erst in eine Zwischensprache übersetzt (Common Intermediate Language) und anschließend zur Laufzeit durch die Common Language Runtime in Maschinensprache übersetzt und ausgeführt (EXE-Datei), dadurch gibt es keinen Performanceverlust im Vergleich zu Programmen, welche beispielsweise in Python oder JavaScript erstellt und zur Laufzeit interpretiert werden. Ein weiterer Vorteil liegt in der großen Community, welche sich um C# gebildet hat und entsprechend einfach ist es Unterstützung beim Programmieren zu bekommen [11].

Das AccuControlCenter ist zur Gänze in C# programmiert und benötigt keine andere Programmiersprache. Bei Beginn des Projekts gab es Überlegungen, die Software in JavaScript umzusetzen, diese Gedanken verflüchtigten sich aber bald, da alle Computer in der Firma mit dem Windows Betriebssystem ausgestattet sind und dadurch die Wartung eines Node.js-JavaScript-Servers eingeführt hätte werden müssen, wodurch zusätzliche Ressourcen benötigt gewesen wären.

Des Weiteren basiert der gegenwärtige Technologiestack der Firma AccuPower hauptsächlich auf Microsoft-Produkten, insbesondere ist Visual Basic stark vertreten. Dies vereinfacht das Finden künftiger Programmierer, da ein einheitlicher Technologiestack genutzt wird. Der aktuell eingesetzte Datenbankserver ist ebenfalls ein Microsoft-Produkt, wodurch eine unkomplizierte Anbindung mittels Treibern möglich ist. Es muss kein zusätzlicher Aufwand betrieben werden, da sich die verwendeten Technologien nahtlos miteinander integrieren lassen.

2.2.2 MS SQL-Datenbank

Der Microsoft SQL Server in der Version 1.0 wurde bereits im Jahr 1989 von Microsoft in Zusammenarbeit mit Ashton-Tate und Sybase veröffentlicht. Er ist ein vollständiges Datenbank Managementsystem, kurz DBMS, und kann sowohl Benutzer und deren Rechte verwalten als auch Datenbanken und Tabellen erzeugen, modifzieren und löschen. Die Datenbanken gehören dem relationalen Datenbanksystem an und sind vergleichbar mit einem Exceltabellenblatt mit seinen Spalten und Reihen [12].

Die MS SQL-Datenbank beinhaltet unterschiedlichste Daten, die zur Bedienung des ACC benötigt oder die vom ACC erzeugt und abgelegt werden. Stand 12.04.2023 besteht die Datenbank aus 14 Tabellen, 6 Sichten und hat eine Gesamtgröße von rund 310 MB.

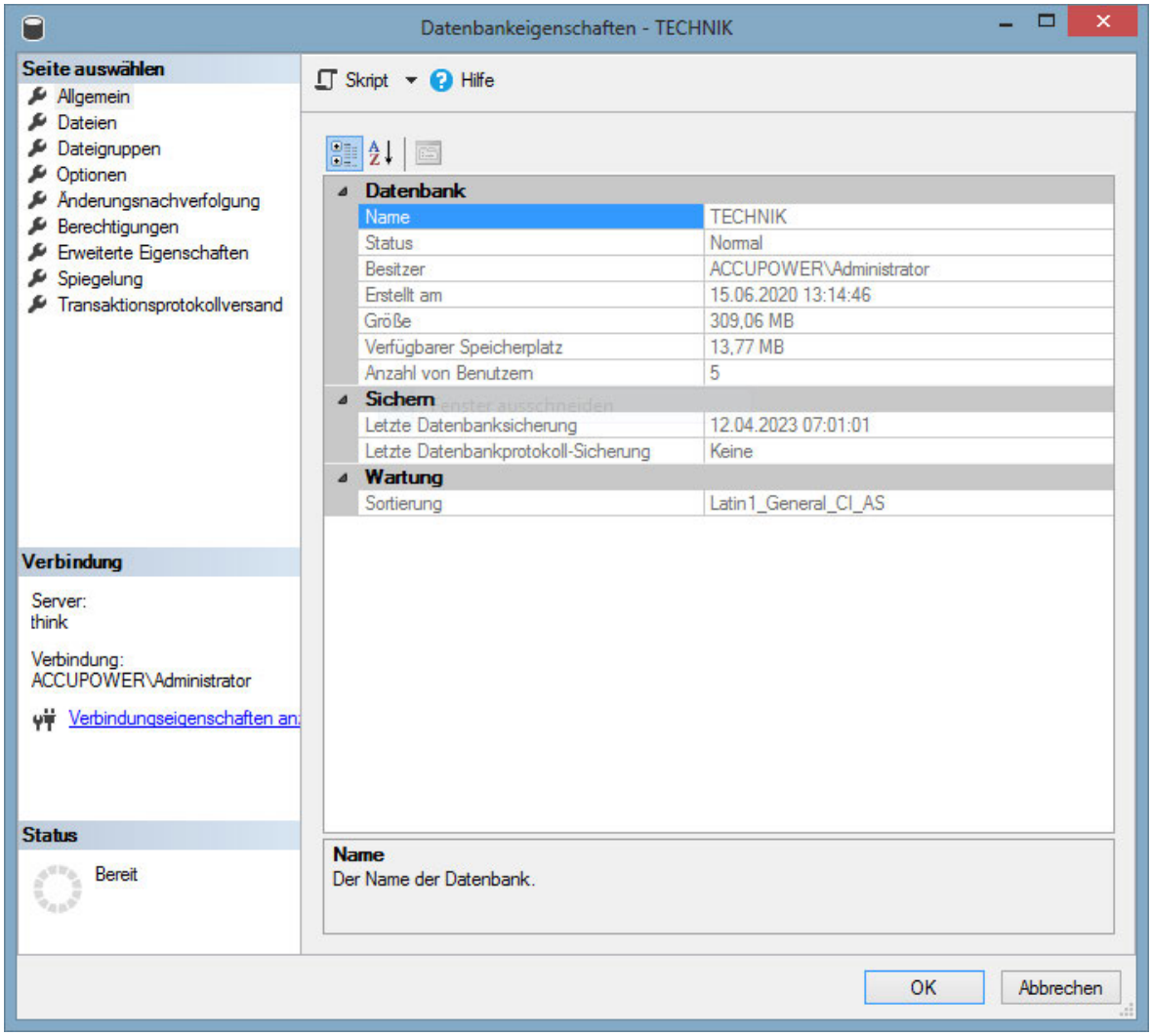


Abbildung 2.2: ACC-Datenbank: Metadaten und Datenbankeigenschaften

Da das aktuelle Enterprise Ressource Managementsystem, kurz ERP, von AccuPower einen MS SQL-Server benötigt und dadurch die entsprechenden Lizenzen vorhanden sind, sowie die Wartung und Konfiguration bereits in das IT-System integriert ist, ist die Entscheidung bezüglich des Datenbankservers sehr einfach gewesen.

2.2.3 Ethernet

Das Ethernetprotokoll basiert auf dem in den 1960er-Jahren in Hawaii in Entwicklung befindlichen Alohanet, welches an der Universität von Hawaii entwickelt wurde. 10 Jahre später in den 1970er-Jahren, begann die Entwicklung bei Xerox in Palo Alto, Amerika. Es dauerte allerdings weitere 10 Jahre, bis sich Ethernet zu einem marktreifen Produkt entwickelte und begann, die Kommunikationswege zu revolutionieren. Heute ist das Ethernetprotokoll im Standard 802.3 vom IEEE definiert und dieses stellt gleichzeitig die am meisten in Verwendung befindliche Version in Unternehmens- und Heimnetzwerken dar [13] [14].

Alle Geräte in einem Ethernet-Netzwerk bekommen eine eindeutig identifizierbare Adresse zugewiesen, die sogenannte MAC-Adresse. Mittels hochfrequenten Elektroniksignalen werden Nachrichten zwischen den Teilnehmern ausgetauscht und durch den CSMA/CD (Carrier Sense Multiple Access/Collision Detection) Algorithmus fehlerfrei ausgetauscht. Der Netzwerkaufbau kann beispielsweise als Stern oder Bus realisiert werden [14].

Das Ethernetprotokoll befindet sich im OSI-Schichtenmodell in der Schicht 1, dies entspricht dem physischen Layer bzw. der Übertragungsschicht und bildet das Fundament für den Netzwerkverkehr. Der Zweck dieser Schicht liegt in der Übersetzung von analogen Signalen in digitale Bits, um diese dann für Maschinen lesbar zu machen [15].

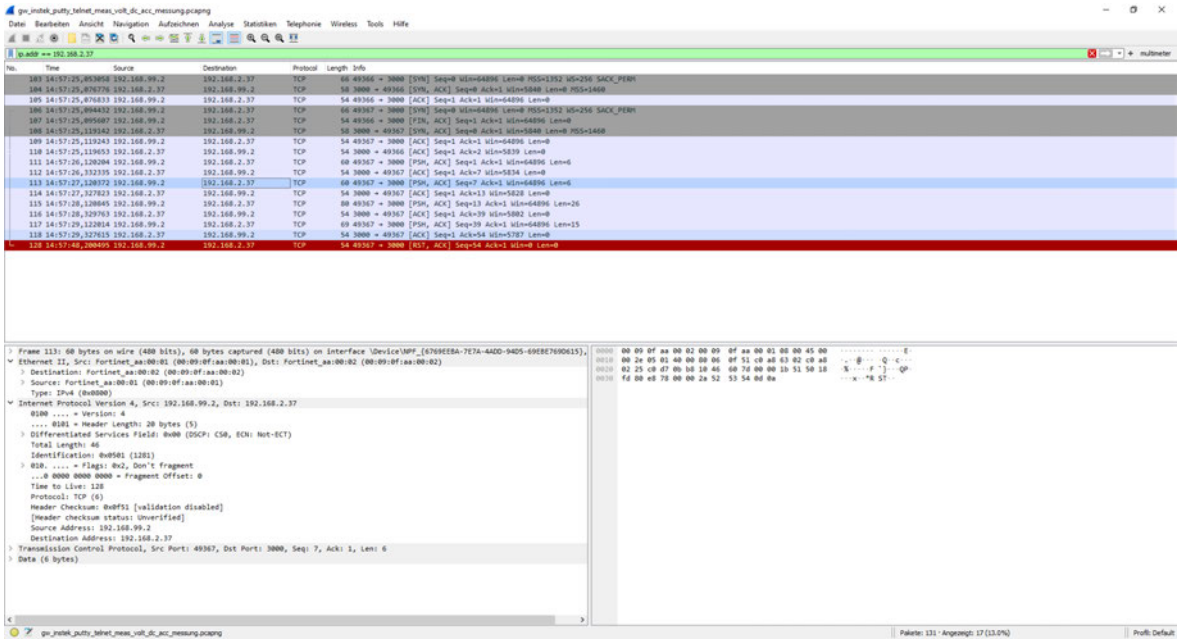


Abbildung 2.3: Debugging einer Telnet SCPI Sitzung um Daten von einem Multimeter auszulesen

In Abbildung 2.3 wird mittels einer VPN-Verbindung versucht, den Grund für einen Verbindungsabruch zu einem Multimeter zu ermitteln. Weitere Informationen dazu finden sich im Kapitel Einbindung der Multimeter und der Last. Das ACC nutzt das Ethernetprotokoll, um Daten in die Datenbank zu transferieren, aus dieser zu laden und um Messgeräte oder elektronische Lasten zu steuern. Somit stellt es einen essenziellen Bestandteil der benötigten Softwareabhängigkeiten dar.

2.2.4 SCPI

SCPI steht für „Standard Commands for Programmable Instruments“, in die deutsche Sprache übersetzt bedeutet dies „Standardbefehle für programmierbare Instrumente“. Ziel von SCPI ist es einen Standard zur Steuerung von unterschiedlichen Herstellern erzeugten Mess- und Testgeräten zu implementieren. Die Entwicklung begann in den 1990er-Jahren und im Jahr 1992 wurde der einheitliche Standard eingeführt [16].

Die Befehle sind in einer hierarschen Struktur angeordnet und basieren auf dem ASCII-Zeichensatz, hier ein Beispiel:

```
OUTPut:
        SYNC {OFF|0|ON|1}

        SYNC:
              MODE {NORMal|CARRier}
              POLarity {NORMal|INVerted}
```

Quelltext 2.1: Beispiel des hierarchischen Aufbaus von SCPI-Befehlen

OUTPut entspricht der Wurzel, dem primären Schlüsselwort, SYNC der nächsttieferen Struktur oder dem sekundären Schlüsselwort und MODE und POLarity sind hierarchisch die niedrigsten Schlüsselwörter. Mittels Doppelpunkt (:) werden hierarschische Schlüsselwörter getrennt bzw. der Beginn der nächsten Ebene angekündigt [17].

Der SCPI-Standard wird über unterschiedliche Kommunikationsschnittstellen in der Praxis angewendet dazu gehören u. a. USB, LAN, RS-232. Das ACC verwendet ausschließlich die LAN Schnittstelle, um mit den Geräten zu kommunizieren. Meist bieten die Geräte einen Telnet-Server an, der auf den Messgeräten konfiguriert wird und anschließend erfolgt die Kommunikation mittels Ethernetprotokoll um das Gerät über SCPI-Kommandos zu steuern.

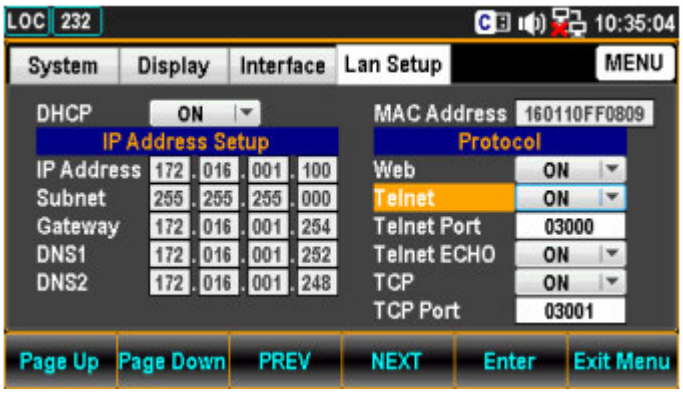


Abbildung 2.4: Einrichtung des Telnet-Servers auf dem GWInstek GDM-9061 [vgl. 18, S. 240]

Eine Beispielkonfiguration eines Telnetservers über das Hardwaremenü des GWInstek GDM-9061 ist in der Abbildung 2.4 zu sehen. So kann konfiguriert werden, ob der Telnetserver gestartet ist, auf welchem Port der Server kommuniziert und ob eine Willkommensnachricht, ein Echo, ausgegeben wird beim Betreten des Servers. In einem erweiterten Menü lässt sich ein Passwort konfigurieren oder deaktivieren.

2.2.5 I²C

Im Jahr 1982 fand der Release von I²C statt und im Jahr 1992 wurde die Version 1.0 herausgegeben. Der Entwickler des I²C-Protokolls ist NXP Semiconductors bzw. früher Philips Semiconductors. Die Abkürzung steht für Inter-Integrated Circuit, in die deutsche Sprache übersetzt „Inter-Integrierter Schaltkreis“ und bildet eine Kommunikationsschnittstelle zwischen unterschiedlichen, in der Peripherie befindlichen, intelligenten elektronischen Komponenten. Der Standardmodus unterstützt Geschwindigkeiten von 100 kbit/s und kann durch eine Anpassung auf den Fast-mode auf eine Geschwindigkeit von 400 kbit/s erhöht werden und erreicht seine höchste Übertragungsgeschwindigkeit im High-Speed-mode mit bis zu 3,4 Mbit/s. Mittels der universellen Übertragungsrichtungsmethode kann im Gegensatz zu den vorhingenannten Geschwindigkeitsmodi, welche über die bidirektionale Datenübertragungsmethode arbeiten, der Ultra Fast-mode verwendet werden und dieser ermöglicht eine Datenübertragung von bis zu 5 Mbit/s [vgl. 19, S. 1 - 5].

Der I²C-Bus besteht aus 2 Leitungen, der Taktleitung und der Datenleitung. Erstere übergibt den vom Master erzeugten Takt an alle mit dem Bus verbundenen Slaves und kann ausschließlich vom Busmaster Amplituden steuernd verwendet werden. Durch diese Definition ist die Übertragungsrichtung vorgegeben. Die Datenleitung hingegen steht allen Teilnehmern schreibend zur Verfügung, sofern die Busspezifikationen eingehalten werden (der Master steuert - Slaves antworten nur nach Aufforderung) [20].

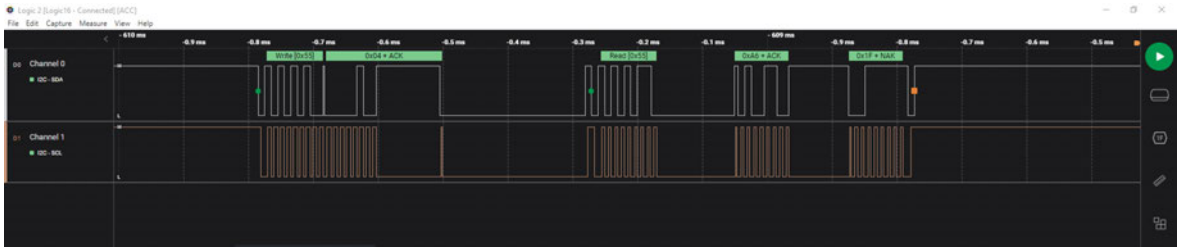


Abbildung 2.5: Abfrage des Kapazitätsregisters eines bq34z100G1-Prozessors auf dem I²C-Bus

In der Abbildung 2.5 ist zu sehen wie eine Abfrage über den I²C-Bus elektronisch aussieht. Channel 0 entspricht der Datenleitung SDA und Channel 1 der Taktleitung SCL. Das ACC sendet eine Abfrage an einen 8S2P-Akku und dessen Prozessor, den bq34z100G1. Der Hex-Wert 0x55 entspricht der I²C-Adresse des Akkupacks und der Hex-Wert 0x04 dem zu lesenden Register. Nach einer Bearbeitungszeit des Prozessors von ca. 0,3 ms wird der Wert des entsprechenden Registers als 2 Byte Rückgabewert, Hex 0xA6 und Hex 0x1F, übergeben. Das Ergebnis erhält man durch eine Rückrechnung über das Little-Endian-System und einer anschließenden Konvertierung in das Dezimalsystem.

Eine vertiefende Betrachtung findet sich im Kapitel Aardvark. In diesem werden anhand mehrerer Beispiele die Berechnung und der Empfang von der Temperatur, des Flashdatums und des Akkuzustandes aufgezeigt, wie sich der Bus verhält und wie die Signale digital und analog aussehen.

2.2.6 SMB

Der System Management Bus wurde 1995 von Intel veröffentlicht und stellt eine Weiterentwicklung und Vereinfachung des I^2C-Buses dar. Ziel ist es, ein Kommunikationsprotokoll für das Batteriemanagement bereitzustellen, welches von intelligenten SoCs (System on a Chip) verwendet werden kann [21].

Zu den größten Unterschieden zwischen I^2C und SMB gehören folgende:

- Geschwindigkeit:
  SMB hat eine geringere Übertragungsrate als I^2C, diese liegt bei 10-100 kbit/s in der 100 kHz Klasse, zwischen 10-400 kbit/s in der 400 kHz Klasse und zwischen 10-1000 kHz in der 1 MHz Klasse. Die niedrigste Übertragungsrate bzw. Frequenz von 10 kbit/s bzw. 10 kHz die Sekunde darf nicht unterschritten werden [vgl. 22, S. 18 Tabelle 2].

- Zeitanforderungen
  SMB hat definierte Bus-Inaktivitätszeiten, I^2C bietet dies nicht [vgl. 22, S. 17].

- Protokollkomplexität
  SMB enthält zusätzliche Funktionen, beispielsweise die Überprüfung auf fehlerhaft übertragene Pakete, dies ist in I^2C nicht implementiert [vgl. 22, S. 36].

- Spannungspegel
  SMB arbeitet mit einer Versorgungsspannung von 1.8 V bis zu 5 V [vgl. 22, S. 22]. I^2C mit 1.8 V, 3.3 V, 2.5 V oder 5 V [vgl. 19, S. 31].

- Adressierung
  Beide Protokolle verwenden eine 7-bit-Adressierung, allerdings reserviert der SMBus spezielle Adressen für Alarme [vgl. 22, S. 73].

Abbildung 2.6: 4S1P Akkupack
[23]

Der einzige Chip der den SMBus in dieser Diplomarbeit verwendet ist der bq40z50R1. Dieser ist u. a. in 4S1P Akkupacks verbaut. Anwendungsbereiche finden sich beispielsweise in der Medizintechnik.

3 Planung und praktische Umsetzung

Um den Überblick über alle Aufgaben zu behalten, wird das Projektmanagementmodul von Odoo verwendet. Odoo, ehemals OpenERP, ist eine Open-Source-Software, welche als ERP-System zum Einsatz kommt. Die Möglichkeit, Odoo On-Premises zu betreiben und dadurch selbstständig über die Datenhoheit zu verfügen, ist ein starker Anreiz. Natürlich ist es auch möglich, Odoo als Saas, Software as a Service oder als PaaS, Platform as a Service, zu betreiben [24] [25].

Die Odoo-Instanz der Fa. Webgeyer liegt auf einem Server in Frankfurt am Main, wird mit der Linux Distribution Debian betrieben und durch den Einsatz von Containern und der dazu notwendigen Virtualisierungssoftware Docker gehostet. Die Zuordnung zum korrekten Port erfolgt durch den Webserver nginx [26] [27] [28].

Das Projektmodul von Odoo ermöglicht die Verwaltung des Projektes als Kanban-Zettel. Dies bedeutet, jede Aufgabe wird zu einem Karteikärtchen und dieses wird durch unterschiedliche Stufen je nach aktuellem Aufgabenstatus gezogen, bis es entweder erledigt oder abgebrochen ist. Zusätzlich kann jeder Aufgabe entsprechend die dafür benötigte Dauer zugeordnet werden und mittels Aktivitäten lassen sich aktuelle Erinnerungen setzen, um kritische Thematiken schnellstmöglich zu bearbeiten [29].

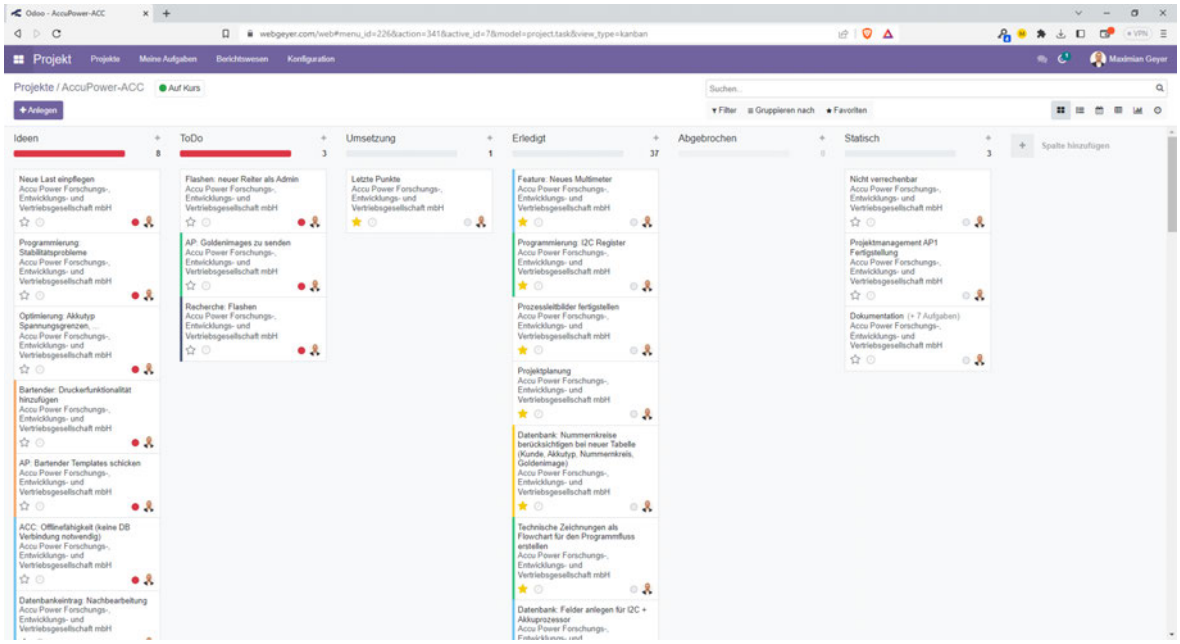


Abbildung 3.1: Odoo: Projektübersicht ACC

Abbildung 3.1 stellt das ACC Projekt in der Kanbanansicht im Odoo Projektmodul dar. Es sind in Summe 6 Spalten vorhanden: Ideen, ToDo, Umsetzung, Erledigt, Abgebrochen und Statisch. Die roten Punkte bedeuten, dass die Aufgabe aktuell blockiert ist und somit nicht bearbeitet werden kann. Mit dem Stern kann man Aufgaben als besonders wichtig markieren und die-

se werden anschließend entsprechend höher gereiht, dadurch gehen diese nicht unter bei mehreren Aufgaben. Eine weitere Hervorhebung der Aufgaben ist durch unterschiedliche Farben möglich.

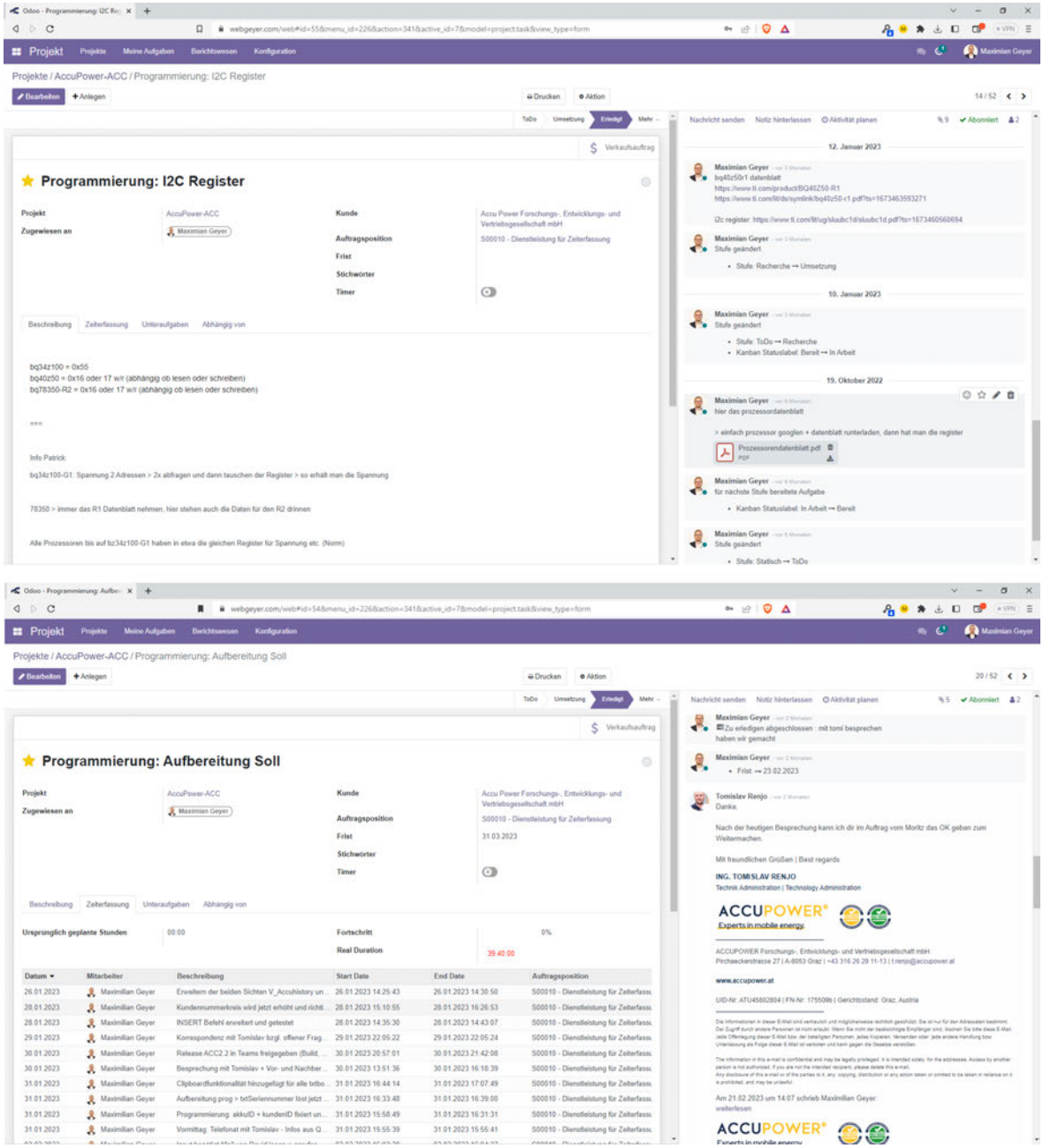


Abbildung 3.2: Odoo: Überblick Aufgabenansicht

In Abbildung 3.2 wird gut erkennbar, was die Vorteile einer Projektverwaltung mit Odoo sind, so kann beispielsweise der Aufgabe eine Beschreibung hinzugefügt werden, in dieser können Bilder enthalten sein, Tabellen oder allgemeine Textformatierungen. Rechts im Chatter, der Timeline, kann nachvollzogen werden, zu welchem Datum welche Tätigkeiten durchgeführt wurden, es ist möglich, extern zu kommunizieren, beispielsweise per Mail Richtung AccuPower und die Mailantwort von AccuPower wird anschließend automatisch der Aufgabe zugeordnet und intern mittels einer Notiz, diese ist dann für AccuPower unsichtbar und dient

einem selbst als Dokumentation. Zusätzlich können *.pdf, *.xlxs, *.docx, u. a. Dateien jeder Aufgabe angehängt werden und durch die Zeiterfassung ist eine Zuordnung der Dauer der jeweiligen Tätigkeit und eine detaillierte Beschreibung möglich.

Das Projektmanagement von Odoo lässt eine einfache Verwaltung aller Anforderungen zu, sowohl der funktionalen als auch der nicht-funktionalen Anforderungen und entsprechende unerwartete Erweiterungen betreffend der Systemarchitektur und neuer benötigter Features, welche sich regelmäßig in einem Softwareprojekt finden.

3.1 Funktionale Anforderungen

Zu den funktionalen Anforderungen des ACC gehören neben der Nachverfolgbarkeit und Historie automatisierte Messungen und Belastungen und das Auslesen und Schreiben des µC. Die genannten Anforderungen werden in den nachfolgenden Unterkapiteln beleuchtet.

3.1.1 Nachverfolgbarkeit und Historie

Die Historie bietet eine Übersicht der letzten getätigten Arbeiten in der Fertigung und bietet dementsprechend für den Arbeiter eine Kontrollfunktion, ob die Daten gewünscht eingetragen wurden. Der Schichtleiter sieht entsprechend eine Übersicht über aktuell abgewickelte Produktionsaufträge und kann entsprechend nachjustieren, sollten gerade Aufträge mit einer wenig höheren Priorität abgewickelt werden. Auf Kundenwunsch ist die Funktionalität des Löschens einzelner Einträge für den Arbeiter möglich, um entsprechende Fehler zu korrigieren, allerdings ist eine Massenlöschung für den Anwender nicht möglich.

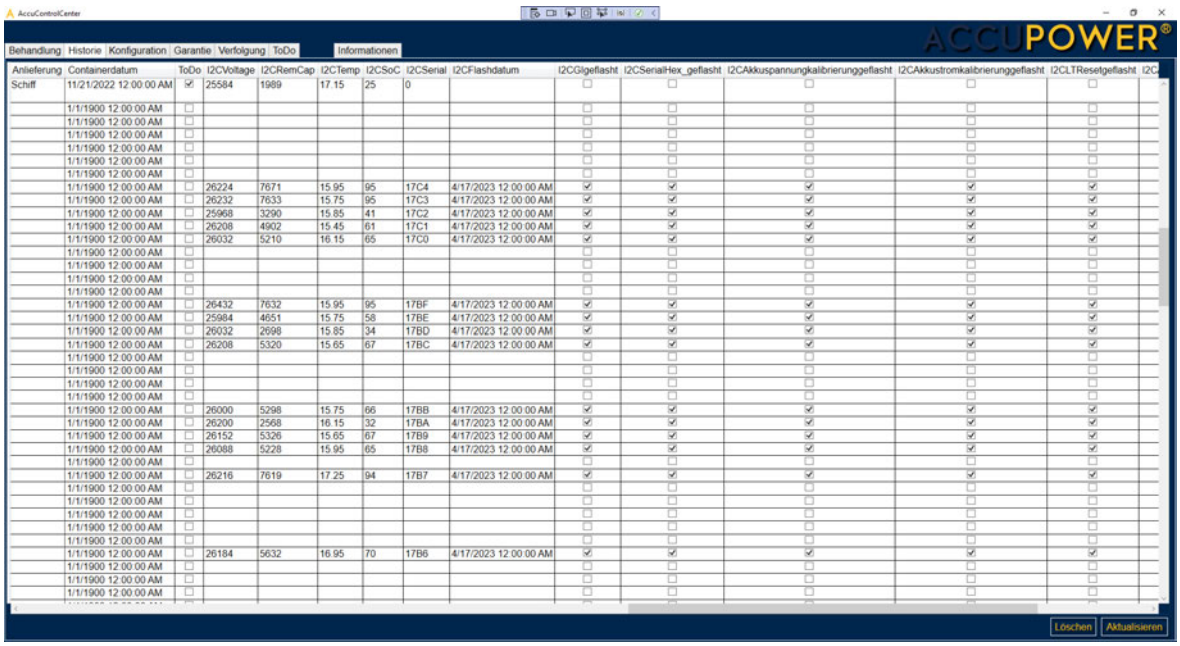


Abbildung 3.3: ACC: Registerkarte „Historie“

Wer auf die Registerkarte „Historie“ wechselt, wie in Abbildung 3.3 dargestellt, erhält eine Liste aller Behandlungsarten, sortiert nach Erstellungsdatum des Eintrags, wobei die aktuellsten als erstes abgebildet werden. Der programmatische Ablauf entspricht dem Aufruf einer Sicht,

welche auf dem SQL-Server angelegt ist. Diese Sicht lädt die aktuellsten 250 Einträge aller Tabellen und bereitet die Daten für den Menschen lesbar auf. Würden mehr Einträge geladen werden, verzögert sich aufgrund der Datenmenge die Darstellung des Tabs „Historie“.

```
SELECT TOP (250) AH.ID, AT.Kuerzel AS Akkutyp, ST.Bezeichnung AS Steckertyp,
 AH.SeriennummerSM, AH.SeriennummerAP, AH.SeriennummerVersand,
 AH.Lieferscheinnr, AH.Auftragsnr, AH.Spannung, AH.SpannungLast,
 AH.EntscheidungshilfeNotiz, I2C.Status AS Schnittstelle, LP.Position AS LEDPosition,
 LS.Status AS LEDStatus, O.Status AS Optikkontrolle, S.Status AS Steckerprüfung,
 W.Status AS Wärmebildtest, AH.Bemerkung, M.Kuerzel AS Mitarbeiter,
 BT.Kuerzel AS Behandlung, KD.Bezeichnung AS Kunde, VS.Versandart AS Anlieferung,
 AH.Containerdatum, AH.ToDo, AH.I2CVoltage, AH.I2CRemCap,
 AH.I2CTemp, AH.I2CSoC, AH.I2CSerial, AH.I2CFlashdatum, AH.I2CGI geflasht,
 AH.I2CSerial Hex geflasht, AH.I2CAkkuspannungkalibrierung geflasht,
 AH.I2CAkkustromkalibrierung geflasht, AH.I2CLTReset geflasht,
 AH.I2CAkku gesperrt geflasht, AH.I2CQM Test bestanden, AH.Erstellt
FROM dbo.accuhistory AS AH
 LEFT OUTER JOIN dbo.mitarbeiter AS M ON M.ID = AH.MitarbeiterID
 LEFT OUTER JOIN dbo.akkutyp AS AT ON AT.ID = AH.AkkutypID
 LEFT OUTER JOIN dbo.behandlungstyp AS BT ON BT.ID = AH.BehandlungsID
 LEFT OUTER JOIN dbo.kunden AS KD ON KD.ID = AH.KundenID
 LEFT OUTER JOIN dbo.i2cstatus AS I2C ON I2C.ID = AH.I2CID
 LEFT OUTER JOIN dbo.ledposition AS LP ON LP.ID = AH.LedpositionID
 LEFT OUTER JOIN dbo.ledstatus AS LS ON LS.ID = AH.LedstatusID
 LEFT OUTER JOIN dbo.optikkontrolle AS O ON O.ID = AH.OptikkontrolleID
 LEFT OUTER JOIN dbo.steckerprüfung AS S ON S.ID = AH.SteckerprüfungID
 LEFT OUTER JOIN dbo.wärmebildtest AS W ON W.ID = AH.WärmebildtestID
 LEFT OUTER JOIN dbo.versandart AS VS ON VS.ID = AH.VersandartID
 LEFT OUTER JOIN dbo.steckertyp AS ST ON ST.ID = AH.SteckertypID
ORDER BY AH.Erstellt DESC
```

Quelltext 3.1: SQL-Code der Sicht „Historie“

Wie im Quelltext 3.1 zu sehen ist, sind 13 Tabellen und und 38 Spalten in der Sicht enthalten. Für die Datenaufbereitung besteht die Möglichkeit, im Tab „Konfiguration“ mittels eines einfachen Knopfdruckes die gesamte Historie, ohne Filter, in eine Exceldatei zu exportieren.

Abbildung 3.4: ACC: Export der Historie

In der Registerkarte „Verfolgung“ ist es dem Benutzer, beispielsweise einem Entwickler, möglich, alle Stationen, die der Akkumulator im Unternehmen durchlaufen hat, nachzuprüfen und entsprechende Informationen über die durchlaufenen Stationen zu erhalten. Dadurch wird es möglich, Daten über die Tiefentladungseigenschaften der jeweiligen Akkumulatorenchemie zu sammeln und auszuwerten.

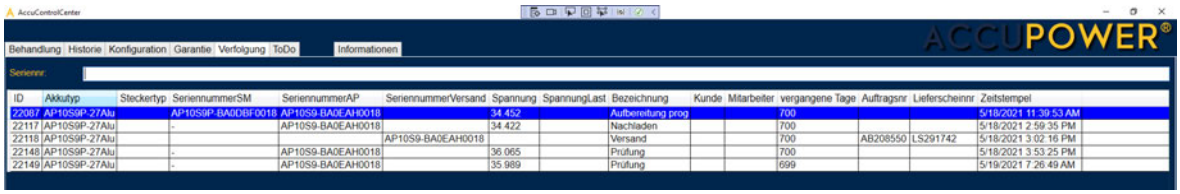


Abbildung 3.5: ACC: Registerkarte „Verfolgung“

Der in Abbildung 3.5 abgebildete Akkumulator hat 2 Prüfungen durchlaufen, wurde entsprechend nachgeladen, um anschließend für den Versand aufbereitet zu werden und das Haus zu verlassen.

3.1.2 Auslesen und Beschreiben des µC

Das ACC benötigt die Fähigkeit, Daten vom Prozessor des Akkus lesen zu können und dafür wird entsprechend die Fähigkeit des Prozessorbeschreibens benötigt, wie dies bereits im Unterkapitel Abfrage eines bq34z100G1 Registers auf dem I²C-Bus dargelegt wurde. Die Firma Texas Instruments, kurz TI, stellt dafür das Programm „Battery Management Studio“, kurz „bqStudio“, zur Verfügung.

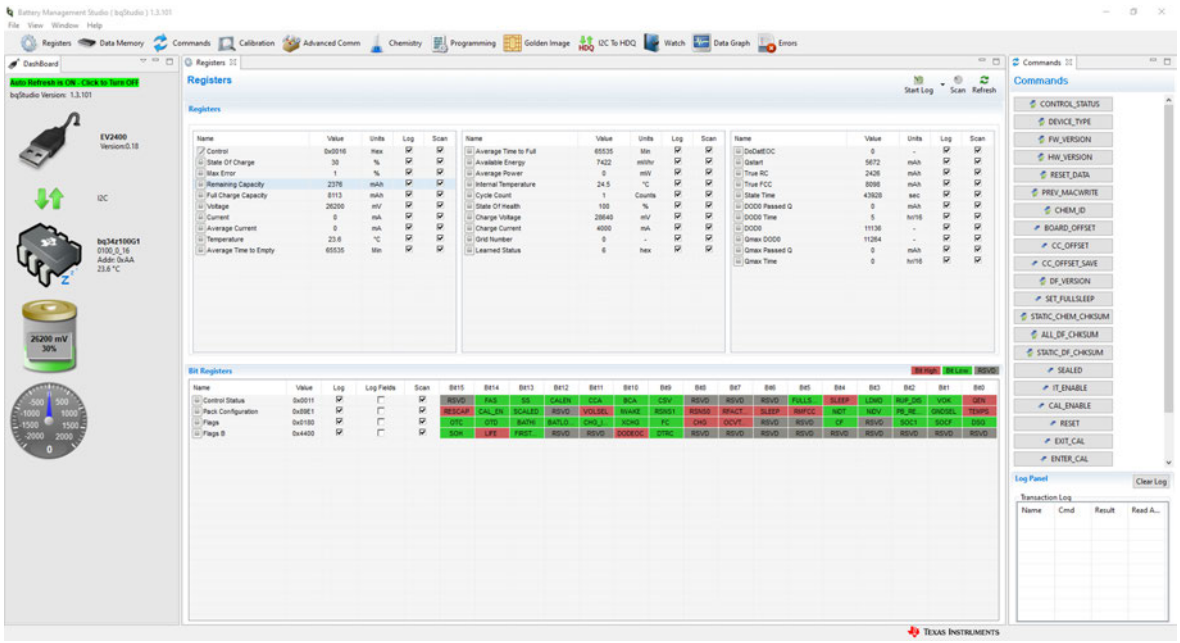


Abbildung 3.6: Battery Management Studio

In Abbildung 3.6 finden sich die Daten eines 8S2P Lithium-Eisenphosphat Akkupacks, welches mittels des bqStudios und dem I²C-Bus ausgelesen werden. Neben der aktuellen Temperatur, dem Ladezustand, der Spannung und vielen weiteren Daten ist es auch möglich, direkt im bqStudio ein Abbild der aktuellen Konfiguration, ein sogenanntes Goldenimage, zu erstellen und dieses auf weitere Akkupacks zu flashen. Eine weitere notwendige Behandlung ist die Kalibrierung des Akkupacks, diese kann auch im bqStudio durchgeführt werden. Abschließend sei noch erwähnt, dass das bqStudio eine direkte Kommunikation über den Bus erlaubt - dadurch ist ein einfaches Debugging möglich [30].

Um das bqStudio verwenden zu können, ist es notwendig, einen Interfaceadapter von TI zu erstehen, im Falle des ACC wird für die Kommunikation mit dem bqStudio ein EV2400 verwendet [31].

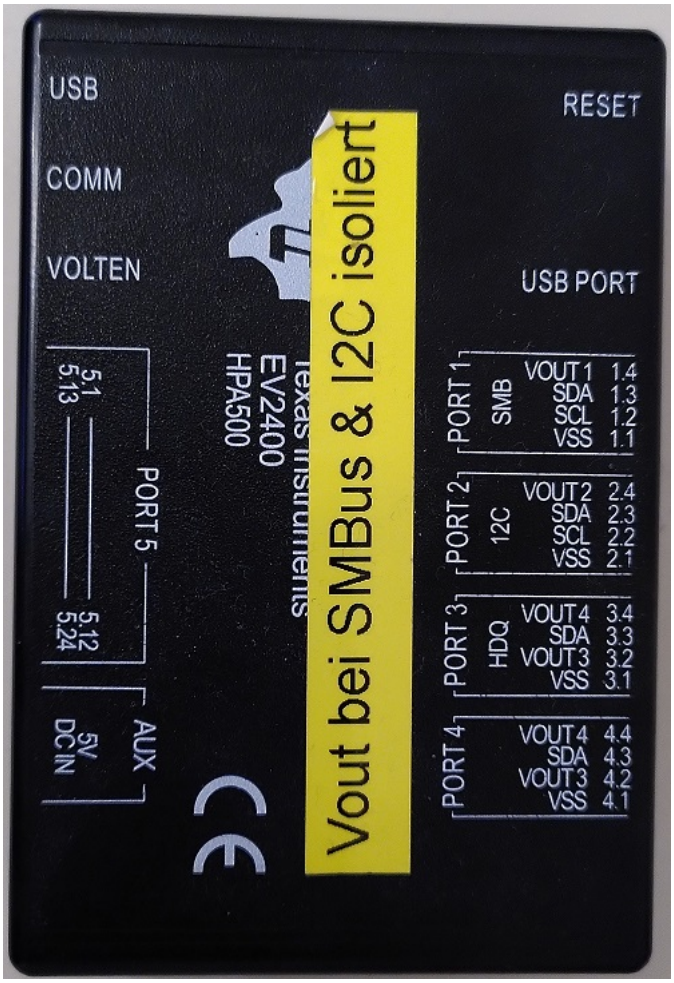


Abbildung 3.7: EV2400

Das in Abbildung 3.7 zu sehende EV2400 wird für das Debugging beim ACC verwendet. TI hält die API des EV2400 unter Verschluss, daher ist es nicht möglich, dieses Gerät für die Zwecke des ACC, welches im Grunde die Funktionalität des bqStudios in einem geringeren Umfang nachbaut, zu verwenden. Das EV2400 wird per USB-Verbindung mit dem PC verbunden und der Akku kann entweder über SMB oder I^2C verbunden werden.

Das gelbe Etikett mit dem Verweis auf die Isolierung des Vout bei SMBus und I^2C ist ein Hinweis darauf, dass bei unsachgemäßer Verwendung die Gefahr besteht, das EV2400 und den PC irreparabel zu beschädigen. Dies resultiert daraus, dass ein Kurzschluss erfolgen kann, wenn das spannungsführende Kabel nicht abgeklemmt oder isoliert wird. Es besteht die Möglichkeit, dass dabei Temperaturen erreicht werden, bei denen Kupfer schmilzt.

Um der Thematik des Reverse Engineering und der proprietären API und Software von seitens TI auszuweichen, verwendet das ACC das Aardvark, eine von Total Phase entwickelte Platine, welche denselben Funktionsumfang wie das EV2400 bietet, allerdings mit dem Vorteil, dass eine öffentliche API zur Verfügung steht. Weitere Ausführungen finden sich im Unterkapitel Aardvark.

3.1.3 Automatisierte Messdatenverarbeitung

Im vorangehenden Kapitel wurde kurz auf die Thematik der Kalibrierung verwiesen. Die Kalibrierung kann nur durch eigens kalibrierte Messgeräte erfolgen und dies ist eine der Schwächen des bqStudios. Dieses hat keine Schnittstelle zur Verfügung, um Messergebnisse automatisiert in die Kalibrierung einfließen zu lassen. Das ACC benötigt daher die Funktionalität, Daten von Messgeräten auslesen zu können, dazu gehören die Spannung und die Stromstärke.

Zu den unterstützten Messgeräten gehören:

- Keysight 34465 A

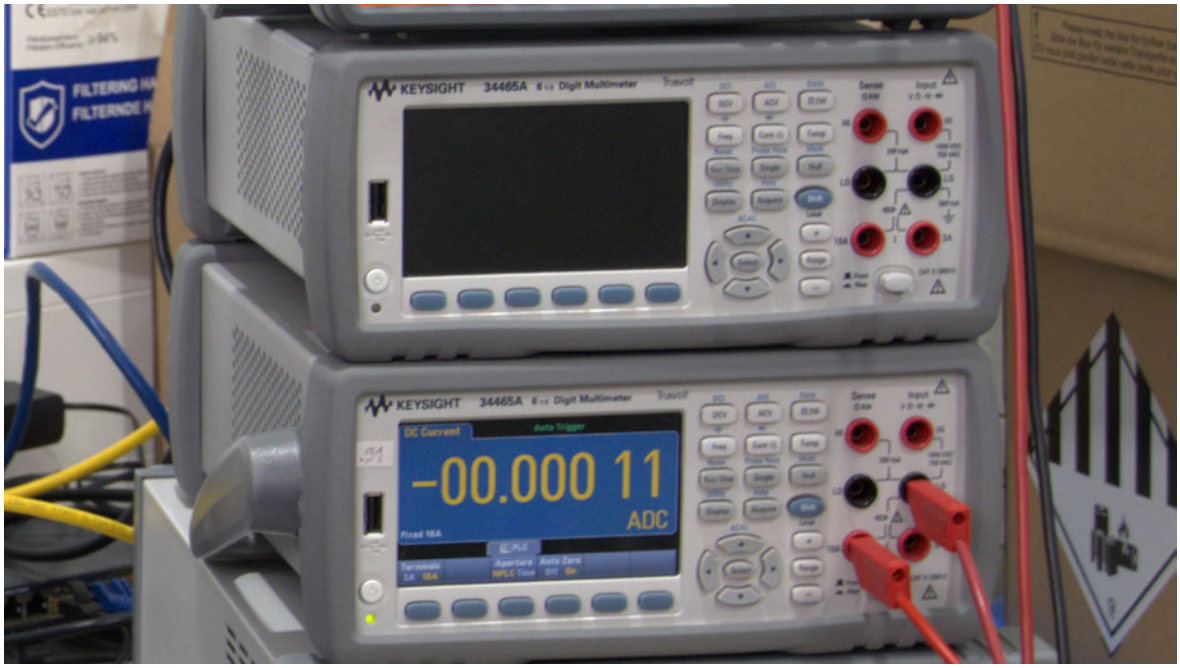


Abbildung 3.8: Keysight 34465 A

Das Keysight 34465 A war der normale Standard innerhalb von AccuPower, allerdings bekam Keysight durch die Coronapandemie Lieferkettenprobleme und konnte keine Geräte mehr liefern. Zu den Vorteilen dieses Geräts gehört vor allem die Geschwindigkeit, mit welcher SCPI-Befehle ausgeführt und deren Daten zurückgegeben werden.

- GWInstek GDM-9061

Abbildung 3.9: GWInstek GDM-9061

Das GWInstek GDM-9061 konnte als einziges Messgerät innerhalb einer akzeptablen Zeitspanne geliefert werden, daher wurde dieses gewählt. Es ist deutlich günstiger als das Keysight Multimeter, allerdings auch spürbar langsamer in der Bearbeitung von SCPI-Befehlen. Des Weiteren war ein Unterschied zwischen dem Testgerät und den Geräten, die geliefert wurden feststellbar bzgl. des Telnetservers. Die genannte Thematik löste einen deutlichen Mehraufwand beim Programmieren aus.

Im Konfigurationsbereich des ACC wird gewählt, mit welchem Messgerät die Kommunikation erfolgen soll und anschließend werden die benötigten Werte über Ethernet und SCPI-Befehle an das ACC zurückgeliefert und entsprechend von diesem für den Menschen leserlich aufbereitet.

3.1.4 Automatisierte Belastungstests

Eine weitere Anforderung, die das ACC funktional abdecken muss, ist die automatisierte Belastung der Akkupacks in einer definierten Zeitspanne und mit einer festgelegten Stromstärke. Derzeit wird die Last EA-EL 3080-60 B der Firma EA verwendet.

Die EL 3080-60 B lässt sich analog zu den Messgeräten mit SCPI-Befehlen über einen integrieren Telnetserver steuern. Die genannte Schnittstelle wird entsprechend vom ACC genutzt, um mittels Timer einen möglichst exakten Belastungstest durchzuführen. Im Anhang A lässt sich unter A.1 Prozessleitbild Eingangsprüfung ein vollständiger Prozess, der vom ACC abgedeckt wird, ansehen.

Abbildung 3.10: EA-EL 3080-60 B

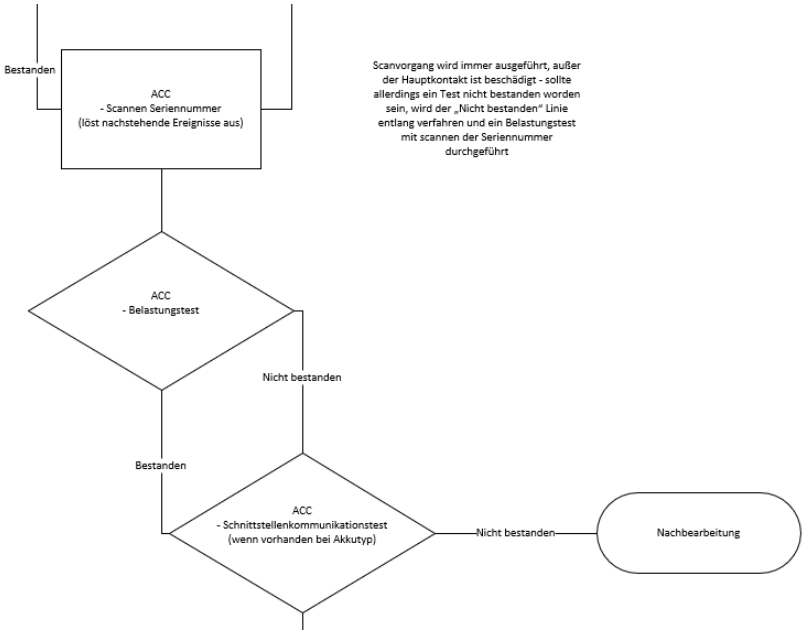


Abbildung 3.11: Ausschnitt Prozessleitbild Eingangsprüfung

In Abbildung 3.11 wird ein Ausschnitt aus dem angeführten Prozessleitbild dargestellt. Erkennbar ist, dass der Trigger bzw. der Auslöser durch das Scannen der Seriennummer bei der Behandlungsart „Eingangsprüfung" erfolgt. Dies bedeutet, dass der Anwender keine überflüssigen Mausklicks oder Derartiges benötigt, um das Programm zu bedienen. Es wird natürlich im Quellcode überprüft, ob es sich um eine gültige Seriennummer handelt. Das Scannen der Seriennummer lässt den Belastungstimer starten, dieser führt nach dem Start für 10 Sekunden eine Belastung des Akkupacks in der definierten Höhe von 10 Ampere im Falle des 8S2P-Akkus durch. Anschließend erfolgt aus den erhobenen Messwerten, ob der Akku den geforderten Anforderungen entspricht oder ob dieser einer „Nachbehandlung" unterzogen werden muss.

3.2 Nicht-funktionale Anforderungen

Anhand folgender, nicht-funktionaler, Qualitätskriterien wird die Programmierung des ACC bewertet:

1. **Benutzerfreundlichkeit:**
   Das ACC wurde mithilfe der täglichen Anwender entwickelt und entsprechend den Bedürfnissen im GUI angepasst, dadurch ist eine erhöhte Benutzerfreundlichkeit gegeben.

2. **Skalierbarkeit:**
   Das ACC kann auf mehreren Arbeitsplätzen gleichzeitig betrieben werden. Flaschenhälse sind die benötigte Hardware, um den Arbeitsplatz auszurüsten und bei steigender Nutzerzahl der SQL Server im Hintergrund.

3. **Performance:**
   Da das ACC direkt für das Windowsbetriebssystem mit Microsoft eigenen Entwicklungstools erstellt wird, ist es entsprechend performant auf Windows Betriebssystemen. Probleme gibt es einzig bei langsamen Telnetservern, allerdings kann hierauf kein direkter Einfluss genommen werden.

4. **Zuverlässigkeit:**
   Das ACC wird seit 3 Jahren täglich in der Produktion verwendet und durch die entsprechenden Tests kann eine erhöhte Zuverlässigkeit und Stabilität garantiert werden.

5. **Sicherheit:**
   Das ACC bietet 2 Sicherheitslevel, eines für den normalen Benutzer und eines für den Schichtleiter. Dieser kann auf das Backend zugreifen und dies ist nur durch das Wissen um das Passwort für den Administrationsbereich möglich.

6. **Wartbarkeit:**
   Das ACC basiert auf einem modernen Technologiestack und kann mit entsprechenden Programmierkenntnissen jederzeit an neue Bedürfnisse angepasst werden.

7. **Kompatibilität:**
   Das ACC kann auf Windows 10 und Windows 11 betrieben werden und neuere Betriebssysteme von Microsoft sollten auch kein Problem darstellen.

8. **Geschwindigkeit:**
   Das ACC ist auf Geschwindigkeit optimiert, dies bezieht sich auf mehrere Tests, um genau jene Zeitfenster abzupassen, die benötigt werden, um zu lesende Daten schnellstmöglich zu erhalten.

9. **Dokumentation:**
   Das ACC ist in mehreren Prozessleitbildern definiert, der Quellcode an den entsprechenden Stellen kommentiert, durch ein Benutzerhandbuch und diese Diplomarbeit schriftlich dokumentiert.

3.3 Systemarchitektur

Abschließend für das Kapitel Planung werden alle bisher vorgestellten benötigten Elemente in einem schematischen Aufbau dargestellt, um die Übersichtlichkeit zu bewahren.

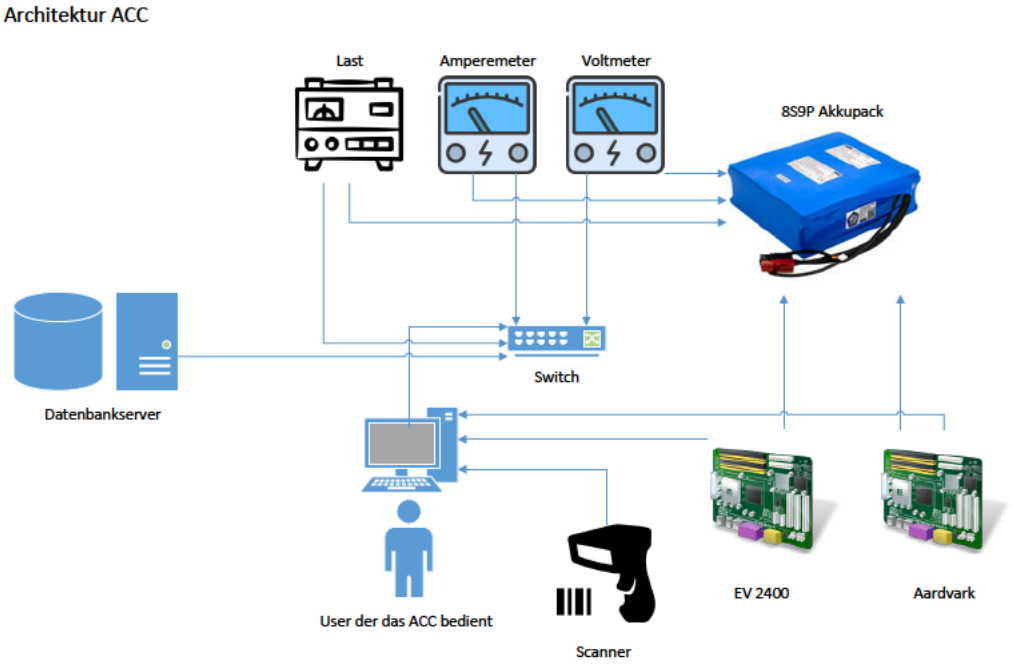


Abbildung 3.12: Schematischer Architekturaufbau des ACC
[32] [33]

Der schematische Aufbau des ACC, welcher in Abbildung 3.12 dargestellt wird, zeigt alle benötigten Komponenten, um das ACC in der Produktion verwenden zu können. Dazu gehört der Datenbankserver, auf welchem die Daten gespeichert werden, die Last, um den Akku einem Belastungstest zu unterziehen, die beiden Multimeter, welche ihre Verwendung als Ampere- oder Voltmeter finden, das EV2400 um aktuell noch nicht im ACC programmierte Funktionen durchführen zu können, das Aardvark um im ACC programmierte Funktionen durchführen zu können, den Scanner, um die Seriennummern zu scannen, einen Switch, um alle netzwerkfähigen Geräte zu verbinden, einen Akkupack in diesem Fall ein 8S9P Akku und schlussendlich den PC mit dem Bediener, der das ACC steuert.

Eine vertiefende Übersicht über die Funktionalitäten und deren Steuerung im ACC findet sich im Anhang A unter A.3 ACC Programmatik. Hierin befinden sich der Start des Programms und dessen ablaufende Funktionen und anschließend für jede Registerkarte die möglichen Operationen und deren weitere Auslöser.

4 Implementierung

Bevor eine Codezeile geschrieben wird, musste das Prozessleitbild erstellt und von der Firma AccuPower freigegeben worden sein, um eine Fehlkommunikation und die daraus resultierende fehlerhafte Programmierung zu vermeiden. Ein Prozessleitbild beschreibt den Fluss eines Prozesses, der vom ACC unterstützt wird und die Erstellung erfolgt in Kooperation zwischen AccuPower und Webgeyer. Nach der Freigabe des Prozessleitbildes wird das Flussdiagramm „Programmatik“ des ACC (Anhang A.3 ACC Programmatik) erweitert und entsprechend diesen Leitbildes werden die Codezeilen programmiert und die Funktionalität Schritt für Schritt aufgebaut. Durch dieses Vorgehen konnte bereits im Vorhinein Klarheit über die kommenden Schritte hergestellt und die Komplexität gemeistert werden. In diesem Kapitel wird ein vertiefender Blick hinter die Leitbilder und deren Bedeutung für die Programmlogik des ACC gewagt.

4.1 Design

Das Design in seiner Farbgebung ist an die Corporate Identity, kurz CI der Firma AccuPower angepasst. Das Blau entspricht der RAL-Farbe 5026 Perlnachtblau und das Gelb der RAL-Farbe 1003 Signalgelb. Die Größe des Hauptprogrammfensters ist auf 1920x1080 px optimiert und die Schriftart Arial wird durchgehend verwendet.

```
<TabItem x:Name="tabBehandlung" Header="Behandlung" Margin="-2,-2,0,0" Foreground="Black">
                <Grid Background="#FF00264D" Margin="0,123,0,3">
                    <TextBox x:Name="txtSpannung" TabIndex="7" HorizontalAlignment="Left"
                        Margin="126,369,0,0" TextWrapping="Wrap" Width="175" FontFamily="
                        Arial" VerticalAlignment="Top" Height="24" FontSize="17"/>
                    <Label x:Name="lblSpannung" Content="SpannungLeer:"
                        HorizontalAlignment="Left" Margin="10,363,0,0" VerticalAlignment="
                        Top" FontFamily="Arial" FontSize="15" Height="36" Width="111"
                        Foreground="#FFC025"/>
```

Quelltext 4.1: XAML-Code zur Einbindung der CI von AccuPower

Im Quelltext 4.1: CI AccuPower XAML-Code ist in Kombination mit der Abbildung 2.1: ACC GUI Tab Behandlung V2.5.2.30928 erkenntlich, dass der Hintergrund des Grids „tabBehandlung“ der RAL-Farbe 5026 Perlnachtblau entspricht bzw. weiterführend in HEX-Darstellung „#FF00264D“ sowie der Foreground, die Schriftfarbe des Labels lblSpannung der RAL-Farbe 1003 Signalgelb bzw. in HEX-Darstellung „#FFC025“. Des Weiteren wurde bei der Benennung von Objekten und Elementen darauf geachtet, dass der Name sprechend ist, dadurch ist eine erleichterte Zuordnung gegeben. Die nachfolgenden Unterkapitel geben Aufschluss über die Bedienung, Logik und Programmierung des ACC.

4.1.1 Konfiguration

Die Registerkarte „Konfiguration“ entspricht der digitalen Steuerungseinheit des ACC. Innerhalb dieses Tabs kann das ACC in Betrieb gesetzt werden, ungeachtet der Geräte, die am Arbeitsplatz Verwendung finden.

Abbildung 4.1: Arbeitsplatz in der Produktion

In Abbildung 4.1 wird einer von aktuell zwei Arbeitsplätzen in der Produktion dargestellt. Erkennbar sind der Scanner, das geöffnete ACC, im Gerätestapel ganz unten die Last, darüber folgend ein Keysight Multimeter, welches als Amperemeter gerade Verwendung findet, darüber ein ausgeschaltetes Multimeter von Keysight, den Abschluss bildet ein GWInstek Multimeter, welches als Voltmeter im Einsatz ist und ganz rechts im Bild ist ein Aardvark erkennbar. In dieser Darstellung wird eine Eingangsprüfung durchgeführt und der Lasttest ist nicht bestanden worden. Dies ist erkenntlich durch die rote Akkuampel und bedeutet, dass die Spannungsgrenze unterschritten wurde und dadurch der Akkupack nicht versendet werden darf. Entsprechend ist ein Nachladen des Akkupacks erforderlich.

Im Konfigurationstab wird konfiguriert, welches Voltmeter Verwendung finden soll. Diese Information bezieht sich auf den Hersteller, da ein Keysight und GWInstek Multimeter unterschiedlich angesprochen werden müssen. Des Weiteren wird die IP-Adresse des Multimeters benötigt, um eine Verbindung aufzubauen. Der Verbindungsaufbau wird anschließend mit dem Ping-Button ausgelöst und durch die entsprechenden Kreise neben der IP-Adressen-Textbox wird in Kombination mit den Textboxen unterhalb angezeigt, ob eine Verbindung zustande gekommen ist. Analog verhält es sich mit dem Amperemeter und der Last. Wenn eine Verbindung erfolgreich aufgebaut wurde, ist es möglich, durch den Button „Trennen“ die Verbindung zu lösen, dieser Button entspricht dem vorausgegangen Ping-Button. Quellcodebeispiele finden sich im Kapitel Einbinden der Multimeter und der Last. Weitere Konfigurationsmöglichkeiten bestehen in einer direkten Bedienung des Aardvark, dadurch ist es möglich, manuell Daten vom I^2C-Bus auszulesen und die Kommunikation bei korrekter Bedienung mit dem bqStudio nicht zu stören. Der technische Aspekt wird tiefer im Kapitel Aardvark beleuchtet. Durch einen Klick auf die Checkbox „Express“ ist es möglich, den automatischen

Reset der Komboboxfelder im Tab „Behandlung“ zu verhindern und dadurch schneller beim Abarbeiten der Fertigungsaufträge voranzuschreiten. Beim Aktivieren der Checkbox „Admin“ erscheint ein weiteres Textfeld, bei der Eingabe des korrekten Passwortes wird ein weiterer Tab zwischen den Tabs „ToDo“ und „Informationen“ angezeigt. Dazu finden sich weitere Informationen im Kapitel Administrationsmenü. Der letzte Button „Historie“ löst den bereits im Kapitel Grundlagen beschriebenen Export der Historie aus. Durch den festgelegten Pfad ist eine Sicherheitsmaßnahme vorhanden, die dafür Sorge trägt, dass nur Computer, welche innerhalb des AccuPower Firmennetzwerkes sind, einen Export auslösen können.

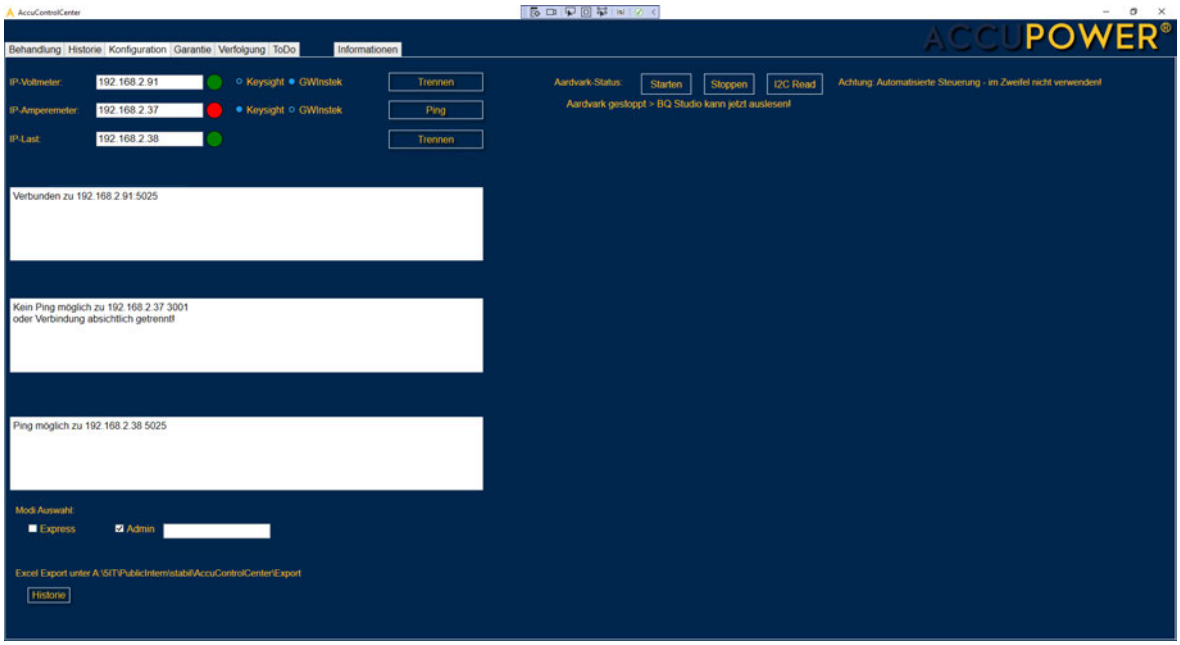


Abbildung 4.2: ACC: Registerkarte „Konfiguration“

Abbildung 4.2 stellt ein typisches konfiguriertes ACC für die Behandlungsart Eingangsprüfung dar. Als Voltmeter kommt ein Keysight 34465 A zum Einsatz unter der IP-Adresse „192.168.2.91“. Durch den grün eingefärbten Kreis neben der eingegebenen IP-Adresse ist sofort erkenntlich, dass eine Verbindung erfolgreich aufgebaut wurde. Weitere Informationen lassen sich der ersten Konsolentextbox entnehmen, diese gibt den Logeintrag „Verbunden zu 192.168.2.91:5025“ aus und der Button „Ping“ ist bereits dem Button „Trennen“ gewichen. Das Amperemeter wird für die Eingangsprüfung nicht benötigt, allerdings wurde versucht, zu einem Gerät eine Verbindung aufzubauen. Dass diese Verbindung nicht zustande gekommen ist, lässt sich leicht an dem rot eingefärbten Kreis neben der IP-Adresse für das Amperemeter, „192.168.2.37“, erkennen und die Konsolentextbox, welche dem hierarchischen Aufbau der IP-Adressen folgt, gibt einen entsprechenden Logeintrag aus, dass keine Verbindung hergestellt werden kann. Schlussendlich ist der Button „Ping“ weiterhin vorhanden. Als Lasten kommen derzeit nur jene vom Hersteller EA in Frage, daher ist es hier nicht notwendig, einen Hersteller auszuwählen. Das Aardvark befindet sich aktuell im „Ruhemodus“ und das bqStudio kann erfolgreich über den I^2C-Bus kommunizieren, dies ist aus der Statusmeldung „Aardvark gestoppt > BQ Studio kann jetzt auslesen!“ ersichtlich und entspricht dem normalen Standard, da das Aardvark lediglich für kurze Momente aktiviert wird. Der Expressmodus ist derzeit deaktiviert und das Passworteingabefeld für den Administrationsbereich ist sichtbar, allerdings noch leer und damit ist der Tab „Admin“ noch nicht sichtbar.

4.1.2 Behandlungsart

Die Registerkarte „Behandlungsart“ entspricht der Hauptansicht des normalen Anwenders des ACC. In diesem Tab werden die benötigten Informationen gesammelt und konfiguriert. Dies beginnt bei der Behandlungsart und endet bei den aktuell verbundenen Geräten, dem aktuellen Stand der geprüften Akkus, der Zahl rechts neben der Akkuampel, dem Status, ob der geprüfte Akkupack alle Tests bestanden hat und der schnellen Möglichkeit, den Expressmodus zu aktivieren.

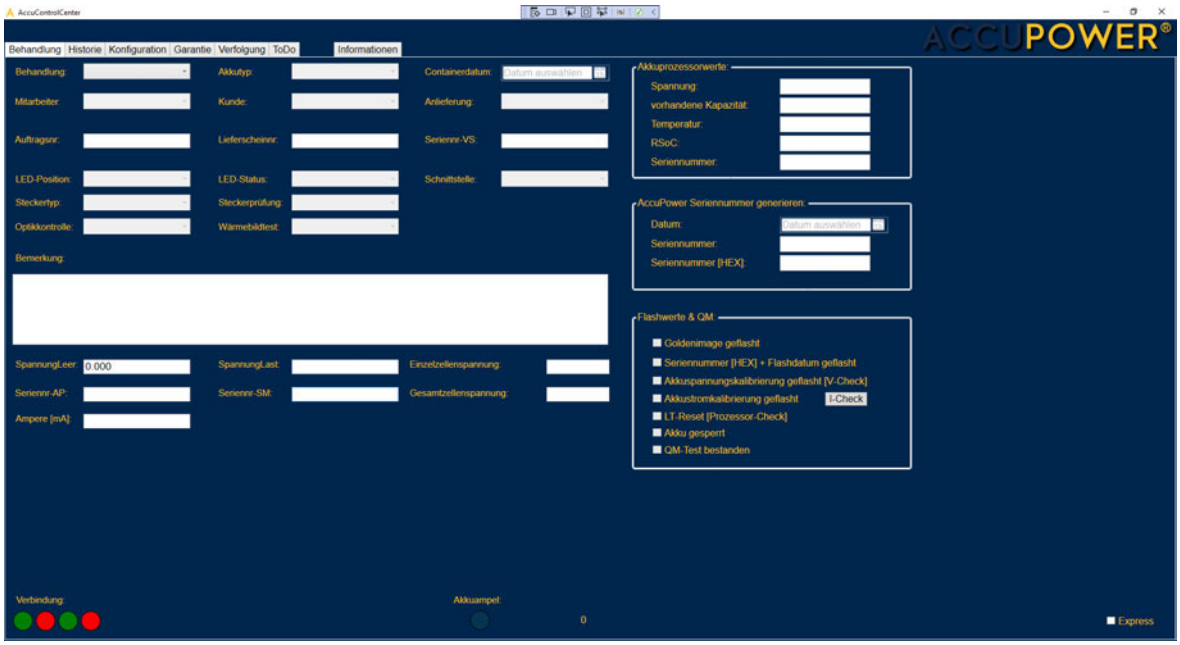


Abbildung 4.3: ACC: Registerkarte „Behandlung“

In der Abbildung 4.3 ist eine Übersicht über alle verfügbaren Felder des ACC gegeben. Entsprechend der Behandlungsart werden anschließend die dazu benötigten Felder angezeigt und überflüssige Felder ausgeblendet. Links unten ist es möglich, sich einen schnellen Überblick über die verbunden Geräte zu verschaffen. Von Links nach rechts sind folgende Verbindungen gegeben:

- **Voltmeter:** Aktive Verbindung zum Voltmeter vorhanden und Kommunikationskanal aufgebaut.
- **Amperemeter:** Keine aktive Verbindung zum Amperemeter vorhanden.
- **Last:** Aktive Verbindung zur Last vorhanden und Kommunikationskanal aufgebaut.
- **Aardvark:** Keine aktive Verbindung zum Aardvark vorhanden.

Die Akkuampel befindet sich in der dargestellten Abbildung im neutralen Status, je nach Testergebnis würde sich dieser Kreis rot oder grün färben, zu diesem visuellen Signal kommen noch Audiosignale, um entsprechend eine akustische Rückmeldung zu geben, ob der Test

bestanden wurde oder nicht. Der Zähler neben der Akkuampel gibt derzeit 0 abgefertigte Akkupacks an und würde sich entsprechend erhöhen, sobald ein Akkumulator abgefertigt wurde. Der Expressmodus ist aktuell deaktiviert, würde sich aber im Tab „Behandlungsart“ anlog zu jenem aus dem Tab „Konfiguration“ verhalten und den automatischen Reset der Komboboxfelder verhindern, wodurch nicht jedes Feld erneut ausgefüllt werden muss.

Das ACC verfügt über 8 Behandlungsarten diese sind: Eingangsprüfung, Nachladen, Versand, Prüfung, Aufbereitung n. prog, Aufbereitung prog, Defekt und InOutCheck.

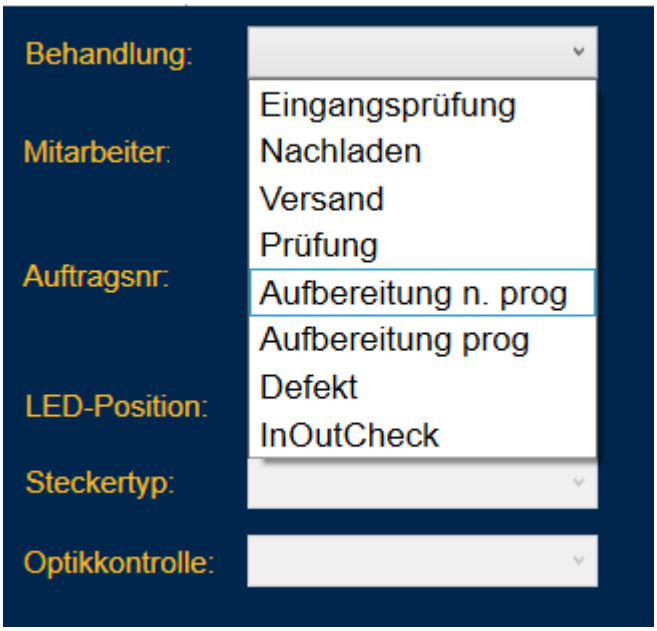


Abbildung 4.4: ACC: Behandlungsartenübersicht

Die in Abbildung 4.4 dargestellten Behandlungsarten unterscheiden sich in den benötigten Feldern und den dahinter stehenden Prozessen. Die „Eingangsprüfung“ entspricht jenem Prozess, der den Akkupack in Empfang nimmt und überprüft, ob dieser einwandfrei funktioniert oder ob Fehlfunktionen vorhanden sind. Der Prozess des „Nachladens“ entspricht einem Nachladevorgang jener Akkupacks, die einen Spannungsgrenzwert unterschritten haben oder die seit längerer Zeit im Lager liegen und Gefahr laufen, sich tief zu entladen. Der „Versand“ verbindet Informationen über die Lieferung des Akkupacks zu einem Kunden und das entsprechende Datum. Die Behandlungsart „Prüfung“ startet einen automatisierten Lasttest und erhebt entsprechende Daten über den Akkumulator. „Aufbereitung n. prog“ steht für die Behandlungsart Aufbereitung für nicht programmierbare Akkupacks und behandelt entsprechend Akkumulatoren ohne Battery Management System, kurz BMS. „Aufbereitung prog“ behandelt hingegen jene Akkupacks mit BMS und nach der Aufbereitung sind die Akkus für den Versandprozess freigegeben. In der Behandlungsart „Defekt“ werden jene Akkumulatoren behandelt, die irreparabel beschädigt sind und entsprechend der Entsorgung zugeführt werden. Abschließend ist zu erklären, dass der „InOutCheck“-Prozess einer Kombination aus Versand und Aufbereitung entspricht, um Expressaufträge zu verarbeiten.

Um ein besseres Verständnis für den unter A.1 Prozessleitbild Eingangsprüfung schematisch abgebildeten Eingangsprüfungsprozess zu bekommen, wird das Szenario dargestellt. AccuPower bezieht gefertigte Akkupacks aus Shenzhen, China und entsprechend wird in regelmäßigen Abständen ein Container angeliefert. Der Container beginnt seine Reise über den Seeweg und wird anschließend in Kroatien auf einen LKW verladen und in den Standort nach Graz geliefert. Beim Standort angekommen wird der Container leer geräumt und die Akkupacks werden im Lager für die Eingangsprüfung aufgereiht. Es gibt eine Expressmöglichkeit mittels Flugzeug und entsprechend dem Lieferweg sind unterschiedliche Kapazitätsgrenzen einzuhalten, ansonsten werden Strafen verhängt [34].

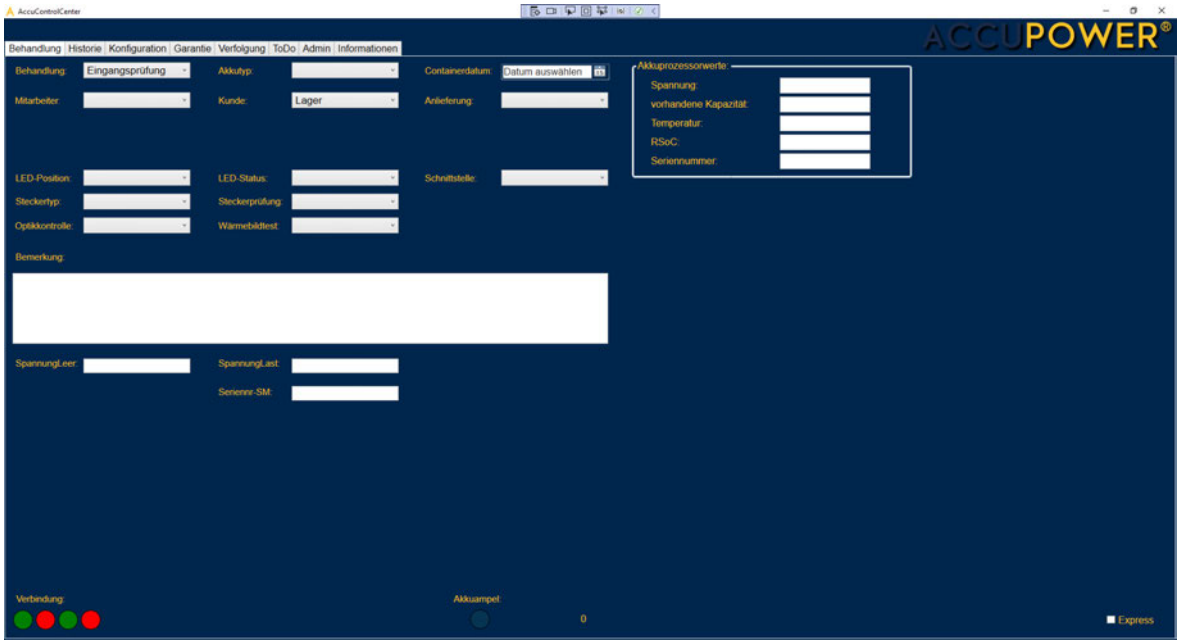


Abbildung 4.5: ACC: Behandlungsart „Eingangsprüfung“

Abbildung 4.5 gibt einen Überblick über alle benötigten Felder, welche ausgefüllt werden müssen, um die Behandlungsart „Eingangsprüfung“ verwenden zu können. In Tabelle Tabelle 4.1: Informationseingabemöglichkeiten Registerkarte Behandlung findet sich eine tabellarische Aufzählung aller in der Registerkarte „Behandlungsart“ zur Verfügung stehenden Eingabemöglichkeiten und deren Funktion. Die Eingangsprüfung besteht aus 3 Modulen, im ersten Modul wird die physische Begebenheit des Akkus überprüft, dazu gehören ein Wärmebildtest, eine Kontrolle der Stecker und weitere optische Überprüfungen und Funktionstests. Nach Abschluss dieses Moduls beginnt durch das Scannen der Seriennummer der Start des Belastungstests. Dabei wird der Akku der 8S2P-Serie 10 Sekunden mit 10 Ampere belastet und darf eine definierte Grenzspannung nicht unterschreiten, ansonsten ist der Test nicht bestanden worden. Das dritte Modul schließt automatisch an das Belastungstestmodul an und liest, wenn vorhanden, von der BMS des Akkumulators über den I^2C-Bus oder SMBus die Register Spannung, vorhandene Kapazität, Temperatur, relativer Ladezustand und die Seriennummer aus. Die Durchlaufzeit beträgt in etwa 3 Minuten pro Akku und im Anschluss wird der Akkupack entweder eingelagert oder der Nachbereitung eingereiht.

Der Prozess der „Aufbereitung prog“ unterscheidet sich vom Prozess der „Eingangsprüfung“ vor allem durch die Komplexität. Der Zweck des im Anhang A.2 Prozessleitbild Aufbereitung Programmierung dargestellten Prozessleitbildes liegt in der wertschöpfenden Tätigkeit der maßgeschneiderten Anpassung des Akkumulators an die Kundenbedürfnisse. Ein Akku, der kurzzeitig eine hohe Stromstärke, beispielsweise für den Start eines Motors, liefern muss, hat eine anders konfigurierte Firmware, welche auf die BMS geflasht wird, als ein Akku, welcher eine möglichst hohe Lebensdauer aufweisen muss, da er die Notbeleuchtung in einem Gebäude während eines Stromausfalles aufrechterhalten soll.

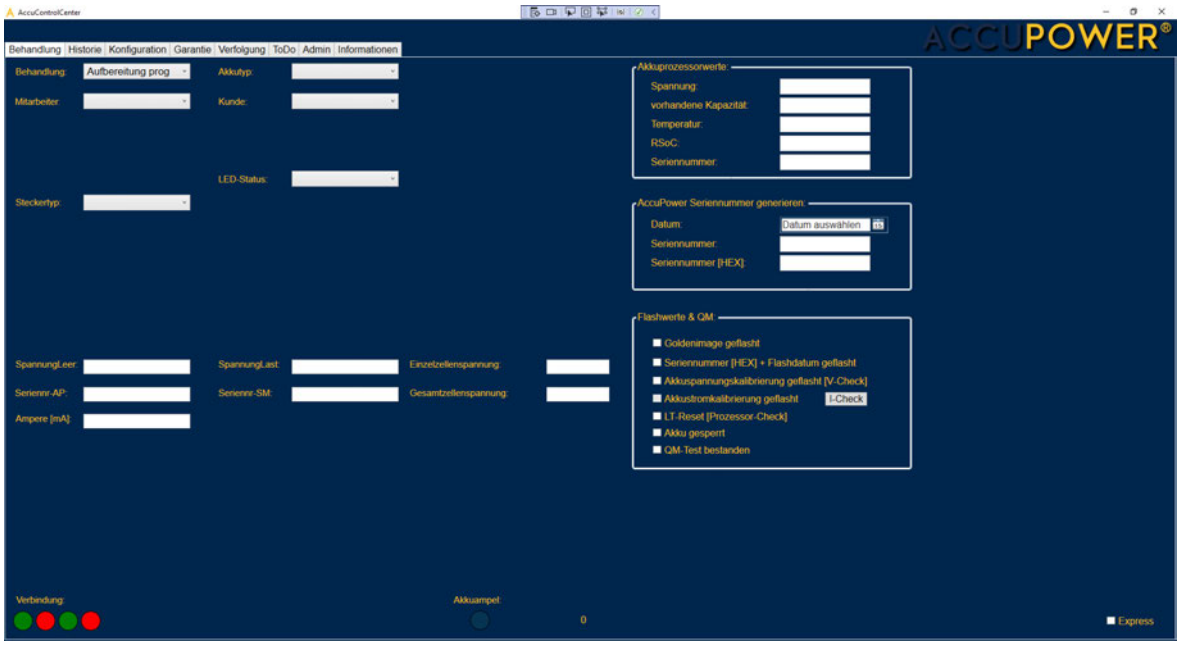


Abbildung 4.6: ACC: Behandlungsart „Aufbereitung prog“

Die in Abbildung 4.6 dargestellte Eingabemaske der Behandlungsart „Aufbereitung prog“ unterscheidet sich durch die wegfallenden Eingabemöglichkeiten: Containerdatum, Anlieferung, LED-Position, Schnittstelle, Steckerprüfung, Optikkontrolle, Wärmebildtest und durch die hinzugefügten Eingabemöglichkeiten: Einzelzellenspannung, Seriennr-AP, Gesamtzellenspannung, Ampere [mA], Datum, Seriennummer, Seriennummer [HEX], Goldenimage geflasht, Seriennummer [HEX] + Flashdatum geflasht, Akkuspannungskalibrierung geflasht [V-Check], Akkustromkalibrierung geflasht, Button I-Check, LT-Reset [Prozessor-Check], Akku gesperrt, QM bestanden von der in Abbildung 4.5 dargestellten Behandlungsart „Eingangsprüfung“. Der Ablauf des Prozesses besteht im Ausfüllen der Felder Behandlung, Akkutyp, Mitarbeiter, Kunde, LED-Status, Steckertyp und Datum, alle anderen Felder werden vom ACC automatisch mit Informationen befüllt oder sind Auslöser für Funktionen. Nach dem Ausfüllen der genannten Felder wird das Flashdatum (Datum) ausgefüllt und es wird für den Kunden und den Akkutyp die aktuelle Seriennummer erzeugt. Das Flashdatum hängt mit der Gewährleistung und Garantie zusammen, daher ist es für AccuPower notwendig, dieses Datum entsprechend anzupassen und nicht durch das gegenwärtige Datum eingeschränkt zu werden. Jeder Kunde hat für jeden Akkutypen einen eigenen Nummernkreis und somit einen genauen Überblick über die erstandenen Akkus, im Unterschied zu einer fortlaufenden Seriennummer für alle Kunden.

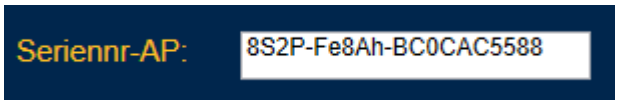


Abbildung 4.7: Beispiel einer AccuPower Seriennummer

Die in Abbildung 4.7 abgebildete Seriennummer von AccPower gibt einen Einblick in die Generierung dieser:

1. **Sie besteht aus 3 Segmenten**
   Aus dem Akkutyp, der Akkuchemie und der Kapazität sowie der verschlüsselten Seriennummer.

2. **Verschlüsseltes Segment**
   Das verschlüsselte Segment ist in codierter Form das Flashdatum und lässt einen Techniker bei einem Kundenbesuch relativ rasch das Alter des Akkupacks bestimmen.

3. **Seriennummer**
   Die Seriennummer setzt an das verschlüsselte Segment an und gibt eine grobe Auskunft über die Anzahl der im Einsatz befindlichen Akkupacks an.

Das zentrale Element der „Aufbereitung prog“ stellt der rechte untere Abschnitt der Abbildung 4.6 dar. In der Legende „Flashwerte & QM“ finden sich die Checkboxen zum Auslösen der erweiterten Funktionalitäten. Zum Verständnis dieses Abschnittes ist es notwendig zu verstehen, dass hier zwischen dem ACC und dem bqStudio gewechselt werden muss, um den Prozess erfolgreich abzuschließen. Die erste Checkbox „Goldenimage geflasht“ entspricht der Bestätigung, dass im bqStudio das Goldenimage, die Firmware, auf den Prozessor des Akkumulators programmiert wurde. Die anschließende Checkbox „Seriennummer [HEX] + Flashdatum geflasht“, gibt an, dass im bqStudio, nach dem Flashen des Goldenimages die korrekte Seriennummer und das korrekte Flashdatum aus dem ACC in das bqStudio zur Programmierung des Akkupacks übertragen und der Programmiervorgang abgeschlossen wurde. Die Checkbox „Akkuspannungskalibrierung geflasht [V-Check]“ entspricht der Übertragung der Einzellen- und Gesamtzellenspannungswerte aus dem ACC in das bqStudio, das Einprogrammieren dieser Werte und dem Vergleich mit den Messungen der kalibrierten Messgeräte. Es darf eine Differenz von +/- 10 mV nicht unter- bzw. überschritten werden zwischen den im Prozessor aktuell kalibrierten Spannungswert und dem aktuell gemessenen Spannungswert der eigens vom Hersteller kalibrierten Messgeräte. Sollte dies nicht der Fall sein wird eine Fehlermeldung ausgegeben und es ist nicht möglich, die Checkbox manuell zu setzen. An die Spannungskalibrierung schließt die Checkbox „Akkustromkalibrierung geflasht“ an, diese verhält sich analog zur Spannungskalibrierung, allerdings mit dem Unterschied der Belastung durch eine Last und der entsprechenden vergleichenden Messung des Stromes, des kalibrierten Akkupacks und des kalibrierten Amperemeters. Die „LT-Reset (Prozessor-Check)“-Checkbox überprüft die Gegebenheiten, ob das Flashdatum und die Seriennummer zwischen dem ACC und der BMS des Akkumulators harmonieren. Der letzte Schritt betreffend der Tests bildet die Checkbox „Akku gesperrt“, diese löst einen Test zur Überprüfung des Akkustatus aus. Der Akku muss im Produktivbetrieb gesperrt sein und kann

nur durch das Wissen um das Passwort entsperrt werden. Bei einem entsperrten Akku ist es möglich, Seriennummer, Flashdatum, Spannungsgrenzen und andere Werte zu ändern. Die Checkbox „QM-Test bestanden“ löst eine Überprüfung aus, ob alle Felder ausgefüllt und gesetzt sind, und gibt eine Warnung aus, sollte dies nicht der Fall. Diese Warnung kann vom Benutzer weggeklickt werden und dient lediglich der Kontrolle, ob ein Versehen vorliegt oder nicht. Bei korrekter Ausfüllung der Felder und gesetzten Checkboxen wird ein Datensatz in die Datenbank eingetragen und es wird eine neue Seriennummer geladen, dieses Verhalten ist auch beim Wegklicken der Warnung gegeben.

Abschließend ein Überblick über alle Informationseingabemöglichkeiten der Registerkarte „Behandlung“:

Tabelle 4.1: Informationseingabemöglichkeiten der Registerkarte „Behandlung“

Bezeichnung	Typ	Funktion
Behandlung	Kombobox	Auswahl der Behandlungsart und somit des Prozesses.
Akkutyp	Kombobox	Auswahl des entsprechenden Akkutyps von den 31 Möglichkeiten.
Containerdatum	Datum	Datum der Containerankunft bei Accu-Power.
Mitarbeiter	Kombobox	Auswahl des ACC Benutzers.
Kunde	Kombobox	Auswahl des Kunden.
Anlieferung	Kombobox	Auswahl der Anlieferungsart Schiff oder Flugzeug.
Auftragsnr	Textfeld	Eingabe der Auftragsnr.
Lieferscheinnr	Textfeld	Eingabe der Lieferscheinnr.
Seriennr-VS	Textfeld	Eingabe der Seriennummer für den Versand.
LED-Position	Kombobox	Eingabe des ausgelesenen Akkuladestandes Blitz, L, M, H, F, - .
LED-Status	Kombobox	Eingabe, ob die LED funktionstüchtig oder defekt ist.
Schnittstelle	Kombobox	Eingabe, ob Schnittstelle vorhanden, defekt oder nicht vorhanden.
Steckertyp	Kombobox	Auswahl aus 10 Steckermöglichkeiten.
Steckerprüfung	Kombobox	Eingabe, ob Steckerprüfung bestanden wurde.
Optikkontrolle	Kombobox	Eingabe, ob Optikkontrolle bestanden wurde.
Wärmebildtest	Kombobox	Eingabe, ob Wärmebildtest bestanden wurde.
Bemerkung	Textfeld	Eingabe zusätzlicher Bemerkungen zum entsprechenden Akkumulator.
SpannungLeer	Textfeld	Spannung vor der Belastung; Voltmeter.

Fortsetzung auf der nächsten Seite

Tabelle 4.1 – Fortsetzung

Bezeichnung	Typ	Funktion
SpannungLast	Textfeld	Spannung nach der Belastung; Voltmeter.
Einzelzellenspannung	Textfeld	Spannung einer einzelnen Zelle.
Seriennr-AP	Textfeld	Seriennummer von AccuPower.
Seriennr-SM	Textfeld	Seriennummer vom chinischen Produzenten.
Gesamtzellenspannung	Textfeld	Spannung des Akkumulators in mV; Voltmeter.
Ampere [mA]	Textfeld	Stromstärke des belasteten Akkupacks; Amperemeter.
Spannung	Textfeld	Spannung vom Akkumulator; µC.
vorhandene Kapazität	Textfeld	Kapazität des Akkumulators; µC.
Temperatur	Textfeld	Temperatur des Akkumulators; µC.
RSoC	Textfeld	Relative State of Charge des Akkumulator = Ladestand in %; µC.
Seriennummer	Textfeld	Seriennummer des Akkumulators; µC.
Datum	Datum	Flashdatum des Akkumulators.
Seriennummer	Textfeld	automatisch erzeugte Seriennummer des Akkumulators basierend auf Kunde und Akkutyp.
Seriennummer [HEX]	Textfeld	automatisch erzeugte Seriennummer im Hexadezimalsystem, zum Flashen für die BMS.
Goldenimage geflasht	Checkbox	Bestätigung, dass das Goldenimage geflasht wurde.
Seriennummer [HEX] + Flashdatum geflasht	Checkbox	Bestätigung, dass die Seriennummer in Hex und das Flashdatum geflasht wurden.
Akkuspannungskalibrierung geflasht [V-Check]	Checkbox	Bestätigung, dass die Spannungskalibrierung erfolgreich durchgeführt wurde - automatisierter Test.
Akkustromkalibrierung geflasht	Checkbox	Bestätigung, dass die Stromkalibrierung erfolgreich durchgeführt wurde - automatisierter Test.
LT-Reset [Prozessor-Check]	Checkbox	Bestätigung, dass der Akkumulator mit der korrekturen Seriennummer und dem korrekten Datum geflasht wurde - automatisierter Test.
Akku gesperrt	Checkbox	Bestätigung, dass der Akkumulator gesperrt wurde - automatisierter Test.

Fortsetzung auf der nächsten Seite

Tabelle 4.1 – Fortsetzung

Bezeichnung	Typ	Funktion
QM-Test bestanden	Checkbox	Letzte Bestätigung, dass alles korrekt durchgeführt wurde oder dass der Eintrag trotzdem erstellt werden soll.

4.1.3 Administrationsmenü

Das Administrationsmenü zwischen den Tabs „ToDo“ und „Information“ ist das einzige unter Sicherheitslevel 2 stehende Menü. Dies bedeutet, es ist erst verfügbar, wenn in der „Konfiguration“ das Administrationspasswort eingegeben wurde, ansonsten ist es unsichtbar. Dies kann in den Abbildung 4.3 ACC Registerkarte „Behandlung„ Übersicht und Abbildung 4.5: ACC Behandlungsart „Eingangsprüfung“ verglichen werden. Eine Abbildung zur Passworteingabe findet sich in Abbildung 4.2: ACC Registerkarte „Konfiguration“. Nach der korrekten Eingabe des Passwortes wird dies mit einem Enter bestätigt und bei korrekter Eingabe wird das Adminmenü sichtbar.

```
private void cbAdmin Click(object sender, RoutedEventArgs e)
{
    if (txtAdminkey.Visibility == Visibility.Hidden)
    {
        txtAdminkey.Visibility = Visibility.Visible;
    }
    else
    {
        txtAdminkey.Visibility = Visibility.Hidden;
        tabAdmin.Visibility = Visibility.Hidden;
        txtAdminkey.Text = "";
    }
}

private void txtAdminkey KeyUp(object sender, KeyEventArgs e)
{
    if (e.Key != System.Windows.Input.Key.Enter) return;

    e.Handled = true;

    if(txtAdminkey.Text == "passwort")
    {
        tabAdmin.Visibility = Visibility.Visible;
    }else
    {
        tabAdmin.Visibility = Visibility.Hidden;
    }
}
```

Quelltext 4.2: C#-Code betreffend die Sichtbarkeit des Administrationsmenüs

Der Quelltext 4.2 stellt die betreffenden Codezeilen dar. Zuerst wird die Methode „cbAdmin_-Click()“ aufgerufen. Dies geschieht, sobald im Tab „Konfiguration“ die Checkbox cbAdmin geklickt wird. Diese Methode überprüft, ob das Textfeld zur Passworteingabe sichtbar oder unsichtbar ist (txtAdminkey). Sollte es unsichtbar sein, wird es sichtbar gemacht und falls es

bereits sichtbar ist, wird es unsichtbar gemacht und das eingegebene Passwort wird geleert, zusätzlich wird der Tab „Admin“ wieder versteckt. Anschließend wird bei der Passworteingabe die Methode „txtAdminkey_KeyUp()“ aufgerufen. Diese entspricht einem Event-Handler für das KeyUp-Ereignis und wird ausgelöst, sobald auf der Tastatur eine Taste losgelassen wird. Im „e“ Parameter wird die losgelassene Taste übergeben und in der Methode wird überprüft, ob es sich um die Enter-Taste handelt. Falls sie es nicht ist, wird die Methode sofort verlassen. Sollte allerdings die Enter-Taste losgelassen worden sein, wird der eingegebene Text im Textfeld „txtAdminkey“ auf Gültigkeit überprüft. Bei der korrekten Eingabe des Passwortes wird die Registerkarte „Admin“ angezeigt und sollte das Passwort ungültig sein, wird die genannte Registerkarte weiterhin im unsichtbaren Status gehalten. Das Passwort wird direkt als Text, ohne Hashes und Datenbankabfrage im ACC geprüft. Dies liegt daran, dass das ACC nur mit einer Datenbank-Verbindung gestartet werden kann und da bis jetzt keine komplexeren Anforderungen an ein Benutzerverwaltungssystem gegeben sind.

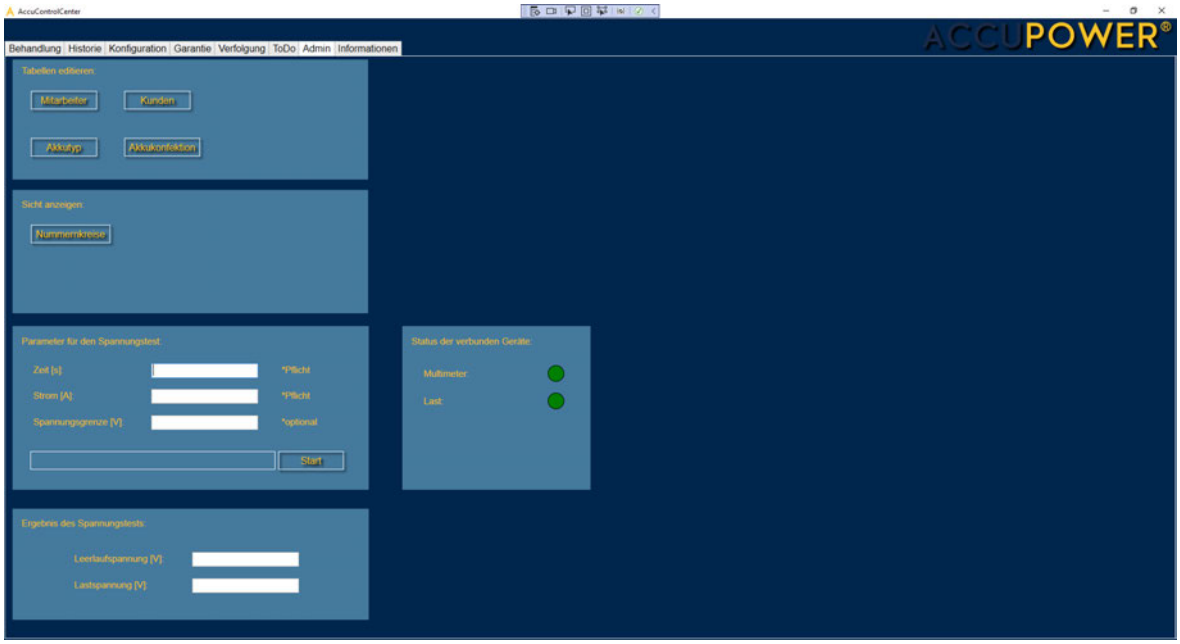


Abbildung 4.8: ACC: Registerkarte „Admin“

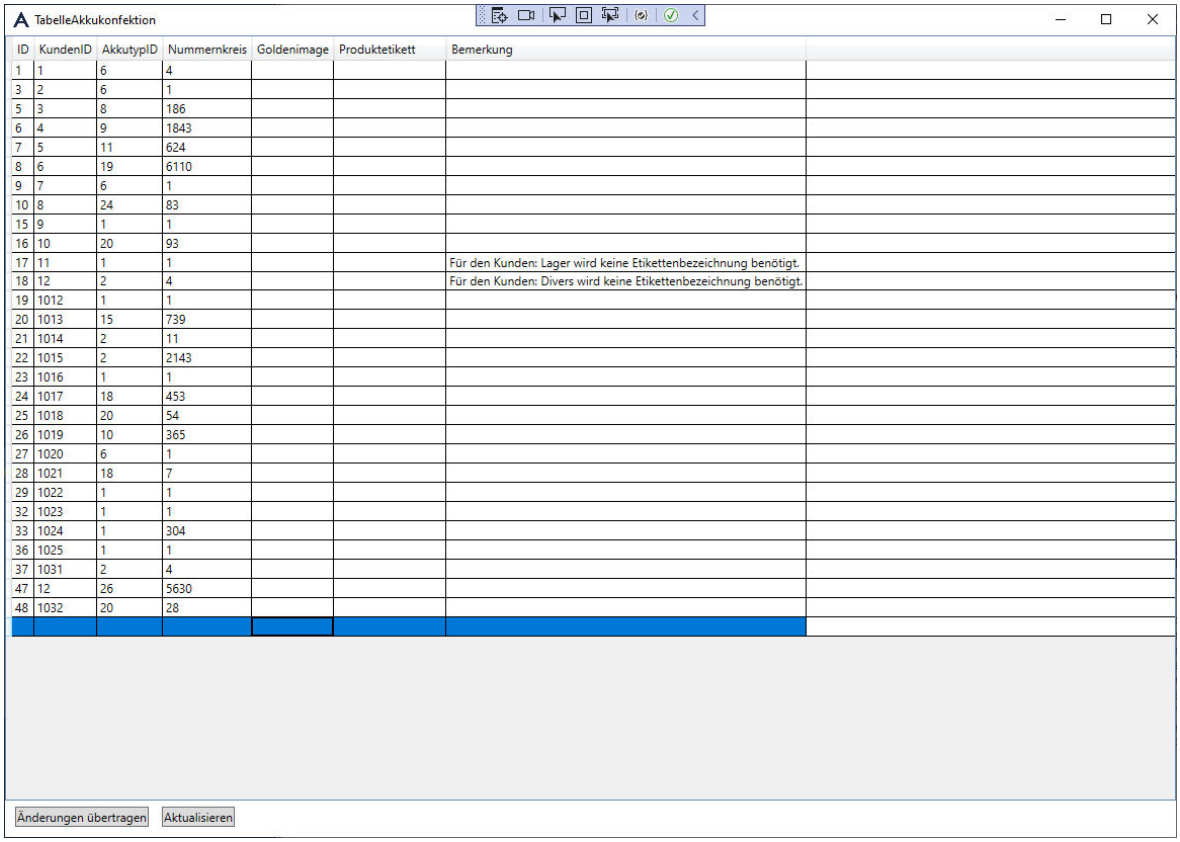
In Abbildung 4.8 wird der Inhalt der Registerkarte „Admin“ abgebildet. Es ist möglich, mit den 4 Tabellen: Mitarbeiter, Kunden, Akkutyp und Akkukonfektion in Interaktion zu treten (CRUD = Create, Read, Update, Delete). Die Anzeige der Sicht Nummernkreise ist verfügbar, diese verknüpft die Informationen aus den genannten Tabellen. Abschließend bietet das Adminmenü die Möglichkeit, einen manuellen Lasttest zu starten, mit der Parametrierung betreffend der Zeit, der Stromstärke und einer Spannungsgrenze, bei deren Erreichung der Test gestoppt wird. Die Verbindung mit Multimeter und Last wird über das bekannte Ampelsystem rechts angezeigt und die Ergebnisse des Belastungstests werden unterhalb angezeigt.

Im Quelltext 4.3 wird der Aufruf des neuen Fensters mit der Tabelle „Akkukonfektion“ abgebildet. Die Auslösung dieser Methode wird durch einen Klick auf den Button „Akkukonfektion“ ausgelöst. Es wird eine neue Instanz der Klasse „TabelleAkkukonfektion“ erstellt und diese öffnet das genannte Fenster mit dem Inhalt der Tabelle „Akkukonfektion“. Dazu wird zunächst das neue Objekt „tb“ vom Typ „TabelleAkkukonfektion“ instanziiert, durch Aufruf des

```
private void cmbAdminAkkukonfektion Click(object sender, RoutedEventArgs e)
{
    TabelleAkkukonfektion tb = new TabelleAkkukonfektion();
    tb.Show();
}
```

Quelltext 4.3: C#-Code für den Aufruf der Tabelle „Akkukonfektion“

Standardkonstruktors der Klasse „TabelleAkkukonfektion“. Abschließend wird die „Show“-Methode des Objektes „tb“ aufgerufen und das entsprechende Fenster mit der Tabelle wird geöffnet.



ID	KundenID	AkkutypID	Nummernkreis	Goldenimage	Produktetikett	Bemerkung
1	1	6	4			
3	2	6	1			
5	3	8	186			
6	4	9	1843			
7	5	11	624			
8	6	19	6110			
9	7	6	1			
10	8	24	83			
15	9	1	1			
16	10	20	93			
17	11	1	1			Für den Kunden: Lager wird keine Etikettenbezeichnung benötigt.
18	12	2	4			Für den Kunden: Divers wird keine Etikettenbezeichnung benötigt.
19	1012	1	1			
20	1013	15	739			
21	1014	2	11			
22	1015	2	2143			
23	1016	1	1			
24	1017	18	453			
25	1018	20	54			
26	1019	10	365			
27	1020	6	1			
28	1021	18	7			
29	1022	1	1			
32	1023	1	1			
33	1024	1	304			
36	1025	1	1			
37	1031	2	4			
47	12	26	5630			
48	1032	20	28			

Abbildung 4.9: ACC: Administrationsmenü Tabelle „Akkukonfektion“

Abbildung 4.9 stellt den Inhalt des geöffneten Fensters durch den Klick auf den Button „Akkukonfektion“ dar. Innerhalb der von der Datenbank geladenen Tabelle befinden sich die Spalten ID, KundenID, AkkutypID, Nummernkreis, Goldenimage, Produktetikett und Bemerkung. Einige Daten wurden aus Datenschutzrechtlichengründen entfernt. Es ist erkennbar, dass diese Tabelle für den Menschen nicht unmittelbar leserlich ist, daher kann der Anwender die Sicht „Nummernkreise“ parallel öffnen und dadurch den Inhalt zielgerichtet bearbeiten. Die Spalten Goldenimage und Produktetikett entsprechen zukünftigen Verweisen auf *.srec Dateien im Falle des Goldenimages und *.prn Dateien im Falle des Bartender Etikettentemplates. Der Anwender kann durch einen Mausklick auf die entsprechende Zeile diese markieren und anschließend mittels Drücken der Löschen-Taste löschen. Ein Bearbeiten der Daten ist durch einen Doppelklick auf den Datensatz möglich, anschließend kann der neue Inhalt in das Feld übertragen werden. In jedem Fall muss abschließend der Button „Änderung übertragen“ angeklickt werden, dieser überträgt die durchgeführten Änderungen auf den Datenbankserver.

Solange dieser Button nicht angeklickt wurde, sind die Änderungen noch nicht übertragen. Der Button „Aktualisieren“ lädt den aktuellen Datenbestand vom Datenbankserver und stellt diesen in der Tabelle dar. Er kann zur Bestätigung der Übertragung der überarbeiteten Daten verwendet werden, indem er den aktuellen geänderten Datenbestand lädt.

Die Programmiersprache C# bietet die komfortable Möglichkeit, ein „SqlCommandBuilder“-Objekt zu erstellen und dadurch die genannte CRUD-Funktionalität zu unterstützen. Zur Verwendung des genannten Objektes wird die Microsoft ADO.NET Bibliothek verwendet und dies entspricht einem der Vorteile, wenn möglichst auf einen Hersteller im Technologiestack gesetzt wird [35].

```
private void cmbAktualisieren Click(object sender, RoutedEventArgs e)
{
    string sqlAkkukonfektion = "Select * FROM TECHNIK.dbo.akkukonfektion";
    dataadapter = new SqlDataAdapter(sqlAkkukonfektion, cnn);

    var builder = new SqlCommandBuilder(dataadapter);
}

private void cmbUpdate Click(object sender, RoutedEventArgs e)
{
    try
    {
        dataadapter.Update(dt);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex);
    }
}
```

Quelltext 4.4: C#-Code zur Erstellung eines SqlCommandBuilder-Objektes

Im Quelltext 4.4 wird die Erstellung „SqlCommandBuilder“-Objekt dargestellt. Die Methoden „cmbAktualisieren_Click()“ und „cmbUpdate_Click()“ entsprechen den Clicks auf die analog benannten Buttons aus Abbildung 4.9. Zuerst wird in der Methode „cmbAktualisieren_Click()“ die SQL-Abfrage als String-Variable „sqlAkkukonfektion“ erstellt. Diese wird im anschließend neu erstellten „SqlDataAdpater“-Objekt als SQL-Abfrage definiert und mit der Variable „cnn“ wird die Verbindung zur Datenbank aufgebaut. Das Herzstück ist der abschließend erzeugte „SqlCommandBuilder“, welcher mit dem „SqlDataAdapter“ initiiert wird. Durch das erstellte „builder“-Objekt werden die SQL-Befehle INSERT, UPDATE, DELETE basierend auf der SELECT-Anweisung der SQL-Abfrage einfach und effizient ermöglicht. In der Methode „cmbUpdate_Click“ ist dies durch den einfachen Abruf der „.Update“-Methode des „dataadapter“-Objektes ersichtlich. Die try-catch-Methodik fängt einen etwaigen Fehler in der Methode „cmbUpdate_Click()“ im „catch“-Block mittels einer Fehlermeldung in einer MessageBox mit dem entsprechenden Error, stehend in der „ex“ Variable ab.

Der Quelltext ist in der Methode „cmbAktualisieren_Click()“ nicht komplett vollständig abgebildet. Es fehlt das Öffnen und Schließen der Datenbankverbindung, darauf wird im Kapitel 4.2.1 MS SQL Server Anbindung genauer eingegangen.

4.2 Schnittstellen

In Kapitel 3.3 Systemarchitektur findet sich bereits ein Überblick über die Architektur des ACC und der benötigten physischen Geräte und ihrer Schnittstellen. In diesem Kapitel wird detailliert auf die technische Implementierung der APIs in das ACC eingegangen und anhand von Abbildungen, Tabellen und Quelltext untermauert. An dieser Stelle folgt ein kurzer Überblick über die nachfolgenden Unterkapitel:

1. **MS SQL Server Anbindung**
   Diese entspricht dem Gedächtnis des ACC. Sämtliche Daten werden nicht innerhalb des in C# programmierten Programmes gespeichert, sondern in der MS SQL-Server-Datenbank. Diese liegt auf einem Datenbankserver innerhalb des AccuPower OT-Netzwerkes und die MS SQL Server Anbindung entspricht den Synapsen im Gehirn, um Informationen an die entsprechenden Stellen, in diesem Fall an das Gedächtnis, den Speicher weiterzuleiten.

2. **Einbinden der Ampere-, Voltmeter und der Last**
   Mit dem Ampere-, Voltmeter und der Last wird über das Ethernetprotokoll kommuniziert. Die Befehlssätze zur Steuerung dieser Geräte entsprechen dem SCPI-Standard und dadurch ist es möglich, sowohl Messergebnisse zu erhalten als auch einen Belastungstest zu starten.

3. **Aardvark**
   Das Aardvark entspricht der Kommunikationseinheit, die am nähesten an den Akkumulatoren liegt. Diese kann über den I^2C-Bus und den SMBus kommunizieren und ist entsprechend für das ACC die einzige Möglichkeit, mit den Akkupacks zu kommunizieren.

4.2.1 MS SQL Server Anbindung

Die MS SQL Server Anbindung im ACC erfolgt über einen einfachen „connectionString“. In diesem sind die IP-Adresse des Datenbankservers, der Datenbankname, der Benutzername, das Passwort des Benutzers, die Aktivierung der Verschlüsselung und die Vertrauenseinstellungen bezüglich der Sicherheitszertifikate enthalten. Zusätzlich wird ein Objekt vom Typ „SqlConnection“ angelegt, um dann in weiterer Folge eine Instanz der Klasse „SqlConnection“ zu bilden. Anschließend wird über die erstellte Instanz der Klasse „SqlConnection“ über den „connectionString“ eine Verbindung zur Datenbank aufgebaut. Dieser geöffnete Kommunikationskanal wird verwendet, um Informationen von der Datenbank und der entsprechenden Tabelle auszulesen oder um Daten in die Tabelle zu schreiben oder um Daten aus der Tabelle zu löschen. Sobald die Transaktion erfolgreich abgeschlossen wird, wird der Kommunikationskanal geschlossen und somit die Verbindung zum Datenbankserver getrennt. Diese theoretische Einleitung wird im nachfolgenden Quelltext praktisch erörtert, um ein besseres Verständnis zu ermöglichen.

```
//Globale Variablen
string connectionString = @"Data Source = 127.0.0.1; Initial Catalog = Datenbank; User ID
    = admin; Password = passwort; Encrypt = True; TrustServerCertificate = True";
SqlConnection cnn;

       private void cbKunden SelectionChanged(object sender, SelectionChangedEventArgs e)
       {
           cnn = new SqlConnection(connectionString);
           SqlCommand cmdKunde = new SqlCommand("Select * FROM TECHNIK.dbo.kunden WHERE
               Bezeichnung = '" + cbKunden.SelectedItem + "'", cnn);

           try
           {
               cnn.Open();

               using (SqlDataReader read = cmdKunde.ExecuteReader())
               {
                   while (read.Read())
                   {
                       kundenID = (Convert.ToInt16(read["ID"]));
                   }
               }
           }

           finally
           {
               cnn.Close();
           }
       }
```

Quelltext 4.5: C#-Code für den Aufbau einer SQL-Verbindung

Die globale Variable „connectionString“ aus dem Quelltext 4.5 entspricht der Zeichenkette zur Verbindung des SQL-Servers, die einzelnen Elemente, die durch ein Semikolon getrennt sind, werden auch „Schlüssel-Wert-Paare“ genannt, im Detail bedeutet diese:

- **Data Source = 127.0.0.1**
  Die „Data Source“ oder auch Datenquelle gibt die IP-Adresse des SQL-Datenbankservers an. In diesem Fall ist mit „127.0.0.1“ der lokale Computer auch localhost genannt.

- **Initial Catalog = Datenbank**
  Der „Initial Catalog“ entspricht dem Namen der Datenbank, auf welche zugegriffen werden soll. In diesem Beispiel heißt die Datenbank „Datenbank“.

- **User ID = admin**
  Die „User ID“ entspricht dem Benutzer, der zur Authentifizierung bei der Datenbank verwendet werden soll. In diesem Fall heißt der Benutzer „admin“.

- **Password = passwort**
  Das „Password“ entspricht dem Passwort des Benutzers, der „User ID“, der zur Authentifizierung auf dem SQL-Datenbankserver verwendet werden soll.

- **Encrypt = True**
  Das True setzen des „Encrypt“ Schlüssel-Wert-Paares gibt an das die Kommunikation zwischen Client und Server mit der SSL/TLS-Verschlüsselung geschützt werden soll.

- **TrustServerCertificate = True**
  Das „TrustServerCertificate“ Schlüssel-Wert-Paar gibt das Verhalten bzgl. Server-Zertifikaten an, welche nicht von einer vertrauenswürdigen Zertifizierungsstelle signiert worden sind. Das Zertifikat, welches von AccuPower verwendet wird, ist nicht von einer Zertifizierungsstelle ausgestellt und daher muss dieser Wert auf True gesetzt werden, ansonsten würde die Kommunikation verweigert.

Die Methode „cbKunden_SelectionChanged()“ wird aufgerufen, wenn in der Kombobox „cbKunden“, beispielsweise in Abbildung 4.6: ACC Behandlungsart „Aufbereitung prog“ ersichtlich, der ausgewählte Wert verändert wird. Sobald der genannte Fall eintritt, wird versucht eine neue Datenbankverbindung aufzubauen und durch das „SqlCommand“-Objekt „cmdKunde“, nach dem Öffnen der Datenbankverbindung die ID des neu ausgewählten Kunden zu laden. Diese ID wird anschließend in die Variable „kundenID“ gespeichert und nach Abschluss der Transaktion wird die Verbindung durch die Methode „.Close()“ geschlossen.

Das Öffnen und anschließende Schließen der Verbindung zum SQL-Server ist notwendig, um einerseits die lokalen Ressourcen und andererseits die Serverressourcen freizugeben. Bei unsachgemäßer Programmierung können Hunderte oder Tausende von Kommunikationskanälen geöffnet sein, daher ist es empfehlenswert, vor der eigentlichen Programmierung des Inhalts der Transaktion bereits das Schließen in die Methode einzubauen.

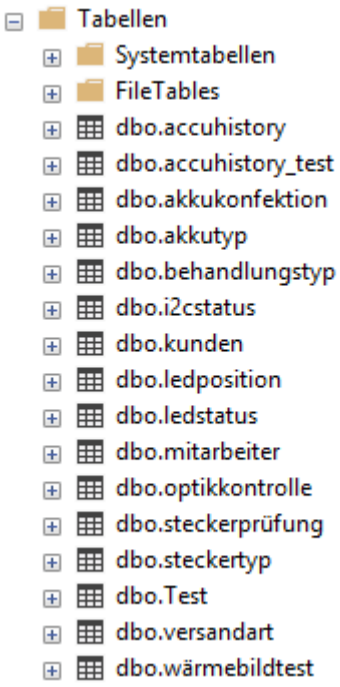


Abbildung 4.10: ACC-Datenbank: Tabellenübersicht

In Abbildung 4.10 werden alle 16 Tabellen in der Datenbank des ACC angezeigt, davon sind 14 Tabellen für den Produktivbetrieb notwendig. Die abgebildeten Tabellen haben folgende Nutzen:

- **accuhistory**
  Ist die Haupttabelle des ACC. Hier laufen alle Informationen zusammen und es ist möglich, die gesamte Produktion zu überwachen und nachzuvollziehen, was gerade passiert und was jemals passiert ist.

- **accuhistory_test**
  Entspricht einer Testtabelle, um gewisse Funktionalitäten des ACC in der Testphase zu testen.

- **akkukonfektion**
  In dieser Tabelle werden Daten zum Akkumulator spezifisch der Kundenwunschproduktion gespeichert.

- **akkutyp**
  Hier sind die Informationen der normalen Akkumulatorserien enthalten.

- **behandlungstyp**
  Detailinformationen zu den Behandlungstypen sind in dieser Tabelle enthalten und ganz allgemein die Behandlungsarten.

- **i2cstatus**
  Alle Statusmöglichkeiten für den I^2C-Bus sind in dieser Tabelle enthalten.

- **kunden**
  Alle Kunden und zusätzliche Informationen zu diesen, werden in dieser Tabelle gespeichert.

- **ledposition**
  Alle möglichen LEDPositionen und weitere Vermerke zu diesen sind in dieser Tabelle enthalten.

- **ledstatus**
  Alle Statusmöglichkeiten für die LED sind in dieser Tabelle enthalten (gewisse Akkupacks haben eine physische Ladezustandsanzeige an der Außenhaut angebracht).

- **mitarbeiter**
  Alle Mitarbeiter und zusätzliche Informationen zu diesen werden in dieser Tabelle gespeichert.

- **optikkontrolle**
  Alle Möglichkeiten der Optikkontrolle und weitere Informationen betreffend dieser werden in dieser Tabelle gespeichert.

- **steckerprüfung**
  Alle Möglichkeiten zur Steckerprüfung und deren Spezifikationen werden in dieser Tabelle gespeichert.

- **Test**
  Entspricht einer Testtabelle und ist nicht für den Produktivbetrieb relevant.

- **versandart**
  Alle möglichen Versandarten und zusätzliche Informationen zu diesen sind in dieser Tabelle enthalten.

- **wärmebildtest**
  Alle möglichen Ergebnisse des Wärmebildtests und weitere Vermerke zu diesen sind in dieser Tabelle enthalten.

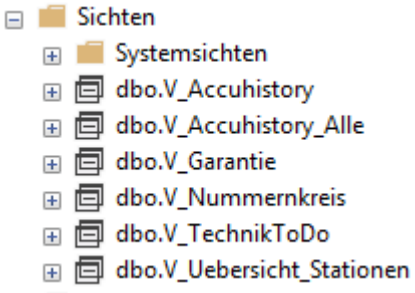


Abbildung 4.11: ACC-Datenbank: Sichtenübersicht

In Abbildung 4.11 sind alle notwendigen 6 Sichten für den Produktivbetrieb des ACC dargestellt. Sichten haben den Vorteil, die Rechenleistung des Datenbankservers zu benutzen und verlangen daher vom Client keine zusätzliche Leistung. Die dargestellten Sichten haben folgende Funktion:

- **V_Accuhistory**
  In dieser Sicht werden die letzten 250 Produktionsschritte der Montage abgebildet.

- **V_Accuhistory_Alle**
  In dieser Sicht werden alle Produktionsschritte der Montage abgebildet.

- **V_Garantie**
  In dieser Sicht ist es für den technischen Support rasch möglich, anhand der Seriennummer nachzuprüfen, ob Garantieansprüche bzw. Gewährleistungsansprüche auf den angefragten Akkupack bestehen.

- **V_Nummernkreis**
  In dieser Sicht wird für den Schichtleiter leserlich die Möglichkeit geboten nachzuvollziehen, welcher Kunde bei welcher Akkuserie gerade welchen Nummernkreis im Einsatz hat (Adminmenü).

- **V_TechnikToDo**
  In dieser Sicht ist es für die Techniker möglich, alle Akkumulatoren einer Liste abzufragen, bei denen notwendige Schritte bestehen, um diese aus einem möglichen Gefahrenbereich zu entfernen. Dies bezieht sich auf Akkumulatoren bei denen die Grenzspannung bei Prüfungen unter- oder überschritten wurden, oder auf Akkumulatoren, welche seit mehreren Monaten im Lager eingelagert sind und bei denen die Gefahr der Tiefentladung besteht (meist 6 Monate).

- **V_Uebersicht_Stationen**
  In dieser Sicht kann mittels der Seriennummer rasch abgefragt werden, welche Behandlungsarten der Akkupack im Unternehmen AccuPower durchlaufen hat und wer welche Behandlungen wann vollzogen hat.

Alle genannten Tabellen, die für den Produktivbetrieb notwendig sind, sind mindestens mit der Funktionalität des Datenlesens im ACC enthalten, gewisse Tabellen wie „accuhistory“, „akkutyp“ und „akkukonfektion“ werden im ACC mit der kompletten CRUD-Funktionalität unterstützt. Die Sichten sind alle im ACC mit Lesefunktionalität enthalten.

4.2.2 Einbinden der Multimeter und der Last

Um die Multimeter Keysight 34465 A und das GWInstek GDM-9061 sowie die Last EA-EL 3080-60 B steuern und Daten von ihnen auslesen zu können, müssen die Geräte im OT-Netzwerk der Firma AccuPower eingebunden sein. Im Idealfall werden diesen vom DHCP-Server „Dynamic Host Configuration Protocol“, statische IP-Adressen zugewiesen und auf den physischen Geräten wird ein Etikett mit dieser reservierten IP-Adresse angebracht, um den Anwendern des ACC eine erleichterte Konfiguration zu bieten. Der DHCP-Server verwaltet alle im Netzwerk befindlichen Geräte und weist diesen basierend auf der Konfiguration des Servers dynamische oder statische IP-Adressen zu, anhand dieser Adresse wird anschließend mit dem Gerät kommuniziert [36].

Die Verbindung bei allen aufgezählten Geräten findet über das Ethernetprotokoll statt, dabei spielt es keine Rolle, ob diese eine Kabelverbindung oder über WLAN aufgebaut wird. Im gegenwärtigen Falle des ACC findet die Kommunikation meist über die Kabelverbindung statt. Voraussetzung für einen erfolgreichen Kommunikationsaufbau ist ein aktivierter und erreichbarer Telnetserver auf den Geräten und eine deaktivierte Authentifizierung. Das ACC könnte eine Authentifizierung durchführen, allerdings wird dies aufgrund des niederschwelligen Bedienungswunsches nicht benötigt. Ein Telnetserver dient der entfernten Steuerung eines Gerätes (=Remote-Zugriff) und ermöglicht Arbeiten, ohne der eigenen physischen Anwesenheit im Raum des Gerätes durchzuführen [37].

Sobald die genannten Voraussetzungen erfüllt sind, eine erfolgreiche Verbindung in das Firmennetzwerk von AccuPower seitens des ACC und seitens der entsprechenden Geräte mit einer gültigen IP-Adresse und bei erfolgreicher Kommunikation mit dem Telnetserver des entsprechenden Gerätes, ist eine erfolgreiche Umsetzung des notwendigen Prozesses möglich. Detaillierte Konfigurations- und Einrichtungsmöglichkeiten sind in der jeweiligen Bedienungsanleitung des Herstellers enthalten [38] [39] [40].

Bevor ein Prozess innerhalb des ACC abgearbeitet wird, wird in der Registerkarte „Konfiguration“ getestet, ob eine erfolgreiche Verbindung mit dem Gerät aufgebaut werden kann. In Abbildung 4.2: ACC Registerkarte „Konfiguration“ findet sich die entsprechende Darstellung der GUI.

```
        public bool PingVoltmeter()
        {
            var client = new System.Net.Sockets.TcpClient();

            if (rbGWInstek.IsChecked == true)
            {
                if (client.ConnectAsync(txtIp.Text, 3001).Wait(1000) && connect == false)
                {

                    elConnectionVoltmeter.Fill = new SolidColorBrush(Colors.Green);
                    elConnectionVoltmeterConfig.Fill = new SolidColorBrush(Colors.Green);
                    elAdminMultimeter.Fill = new SolidColorBrush(Colors.Green);

                    multimeter = true;
                    client.Close();

                    SocketConnectionVoltmeter();

                    connect = true;
                    cmbCheckConnectionDevice.Content = "Trennen";

                    return true;
                }
                else
                {
                    elConnectionVoltmeter.Fill = new SolidColorBrush(Colors.Red);
                    elConnectionVoltmeterConfig.Fill = new SolidColorBrush(Colors.Red);
                    elAdminMultimeter.Fill = new SolidColorBrush(Colors.Red);

                    multimeter = false;
                    lblConsoleVoltmeter.Content = "Kein Ping möglich zu " + txtIp.Text.
                        ToString() + " 3001" + Environment.NewLine + "oder Verbindung
                        absichtlich getrennt!";
                    client.Close();

                    connect = false;
                    cmdCheckConnectionDevice.Content = "Ping";
                    txtSpannung.Text = "";

                    return false;
                }
            }
        }
```

Quelltext 4.6: C#-Code für den Aufbau einer Telnet-Verbindung

In Quelltext 4.6 wird die Funktionalität hinter der Aktivität des Klicks auf den Button „Ping“ auf Voltmeterhöhe dargestellt, im Genaueren für das GWInstek GDM-9061. Der Aufruf der Methode „PingVoltmeter“ initiiert eine neue Instanz der Klasse „TcpClient“. Anschließend wird überprüft, um welchen Hersteller es sich handelt und mittels einer asynchronen Anfrage, welche 1000 ms dauert, getestet, ob das Gerät erreichbar ist. Sollte dies der Fall sein, werden die entsprechenden Ellipsen oder Ampeln in allen Registerkarten auf grün geschalten und der asynchrone Client wird mittels der „.Close“ geschlossen. Bevor der Block der „if“-Abfrage verlassen wird, wird noch mittels der „SocketConnectionVoltmeter()“-Methode der aktuelle

Spannungswert ausgelesen, die Variable „connect“ wird auf True gesetzt und der „Ping“-Button wird zum „Trennen“-Button umkonfiguriert. Sollte keine Verbindung möglich sein, werden die Ampeln in allen Registerkarten auf Rot geschalten, die entsprechenden Variablen werden auf False gesetzt und eine Fehlermeldung wird ausgegeben.

Das Auslesen entsprechender Spannungs- oder Stromwerte erfolgt im weiteren Schritt per ASCII-Zeichensatz, sowohl die Anfrage als auch die Rückgabe. Dabei müssen die gesendeten Befehle in den ASCII-Bytecode übersetzt und die empfangenen Werte in höherwertige Datentypen übersetzt werden. Der ASCII-Code „American Standard Code for Information Interchange“ entspricht einer Zuordnung von Zeichen in einen 7-bit-Code [41].

```
    private void SocketConnectionAmperemeter()
    {
                IPHostEntry ipHostInfo = Dns.GetHostEntry(Dns.GetHostName());
                IPAddress ipAddress = System.Net.IPAddress.Parse(txtIpAmperemeter.Text);
                IPEndPoint remoteEP = new IPEndPoint(ipAddress, 5025);

                Socket sender = new Socket(ipAddress.AddressFamily, SocketType.Stream,
                    ProtocolType.Tcp);

                try
                {
                    sender.Connect(remoteEP);

                    lblConsoleAmperemeter.Content = ("Verbunden zu " + sender.
                        RemoteEndPoint.ToString());
                    elConnectionAmperemeter.Fill = new SolidColorBrush(Colors.Green);
                    elConnectionConfigAmperemeter.Fill = new SolidColorBrush(Colors.
                        Green);

                    byte[] msg = Encoding.ASCII.GetBytes("*CLS\n");
                    int bytesSent = sender.Send(msg);

                    msg = Encoding.ASCII.GetBytes("*RST\n");
                    bytesSent = sender.Send(msg);

                    msg = Encoding.ASCII.GetBytes("MEAS:CURR:DC? 10\n");
                    bytesSent = sender.Send(msg);

                    int bytesRec = sender.Receive(bytesampere);
                    txtAmpere.Text = Encoding.ASCII.GetString(bytesampere, 0, bytesRec);
                    double ampere aktuell = Math.Round(Double.Parse(txtAmpere.Text,
                        CultureInfo.InvariantCulture) * 1000, 0);
                    txtAmpere.Text = ampere aktuell.ToString("0", CultureInfo.
                        InvariantCulture);

                    sender.Shutdown(SocketShutdown.Both);
                    sender.Close();
                }
    }
```

Quelltext 4.7: C#-Code für das Senden und Empfangen von SCPI-Befehlen

In Quelltext 4.7 wird ein gekürzter Quellcode zur Kommunikation mit einem Keysight 34465 A Multimeter in der Funktion als Amperemeter dargestellt. Dazu wird vom entsprechenden Timer die Methode „SocketConnectionAmperemeter()“ aufgerufen, dabei wird eine neue Instanzierung der Klasse „Socket“ erstellt und mit Daten aus dem Reiter „Konfiguration“ befüllt, dies betrifft vor allem die IP-Adresse. Das Keysight Multimeter wird im Gegensatz zum

GWInstek Multimeter nicht über den Port 3001, sondern über den Port 5025 angesteuert. Anschließend wird versucht, die Verbindung zu öffnen und bei Erfolg werden die Ampeln weiterhin grün eingefärbt. Im ersten Schritt wird über das Bytearray „msg“ der Befehl „*CLS\n“ im ASCII-Bytecode versendet. Dieser Befehl wird in Kombination mit dem nachfolgenden Befehl „*RST\n“ verwendet, um das Gerät zurückzusetzen. Durch den nachfolgenden Befehl „MEAS:CURR:DC? 10 \n“ wird der Messbereich im Zehnerpotenzenbereich für das Messen der Stromstärke konfiguriert und veranlasst, dass ein entsprechender Wert zurückgesendet wird. Durch die Methode „.Receive()“ wird der empfangene ASCII-Bytecode in die Variable „bytesampere“ vom Typ Bytearray gesichert und in den höherwertigen Datentyp integer in die Variable „bytesRec“ rückgesichert. Anschließend werden die empfangenen Ampere in Milliampere umgerechnet und in das Textfeld „txtAmpere“ für den Anwender lesbar eingefügt. Im letzten Schritt wird die Verbindung über die Methode „.Shutdown()“ und „.Close()“ geschlossen.

Die in Kapitel 2.2.3 Ethernet dargestellte Debugging Telnet Sitzung, welche mittels Wireshark aufgezeichnet wurde, entspricht einem derzeit im ACC angewandten Hack, um das GWInstek GDM-9061 Multimeter performanter verwenden zu können. Das genannte Multimeter benötigt im Gegensatz zum Keysight Multimeter deutlich mehr Zeit, um auf Anfragen reagieren zu können und der rückgesendete ASCII-Code benötigt eine zusätzliche Behandlung, um für die erfolgreiche Messtechnik zum Einsatz zu kommen.

```
    SocketConnectionVoltmeter()
                {
                        Thread.Sleep(100);
                        byte[] msg = Encoding.ASCII.GetBytes("*cls \r\n");
                        int bytesSent = multisender.Send(msg, SocketFlags.None);

                        Thread.Sleep(100);
                        msg = Encoding.ASCII.GetBytes("*rst \r\n");
                        bytesSent = multisender.Send(msg, SocketFlags.None);

                        Thread.Sleep(100);
                        msg = Encoding.ASCII.GetBytes("configure:voltage:dc 100 \r\n");
                        bytesSent = multisender.Send(msg, SocketFlags.None);

                        Thread.Sleep(100);
                        msg = Encoding.ASCII.GetBytes("meas:volt:dc? \r\n");
                        bytesSent = multisender.Send(msg);

                        Thread.Sleep(100);
                        int bytesRec = multisender.Receive(buffer);
                        string bytesRecString = "";
                    }
```

Quelltext 4.8: C#-Code mit SCPI-Befehlen zur Spannungsmessung auf einem GWInstek GDM-9061 (1/2)

Der in Quelltext 4.8 abgebildete Quellcode entspricht einer gekürzten Fassung von der in Quelltext 4.7 angewandten Logik, allerdings für das GWInstek Multimeter und zum Messen der Spannung. Ein grundlegender Unterschied besteht in der Notwendigkeit des Einfügens von Pausen, damit das GWInstek die Befehle verarbeiten kann. Des Weiteren kommt im Hintergrund das Betriebssystem Windows vermutlich zum Einsatz, da die Notwendigkeit des „\r \n“-Zeilenendes besteht [42].

Ein weiterer Unterschied besteht in der Notwendigkeit, die SCPI-Befehle kleingeschrieben zu übergeben.

```
    SocketConnectionVoltmeter()
                {
                        while (true)
                        {
                           Thread.Sleep(100);
                           msg = Encoding.ASCII.GetBytes("meas:volt:dc? \r\n");
                           bytesSent = multisender.Send(msg, SocketFlags.None);

                           bytesRec = multisender.Receive(buffer);
                           bytesRecString = Encoding.ASCII.GetString(buffer, 0, bytesRec);

                           if (bytesRecString.Contains("+"))
                           {
                               break;
                           }
                        }

                        txtSpannung.Text = bytesRecString;

                        multisender.Shutdown(SocketShutdown.Both);
                        multisender.Close();
                }
```

Quelltext 4.9: C#-Code mit SCPI-Befehlen zur Spannungsmessung auf einem GWInstek GDM-9061 (2/2)

Der in Quelltext 4.9 dargestellte Quellcode kommt direkt unterhalb von Quelltext 4.8 zur Anwendung. Da das GWInstek Multimeter deutlich langsamer reagiert, wird nach der Abarbeitung von Quelltext 4.8 erst das Echo des Telnet-Servers ausgegeben, also die Willkommensnachricht. Da allerdings die Rückgabewerte der Spannungsmessung benötigt werden, ist eine BruteForce-Anfrage bis zur korrekten Antwort notwendig (=wiederholte Anfrage). Rückgabewerte des GWInstek GDM-9061 beginnen in der Regel mit einem „+“, daher wird gelauscht, bis der entsprechende Rückgabewert dieses Zeichen beinhaltet. Danach wird die Verbindung analog zu der in Quelltext 4.7 beschriebenen Weise geschlossen und es wäre für die korrekte Schließung der Sitzung eine weitere Pause von 700 ms notwendig. Auf diese wird verzichtet, um die Durchlaufgeschwindigkeit zu erhöhen, wodurch es zu der in Abbildung 2.3 rot markierten Zeile kommt.

4.2.3 Aardvark

Das Aardvark, in die deutsche Sprache übersetzt Erdferkel, ist ein I2C/SPI Adapter, welcher über die USB-Verbindung mit dem Computer kommuniziert. Treiber stehen für Windows, Linux und Mac zur Verfügung und ermöglichen einen reibungslosen Kommunikationsaufbau mit den gängigen modernen Betriebssystemen [43].

Relevant für die Auswahl des Aardvarks für das ACC war die Unterstützung der Programmiersprache C# und eine solide Dokumentation. Neben C# werden Python und C unterstützt [44] [45].

Um das Aardvark in das ACC einzubinden, ist es notwendig, für die Entwicklung die in der API mitgelieferte „aardvark.cs“-Datei in das C# Projekt aufzunehmen. Des Weiteren muss der USB-Treiber installiert werden und für die Produktion muss die kompilierte Aardvark Bibliothek „aardvark.dll“-Datei entweder in einen Ordner in der „Path“-Umgebungsvariable eingepflegt, beispielsweise „C:\Windows \System32“ oder im Ordner, in dem die „AccuControlcenter.exe“ ausgeführt wird, auf gleicher Ebene eingebunden sein.

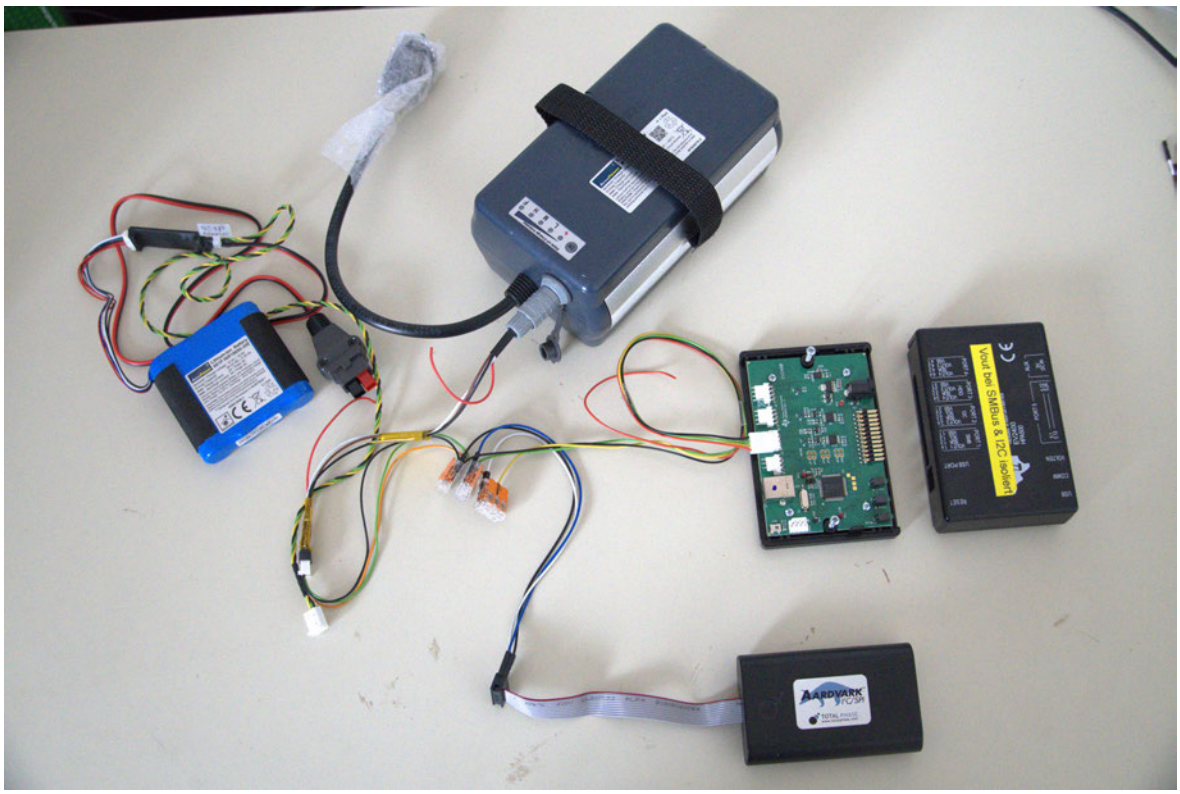


Abbildung 4.12: ACC: Benötigte Hardware für die Buskommunikation

In Abbildung 4.12 werden jene Hardwaregeräte abgebildet, die direkt für die Kommunikation des Produktiveinsatzes für das ACC notwendig sind. Dabei sind in der linken schrägen Bildhälfte ein 4S1P-Akkupack (blau) und rechts oberhalb der größere 8S2P-Akkupack (grau) abgebildet. Die unterhalb ersichtliche Platine entspricht dem EV2400, dieses ist geöffnet, da die Pins für das Debugging erreicht werden müssen, die rechts daneben platzierte Hülle gibt Aufschluss darüber, dass der 8S2P-Akkumulator über den I^2C-Port angesteckt ist. Unterhalb mit dem Flachbandkabel befindet sich das Aardvark. Die Verbindung erfolgt mittels Wago-Klemmen, um gleichzeitig das Aardvark und das EV2400 im Einsatz haben zu können. Der 4S1P-Akku würde einer Veränderung des beim EV2400 verwendeten Ports benötigen, dieser müsste auf den SMBus abgeändert werden.

Beim Debugging der maschinennahen Schnittstellen gibt es mehrere Fehlerquellen, diese können von der Software bis zur Hardware reichen. Daher ist es notwendig, auf beiden Ebenen über eine Analysemöglichkeit zu verfügen. Im Falle des ACC kommen das Rigol 1054 Z Oszilloskop für die Hardwarefehlerquellen und ein Saleae Logic 16 Klon für die Softwareebene zum Einsatz. Für Letzteres ist es essentiell, eine zusätzliche Software herunterzuladen und zu installieren, die sogenannte „Logic 2“-Software [46] [47] [48].

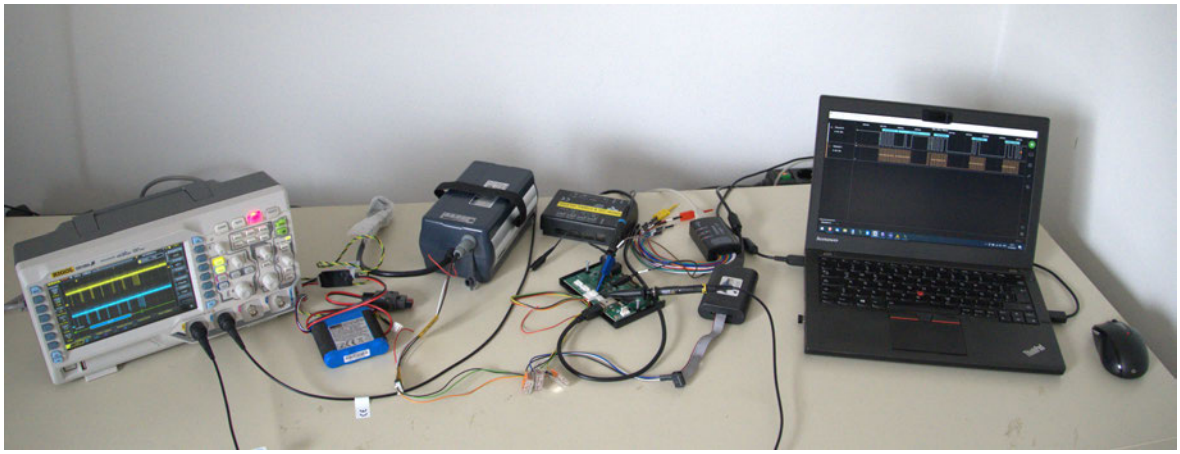


Abbildung 4.13: ACC: Benötigte Hardware für das Debugging der Buskommunikation

Abbildung 4.13 stellt eine Erweiterung von Abbildung 4.12 dar. In der linken Bildhälfte befindet sich das Rigol 1054 Z Oszilloskop. Dieses besitzt eine Decoderfunktion für das I^{2}C-Protokoll. Rechts befindet sich ein Windows 10 Laptop, an welchem der Saleae Logic 16 Klon angeschlossen ist, dieser befindet sich oberhalb des Aardvarks. Auf dem Laptop ist die„Logic 2“-Software gestartet. Bei der EV2400 Platine laufen die Messspitzen für das Oszilloskop und den Logic 16 Klon zusammen. Channel 1 (gelb) beim 1054 Z und Channel 0 (blau) in der „Logic 2“-Software entsprechen der Datenleitung SDA und umgekehrt Channel 2 (blau) und Channel 1 (orange) der Taktgeberleitung SCL. Ausgelesen wird der 8S2P-Akkupack und das Oszilloskop stellt seine Nützlichkeit unter Beweis, wenn gewisse Flanken von der Software nicht erkannt werden und entsprechend die Fehlerquelle beim Akku und dessen Prozessor liegt.

Die Adressierung I^{2}C-Protokoll kann anfangs sehr komplex erscheinen. Dies liegt an der 7-bit, 8-bit oder 10-bit Adressierung, die von jedem Hersteller unter Umständen anders angewendet wird. Offiziell werden die 7-bit Adressen und die später hinzugefügten 10-bit Adressen unterstützt, die 8-bit Adresse bildet sich aus dem letzten Bit, dass entweder durch eine 0 die Adresse zu einer Schreibadresse und mit einer 1 zu einer Leseadresse formt [49] [19].

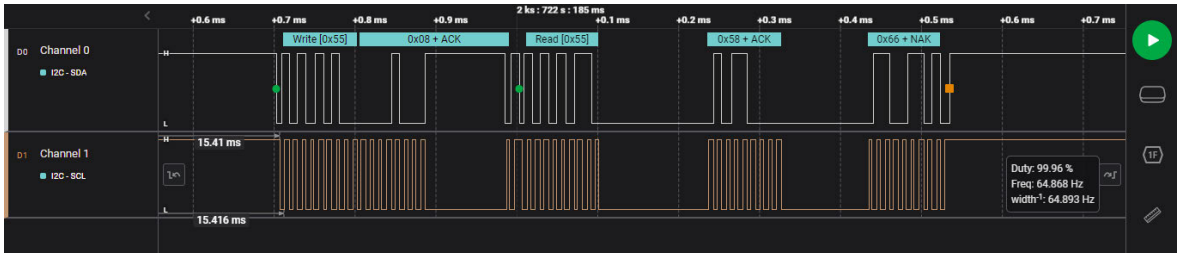


Abbildung 4.14: Aardvark: Auslesen des I^{2}C-Buses mit der „Logic 2“-Software

In der Abbildung 4.14 wird mittels des Aardvark von einem TI bq34z100g1 Prozessor das Register „0x08“ angefordert. Die 7-bit-Adresse des Prozessors entspricht dem Hex-Wert „0x55“. Das „0x08“-Register enthält den Wert der Spannung und aus der Abbildung geht hervor, dass die Lese- und Schreibadresse jeweils der „0x55“-7-bit-Adresse entsprechen. Daraus wird ersichtlich, dass die Software „Logic 2“ auf das letzte Bit bei der Darstellung verzichtet, um die Adresse als lesend oder schreibend zu kennzeichnen. Das Register „0x08“ enthält folgende Werte: „0x58“ und „0x66“. Diese müssen in den Little-Endian-Wert „0x6658“ umgerechnet

werden und entspricht in der Dezimaldarstellung „26200“, also 26.200 mV. Dies ist jedoch noch nicht korrekt. Um den korrekten Wert zu erhalten, muss der Wert „0x58“ um den Betrag des ACK (Acknowledge) bereinigt werden. Das bedeutet, der Hex-Wert „0x58“ wird in seinen Binärwert rückgerechnet und dies entspricht „0101 1000“. Der Wert des ACK wird entfernt und der Binärwert „0101 0000“ errechnet, in der Hex-Darstellung entspricht dies „0x50“. Im Anschluss wird der Little-Endian-Wert erneut berechnet, diesmal mit den korrekten Werten und ergibt „0x6650“, was in das Dezimalsystem rückgerechnet „26192“, also 26.192 mV entspricht.

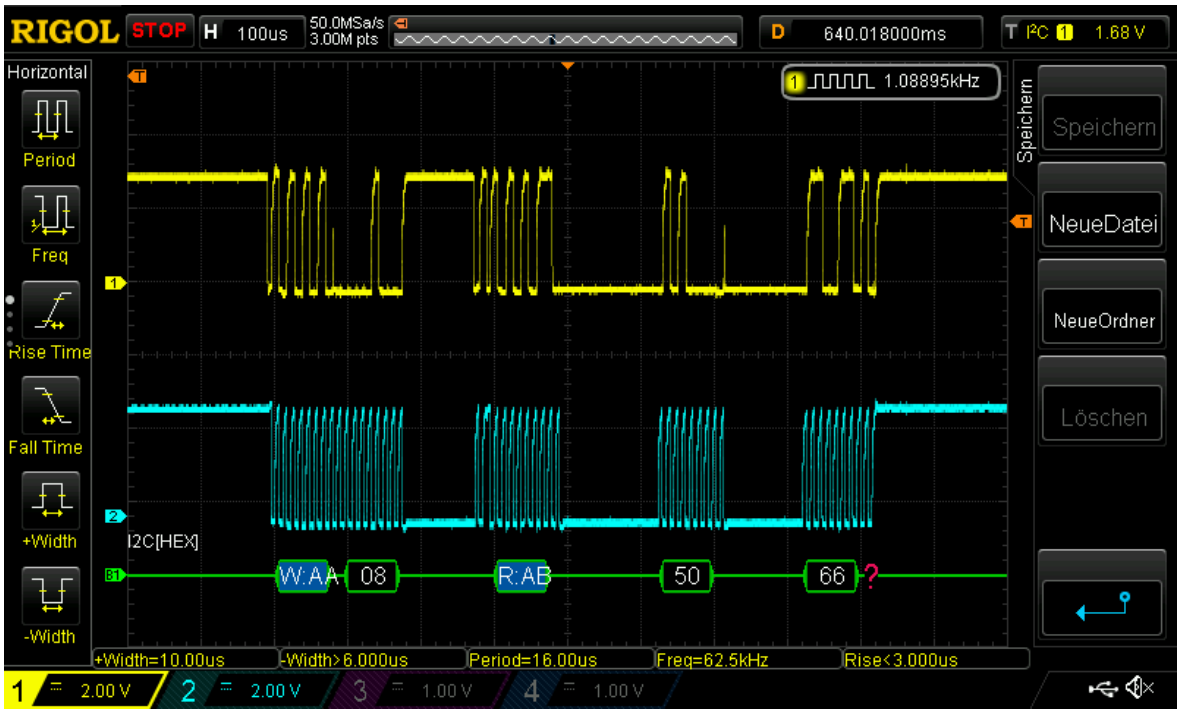


Abbildung 4.15: Aardvark: Auslesen des I²C-Buses mit dem Rigol 1054 Z

Im Gegensatz zur in Abbildung 4.14 dargestellten „Logic 2“-Software verwendet das Rigol 1054 Z Oszilloskop im Decoder das letzte Bit, um eine Lese- oder Schreibadresse zu bilden, also eine 8-bit-Adresse. Aus der 7-bit-Adresse „0x55“ (in Binärdarstellung „0101 0101“), wird die Schreibadresse „0xAA“ (in Binärdarstellung „1010 1010“) und die Leseadresse „0xAB“ (in Binärdarstellung „1010 1011“). Das ACK (Acknowledge) Signal findet keine Verwendung und daher kann sofort der korrekte Little-Endian-Wert „0x6650“ gebildet und in die Dezimaldarstellung „26192“, also 26.192 mV, umgerechnet werden.

Das ACC unterstützt folgende TI Prozessoren: bq40z80, bq40z50R1, bq40z50R2, bq78350, bq78350R1, bq78350R2 und bq34z100g1. Der letzte Prozessor bq34z100g1 weicht im Gegensatz zu den vorangegangen Prozessoren bei den für das ACC relevanten Registern (Spannung, Strom, ...) von der Adressierung ab [50] [51] [52] [53] [54] [55] [56].

Eine Übersicht der Register findet sich in der nachfolgenden Tabelle 4.2.

Tabelle 4.2: Hex-Adressen von benötigten Werten auf unterschiedlichen TI-Prozessoren

Bezeichnung	bq34z100g1	bq40z80, bq40z50R1, bq40z50R2, bq78350, bq78350R1, bq78350R2	Einheit und Erläuterung
Spannung	0x08	0x09	mV - gibt die momentane Spannung aus.
Kapazität	0x04	0x0F	mAh - gibt die absolute Kapazität aus.
Temperatur	0x0C	0x08	Kelvin - Umrechnung in °C notwendig.
SoC	0x02	0x0D	% - relativer Ladezustand.
Strom	0x10	0x0A	mA - gibt den momentanen Strom aus.
Flashdatum	0x40	0x1B	Integer - Rückrechnung mittels einer Formel in ein Datum notwendig.
Akkustatus	0x00	0x44	Bytearray - Status muss aus dem Array herausgefiltert werden.

Die genannten Register können durch folgende Initialisierung des Aardvark innerhalb des ACC abgefragt werden:

```
int handle;
int port = 0;
int bitrate = 100;
int bus timeout;
byte device = 0x0B; //0x55 beim bq34z100g1
```

Quelltext 4.10: C#-Code zur Initialisierung eines Aardvark

Die Variable „handle“ dient nachfolgend als Identifikator für das Aardvark, nachdem es über die Variable „port“, welche standardmäßig auf 0 gesetzt ist (außer es werden 2 Aardvarks parallel betrieben), initialisiert wurde. Die Bitrate entspricht 100 kHz und das Timeout ist auf 150 ms konfiguriert (die Zuordnung erfolgt im späteren Quelltext). Die 7-bit-Adresse entspricht „0x55“ beim bq34z100g1 und bei den anderen Prozessoren dem Hex-Wert „0x0B“.

```
double temperatur = Math.Round((double.Parse( readMemory(handle, device, 0
    x08, 2).ToString(), CultureInfo.InvariantCulture.NumberFormat) / 10) -
     273.15, 2);
txtI2CTemperatur.Text = temperatur.ToString();
```

Quelltext 4.11: C#-Code für die Temperaturberechnung

In Quelltext 4.11 erfolgt die Abfrage der Temperatur, diese ist bis auf das Register bei allen Prozessoren ident. Über die Methode „_readMemory(handle, device, 0x08, 2).ToString()“ wird ein Schreib- und anschließender Lesevorgang auf das initialisierte Aardvark („handle“) gestartet. Dabei wird das Gerät mit der 7-bit-Adresse („device“) „0x0B“ angefragt, die Werte des

Registers „0x08“ zu senden und zwar die ersten 2 Byte. Diese werden entsprechend in den Little-Endian-Wert umgewandelt und in den String-Datentyp konvertiert, um abschließend in den Double-Datentyp konvertiert zu werden. Anschließend wird durch die überflüssige Zehnerpotenz dividiert, der absolute Nullwert (273,15) subtrahiert und auf 2 Nachkommastellen gerundet. Durch diesen Vorgang werden für den Menschen leserlich °C als Einheit ausgegeben.

Die Abfrage des Flashdatums unterscheidet sich dabei stark zwischen dem bq34z100g1-Prozessor und den bq40z80, bq40z50R1, bq40z50R2, bq78350, bq78350R1, bq78350R2 Prozessoren. Die Formel zur Rückgewinnung des Datums lautet:

$$Day + Month * 32 + (Year - 1980) * 512 \tag{4.1}$$

und ist für alle genannten Prozessoren gültig. In der Dokumentation des bq34z100g1 befindet sich jedoch ein fehlerhafter Eintrag. Laut Dokumentation würde die Formel lauten:

$$Day + Mo * 32 + (Yr - 1980) * 256 \tag{4.2}$$

[vgl. 56, S. 25]. Allerdings wurde im TI Forum bestätigt, dass die oben genannte Formel korrekt ist [57].

```
readMemory(handle, device, 0x1B, 2).ToString()
```

Quelltext 4.12: C#-Code zum Auslesen des Flashdatums bei unterschiedlichen TI-Prozessoren

Der Quelltext 4.12 stellt die einfache Art der Abfrage für alle Prozessoren, die nicht dem bq34z100g1 entsprechen, dar. Es werden einfach die 2 Bytes des Registers „0x1B“ in den Little-Endian-Wert geformt und in das Dezimalsystem rückgerechnet.

Die Abfrage beim bq34z100g1 ist komplexer, da kein einzelnes Register für das Flashdatum oder das „Manufacture Date“ gegeben ist. Das bedeutet, es muss ein allgemeines Register abgefragt werden. In diesem Register befinden sich unterschiedliche Daten zum Akkumulator und entsprechend sind die benötigten Daten die Minorität und müssen herausgefiltert werden, dazu wurde die originale „_readMemory()“-Methode der Aardvark-Bibliothek erweitert.

```
flashdatumI2C =  readMemoryReturnByteSpezifischFlashdatum(handle, device, 0x40, 4, 2,
    3).ToString();
```

Quelltext 4.13: C#-Code zum Auslesen des Flashdatums beim Prozessor: bq34z100g1 (Aufruf der Funktion)

In Quelltext 4.13 wird der Aufruf der modifizierten Methode zum Auslesen des Flashdatums für den bq34z100g1-Prozessor abgebildet. Das Register „0x40“ beinhaltet die „Manufacturer Data“ oder Herstellerdaten. Die 4 bezieht sich auf die Länge der zu schreibenden Daten und 2 und 3 entsprechen den Stellen, an denen die Bytes im Register „0x40“ stehen. Diese werden anschließend wieder in den Little-Endian-Wert umgerechnet und im letzten Schritt in das Dezimalsystem rückgerechnet, um die Formel (4.1) anzuwenden. Dadurch wird es möglich zu überprüfen, ob auf der BMS das korrekte Herstellerdatum bzw. Flashdatum gesetzt ist,

um künftigen Garantie- bzw. Haftungsansprüchen genüge zu tun. Im Falle eines nicht übereinstimmenden Datums wird eine Fehlermeldung ausgegeben und die Checkbox wird nicht gesetzt, des Weiteren wird auch ein akustischer Fehlerton abgespielt und ein visuelles Signal durch die Fehlermeldung gegeben.

```
public int  readMemoryReturnByteSpezifischFlashdatum(int handle, byte device, byte
     addr,  short length, int byte an der Stelle, int byte zwei an der Stelle)
{
    int count, i;
    int erg = 0;
    byte[] dataOut = { addr };
    byte[] dataIn = new byte[length];

    // Write the address
    AardvarkApi.aa i2c write(handle, device,
                             AardvarkI2cFlags.AA I2C NO FLAGS,
                             (ushort)length, dataOut);

    AardvarkApi.aa sleep ms(200);

    count = AardvarkApi.aa i2c read(handle, device,
                                    AardvarkI2cFlags.AA I2C NO FLAGS,
                                    (ushort)length, dataIn);

    string hex1 = dataIn[byte an der Stelle].ToString("X");
    string hex2 = dataIn[byte zwei an der Stelle].ToString("X");

    string hex summe = hex1 + hex2;
    int flashdatum local = Convert.ToInt16(hex summe, 16);

    return Convert.ToInt32(flashdatum local);
}
```

Quelltext 4.14: C#-Code zum Auslesen des Flashdatums beim Prozessor: bq34z100g1 (Code der Funktion)

Quelltext 4.14 zeigt die Methode „_readMemoryReturnByteSpezifischFlashdatum(int handle, byte device, byte addr, short length, int byte_an_der_Stelle, int byte_zwei_an_der_Stelle)“. Im ersten Schritt wird dem bq34z100g1-Prozessor mitgeteilt, welches Register angesprochen wird („0x40“ in der Variable „dataOut“). Danach wird eine Pause von 200 ms eingelegt, um dem Prozessor die notwendige Zeit zur Berechnung zu zugestehen. Anschließend wird auf dem I^2C-Bus der Antwort gelauscht und die Daten werden in das Bytearray „dataIn“ geschrieben. Die „count“-Variable dient zum Fehler abfangen (dieser Teil wurde aus dem Quelltext 4.14 gekürzt). Die empfangenen Daten werden dann als String in die Variable „hex_summe“ zusammengesetzt und anschließend als Integer-Datentyp zurückgegeben.

Bevor die genannte Funktion zum Einsatz kommen kann, muss sichergestellt werden, dass der bq34z100g1-Prozessor die korrekten Daten liefert. Dazu wird der Chip veranlasst, den Zugang zum benötigten Herstellerdatenblock zu gewähren. Dafür ist eine Reihe von Anfragen an die BMS notwendig. Im ersten Schritt wird das Register „0x61“ mit „00“ beschrieben. Im zweiten Schritt wird das Register „0x3E“ mit „30“ beschrieben und im letzten Schritt wird das Register „0x3F“ mit „00“ beschrieben. Anschließend ist es möglich, das Register „0x40“ und die benötigten Daten auszulesen. Dies gilt für die Seriennummer, den Akkustatus (offen, gesperrt, halb offen) und das Flashdatum. Dieser Vorgang ist darauf zurückzuführen, dass das Register zusätzlich zur Authentifizierung verwendet wird und entsprechend einer dualen Verwendung

eingeordnet ist, wenn der Akku gesperrt ist. Wenn der Akku ungesperrt ist, ist es möglich, den Datenspeicher auszulesen und zu beschreiben (im Register „0x40“). Daraus geht hervor, das im ungesperrten Zustand mehr Möglichkeiten als das Auslesen der Herstellerdaten bestehen [vgl. 56, S. 18] [vgl. 58, S. 6].

```
byte[] write reg addr = { 0x61, 0x00 };
ushort write reg addr length = 2;
AardvarkApi.aa i2c write(handle, device,
         AardvarkI2cFlags.AA I2C NO FLAGS,
         (ushort)write reg addr length, write reg addr);
AardvarkApi.aa sleep ms(200);

byte[] write reg addr subclass address = { 0x3E, 0x30 };
write reg addr length = 2;
AardvarkApi.aa i2c write(handle, device,
         AardvarkI2cFlags.AA I2C NO FLAGS,
         (ushort)write reg addr length, write reg addr subclass address);
AardvarkApi.aa sleep ms(200);

byte[] write reg addr general purpose block = { 0x3F, 0x00 };
write reg addr length = 2;
AardvarkApi.aa i2c write(handle, device,
         AardvarkI2cFlags.AA I2C NO FLAGS,
         (ushort)write reg addr length, write reg addr general purpose
             block);
AardvarkApi.aa sleep ms(200);
```

Quelltext 4.15: C#-Code zur Abfrage des Herstellerdatenblocks auf dem Prozessor: bq34z100g1

In Quelltext 4.15 wird das oben beschriebene Verfahren zum Auslesen der Herstellerdaten in C#-Code abgebildet. Durch diese Sequenz wird sichergestellt, dass auf die Herstellerdaten und nicht auf den Datenflashspeicher zugegriffen wird. Auf jeden Schreibvorgang folgt eine kurzzeitige Pause und anschließend ist es möglich, den nächsten Vorgang einzuleiten.

Zum Auslesen des Akkuzustandes, ob dieser „gesperrt“, „halb offen“ oder „offen“ ist, wird aus dem Register „0x00“ ein Byte benötigt. Dieses befindet sich an der ersten Stelle.

```
int status =  readMemoryReturnByteSpezifisch(handle, device, 0x00, 2, 1);

if (status == 96)
{
    prozessor zugang = "gesperrt";
}
else if (status == 64)
{
    prozessor zugang = "halb offen";
}
else if (status == 0)
{
    prozessor zugang = "offen";
}
else
{
    prozessor zugang = "undefiniert";
}
```

Quelltext 4.16: C#-Code zum Auslesen des Akkumulatorenzustandes

Die Variable „status“, im Quelltext 4.16 zu finden, enthält das Statusbyte aus dem Register „0x00“. Dieses empfangene Byte wird durch die Methode „_readMemoryReturnByteSpezifisch()“ direkt in einen Integer-Datentyp umgewandelt. Dadurch muss beim Debuggen darauf geachtet werden, dass das Byte am Bus im Hex-Wert übertragen wird (Dez 96 = Hex 60, Dez 64 = Hex 64, Dez 0 = Hex 0).

Abschließend in der Vergegenwärtigung dessen, dass das ACC mittels des Aardvarks und das bqStudio mittels des EV2400 mit dem BMS des Akkumulators kommuniziert und zwar parallel, sei erwähnt, dass es zu Kollisionen auf dem I^2C-Bus kommen kann. Dadurch können unvorhergesehene Daten empfangen werden und der bq34z100g1 kann in unterschiedlichen Zugangsmodi antworten. Damit die benötigten Daten empfangen werden, wird ein spezifischer technischer Eingriff angewendet - der Akku wird nach jedem Schreib- und Lesezugriff resettet, also in seinen Ausgangszustand gebracht. Dadurch ist das Antwortverhalten kontrollierbar.

```
//Reset des Chips
AardvarkApi.aa sleep ms(200);
byte[] write reg addr reset = { 0x00, 0x00, 0x00, 0x00 };
write reg addr length = 4;
AardvarkApi.aa i2c write(handle, device,
         AardvarkI2cFlags.AA I2C NO FLAGS,
         (ushort)write reg addr length, write reg addr reset);
AardvarkApi.aa sleep ms(200);
```

Quelltext 4.17: C#-Code für den Reset eines Akkumulators

In Quelltext 4.17 wird der genannte Hack als Quellcode abgebildet. Es wird das Register „0x00“ mit „00 00 00“ beschrieben. Dieses Vorgehen ist nirgends dokumentiert, konnte aber beim Auslesen des I^2C-Buses in Verwendung des bqStudios beobachtet werden und hat sich in der praktischen Anwendung als zielführend erwiesen, um mit einer sehr hohen Wahrscheinlichkeit die angeforderten Daten in Empfang zu nehmen.

5 Deploymentprozess

Der Deploymentprozess regelt die Bereitstellung des ACC im Unternehmen AccuPower. Dieser Prozess ist erforderlich, um eine aktuelle Software zu garantieren und bei der Fehlersuche ausschließen zu können, dass eine alte Software verwendet wird. Zum Einsatz kommt dafür Git als Instrument zur Quellcodeverwaltung und Microsoft Teams zur Verteilung der Software [59].

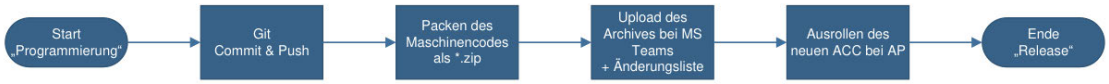


Abbildung 5.1: Schematischer Ablauf des Deployments

In Abbildung 5.1 findet sich der schematische Ablauf des Deploymentprozesses. Beginnend bei der Programmierung erfolgt nach Fertigstellung der aktuell gestellten Anforderungen ein „git commit && git push“. Das bedeutet, der Quellcode wird in die Quellcodeverwaltung GitLab hochgeladen. Eine detaillierte Beschreibung findet sich in der nächsten Sektion. Daran anschließend erfolgt das Bündeln der benötigten Dateien als Archivdatei, um diese in MS Teams hochzuladen. Der dabei entstehende Kommentar in MS Teams erhält eine Liste der aktuellen Anpassungen und Änderungen und anschließend wird die neue Software bei AccuPower ausgerollt.

5.1 Quellcodeverwaltung

Der Quellcode wird mittels Git verwaltet. Git wurde 2005 von Linus Torvalds, dem Entwickler des Linux Kernels, entwickelt und verfolgt u. a. folgende Ziele: ein schnelles, simples und nicht-lineares Open Source Softwareversionsverwaltungswerkzeug zu sein [60].

Die Firma Webgeyer betreibt einen eigenen Git-Server, dieser wird von der Software GitLab bereitgestellt. In GitLab können entsprechende Repositories (Programmierprojekte) angelegt und verwaltet werden. Es ist des Weiteren möglich, ganze Deployments auf Servern vollständig zu automatisieren. Dazu können GitLab-Runner verwendet werden, diese werden allerdings nicht für das ACC-Projekt verwendet [61] [62].

Jeder Softwareentwickler steht früher oder später vor der Frage, wie der Quellcode bestmöglich verwaltet werden kann. Ohne Softwareversionsverwaltungssoftware ist es sehr schwer möglich, Änderungen in der Software zu dokumentieren und zu archivieren, da dieses Thema sehr komplex wird. Wenn beispielsweise ein Kunde einen Softwarefehler entdeckt, ist es deutlich einfacher, auf einen älteren funktionierenden Quellcode zu wechseln und auch die Änderungen anzusehen und somit den Fehler zu entdecken. Durch die Verwendung eines

eigenen GitLab-Servers ist der Quellcode zur Gänze unter der Datenhoheit von Webgeyer und entsprechend besteht ein deutlich geringeres Risiko, dass der Quellcode entwendet oder von KIs eingelesen wird [63].

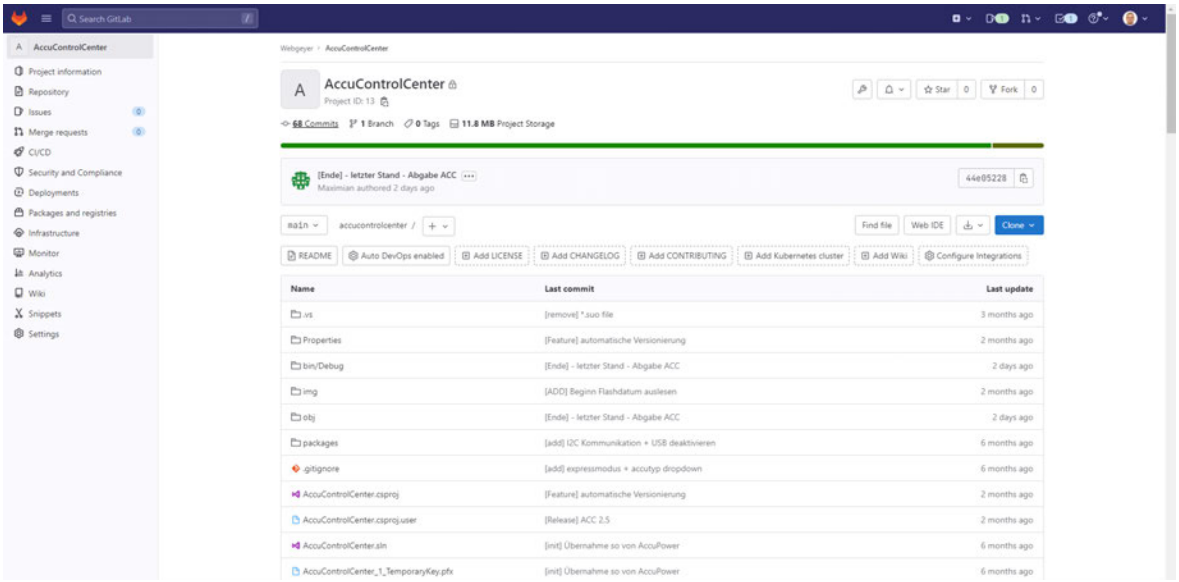


Abbildung 5.2: GitLab: ACC Repository Metadaten

In Abbildung 5.2, erstellt am 06.05.2023, erkennt man anhand des grünen Balkens im oberen Bildbereich, dass 93.3% des Quellcodes in der Programmiersprache C# geschrieben sind. Die restlichen 6.7% sind, einer fehlerhaften Interpretation von GitLab zu Folge, in der Programmiersprache Smalltalk geschrieben (Olivgrüner Balken). Unterhalb der Balken wird ersichtlich, dass derzeit 68 Commits, also Updates für den Quellcode des ACC erfolgt sind und dass das gesamte Repository fast 12 MB an Speicher benötigt. Da aktuell nur ein Programmierer an diesem Projekt arbeitet, wird kein 2ter Branch, beispielsweise ein „Dev“-Branch verwendet. Ein Branch entspricht der Verzweigung eines Baumes, vom Stamm ausgehend kann ein Softwareprojekt in einem „Main“- und einem „Dev“-Branch weitergeführt werden. Sobald der Quellcode im „Dev“-Branch stabil ist, ist es möglich, mittels eines „Merges“, einer Zusammenführung den „Dev“- in den „Main“-Branch zu integrieren und somit bleibt der stabile Quellcode vom entwickelten Quellcode unberührt, bis dieser für den Produktivbetrieb freigegeben wird. Des Weiteren ist in der Webansicht von GitLab ersichtlich, welche Datei und welcher Ordner von welchem Update (commit) die letzten Änderungen erfahren hat. Der Ordner „.vs“ wurde das letzte Mal im Commit „[remove] *.suo file“ im Februar 2023 einer Veränderung unterzogen. Der „bin/Debug„-Ordner wurde hingegen das letzte Mal am 04.05.2023 im Commit „[Ende] - letzter Stand - Abgabe ACC“ bearbeitet, dieser Commit entspricht wie bereits im Namen erkenntlich, der aktuellen Abgabe dieses Softwareprojektes.

Für die Verwaltung des Quellcodes auf Windows kommt der Git-Client von Github „GitHub Desktop“ zum Einsatz. Die genannte Software ist sehr einfach in der Handhabung und bietet eine gute Übersicht über die Änderungen im Quellcode des Weiteren ist sie Open Source [64].

Bei der Programmierung auf Linuxrechnern ist Git u. a. in den OpenSource Codeeditor von Microsoft „Visual Studio Code“ integrierbar und daher ist es nicht notwendig, eigens einen Client zu installieren. Allerdings ist die Übersichtlichkeit über die Commits nicht so komfortabel wie in einem eigenen Git-Client [65].

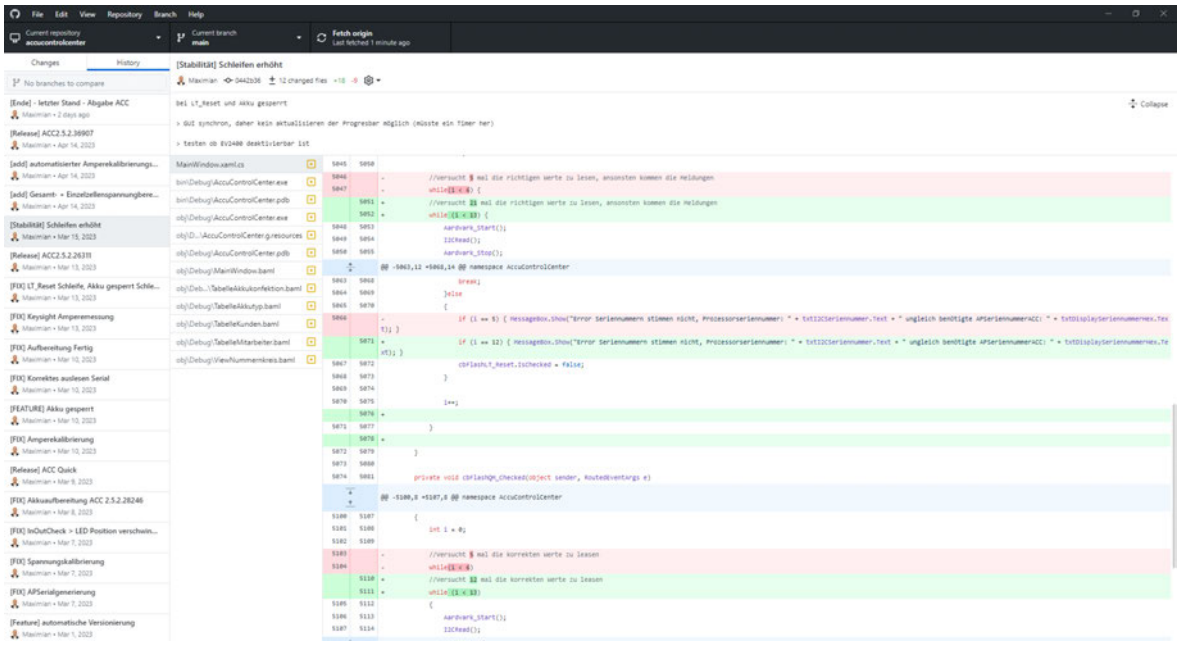


Abbildung 5.3: GitHub-Desktop: ACC Commit History

Abbildung 5.3 stellt den Updateverlauf (Commits) des ACC im „GitHub Desktop“-Client dar. Ersichtlich sind im oberen Bildabschnitt das gewählte Repository „accucontrolcenter“, der Branch „main“ und der letzte Datenabgleich mit dem Git-Server „Fetch origin“, welcher vor 1 Minute das letzte Mal durchgeführt wurde. Durch die blaue Unterstreichung der Registerkarte „History“ ist erkenntlich, dass diese sichtbar ist. In der Registerkarte „Changes“ ist ein Überblick über aktuelle Änderungen im Quelltext des Softwareprojektes ersichtlich, diese ist analog in der Art der Darstellung zum Tab „History“ und listet entsprechend alle veränderten Dateien und den betreffenden Quelltext auf. Des Weiteren ist es in der genannten Registerkarte möglich, die durchgeführten Änderungen hochzuladen und zu kommentieren, dies entspricht einem „git commit“ auf der Kommandozeile und einem anschließenden „git push“ zum Hochladen der Änderungen. Der aktiv markierte Commit „[Stabilität] Schleifen erhöht“ beinhaltet 12 veränderte Dateien. Davon ist die „MainWindow.xaml.cs“-Datei jene, die den Quellcode beherbergt. Alle anderen Dateien werden nach einem Kompiliervorgang erzeugt und sind entsprechend von dieser abhängig. Über der angezeigten Datei „MainWindow.xaml.cs“ findet sich das Commit mit seiner Beschreibung, um nachvollziehen zu können, was geändert wurde und der größte Bildausschnitt rechts zeigt den Quellcode der geöffneten Datei an. In diesem ist anhand der rot und grün markierten Zeilen erkenntlich, was geändert wurde. Rot bedeutet eine Streichung und Veränderung und grün hinterlegte Zeilen sind jene mit der aktuell gültigen Veränderung.

Eine weitere Möglichkeit von git verwaltetem Quellcode besteht darin, innerhalb eines Repositorys ein weiteres git Repository hinzuzufügen. Durch die Ausführung des folgenden Terminalbefehls „git submodule update –init –recursive“ ist es dann möglich, eine halbautomatisierte Updatefunktionalität zu verwenden, um den Quellcode des hinzugefügten Repositories aktuell zu halten und etwaige Sicherheitslücken rasch zu schließen [66].

5.2 Release der neuen Softwareversion

Zur eindeutigen Identifikation jedes Softwarereleases beinhaltet diese in der Registerkarte „Informationen“ eine eindeutige Identifikationsnummer, die sogenannte Assemblyversion. Diese besteht aus 4 Segmenten, wobei die ersten drei manuell festgelegt werden und der letzte Abschnitt wird automatisiert über einen Hash berechnet.

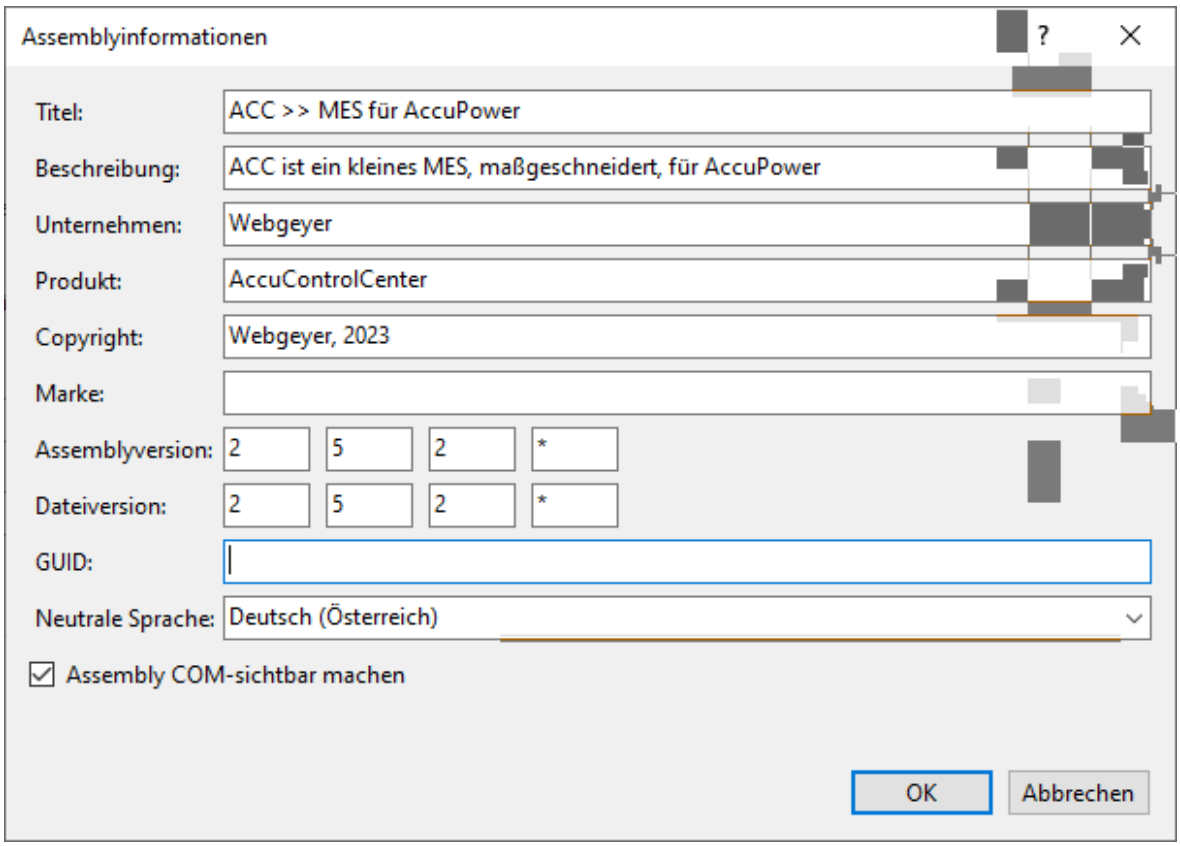


Abbildung 5.4: ACC: Assemblyinformationen

In Abbildung 5.4 sind die genannten Segmente ersichtlich und der letzte Abschnitt ist mit dem Platzhalter Asterisk (*) gefüllt. Diese Konfiguration wird innerhalb des Visual Studio Projektes durchgeführt, ist allerdings nur durch eine Abänderung innerhalb der„*.csproj“ möglich. Die genannte Datei muss mit einem Texteditor geöffnet werden und anschließend müssen folgende Codezeilen, wie abgebildet, adaptiert werden:

```
<Deterministic>false</Deterministic> //ist normalerweise auf ‘true‘
<GenerateSerializationAssemblies>On</GenerateSerializationAssemblies>
```

Quelltext 5.1: Konfiguration für die automatisierte Versionierung eines Visual-Studio-Projektes

Erst durch die Anpassung, dargestellt in Quelltext 5.1 ist die deterministische Versionierung, welche in seit Visual Studio 2017 - Version 15.8.7 aktiviert ist, deaktiviert und die automatische Versionierung wieder möglich [67].

Die genannte Identifikationsnummer wird anschließend für die Namensgebung der Archivdatei verwendet, welche erzeugt wird, um die kompilierten Dateien innerhalb des „bin“-Ordners des Softwareprojektes zu AccuPower zu senden. Das ACC wird ausschließlich seit der Einführung der Aardvarkunterstützung mit den Dateien innerhalb des „Debug“-Unterordners freigegeben. Dies liegt daran, dass das ACC mit Administrationsberechtigungen gestartet werden muss, damit die Möglichkeit der Deaktivierung des Aardvark-Treibers besteht. Eine Verwendung der Dateien des „Release“-Unterordners unterstützt diese Funktionalität nicht ohne weitere Recherche und Programmierung.

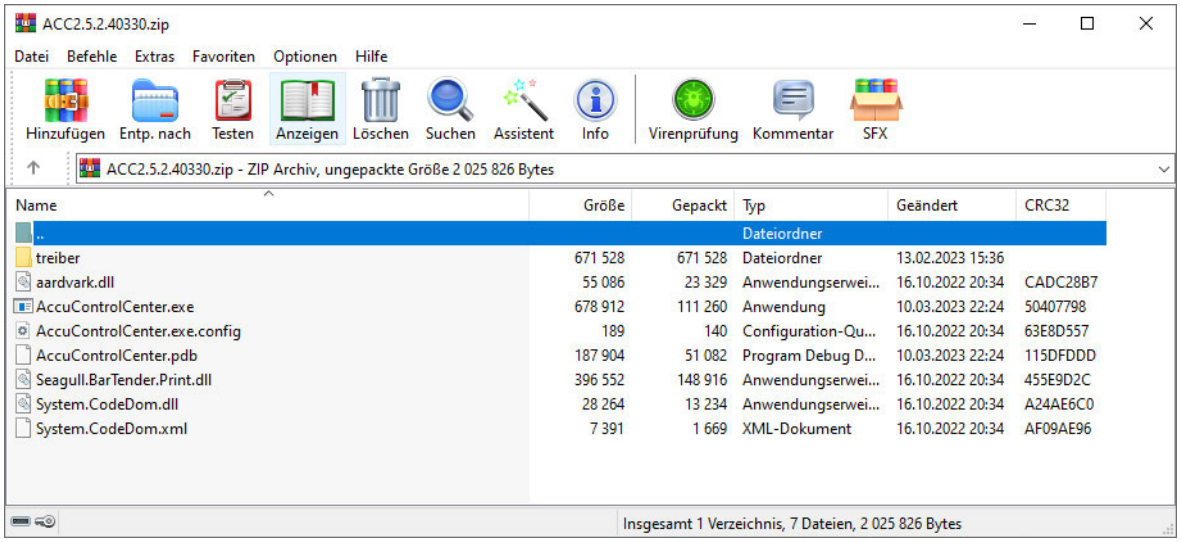


Abbildung 5.5: ACC: Archivdatei

Die in Abbildung 5.5 dargestellte Archivdatei beinhaltet alle notwendigen Treiber, Bibliotheken und Dateien, um das ACC in Betrieb nehmen zu können. Im Ordner „Treiber“ befindet sich der Windows-Treiber des Aardvarks und im Wurzelverzeichnis befinden sich die notwendigen Bibliotheken (*.dll-Dateien) und die „AccuControlCenter.exe“, die der auszuführenden Software entspricht.

Abschließend wird das Archiv in MS Teams mit einem Kommentar betreffend der aktuellen Anpassungen und Veränderungen hochgeladen. Zusätzlich finden sich im Kommentar Hinweise darauf, was zu testen ist und in welchem Bereich etwaige Probleme noch bestehen könnten und die Bitte um Überprüfung dessen mit der Dokumentierung mittels eines Videos oder von Screenshots, damit ein nachfolgendes Debugging leichter vonstattengeht.

6 Fazit

Innerhalb der Entwicklungszeit, ausgehend von der Quellcodeübernahme am 13.10.2022 bis zur Endabgabe am 04.05.2023, wuchs das ACC um 2471 Codezeilen in der Hauptdatei „MainWindow.xaml.cs“, was einer Steigerung oder einem Wachstum von 88% entspricht und um Hunderte weitere Codezeilen in neuen und alten Dateien. Es wurden in der MS SQL Datenbank Tabellen und Sichten erweitert bzw. neu erstellt und viele neue Erweiterungen in das ACC eingepflegt, u. a.:

- Unterstützung des I^2C- und SMBuses
- Unterstützung von GWInstek Multimetern
- Expressmoduserweiterung hinzugefügt
- Tabellenbearbeitung hinzugefügt und Einbau einer neuen Sicht
- Nummernkreisverwaltung hinzugefügt
- Stromkalibrierungsprüfung
- Uvm.

Durch die Automatisierung von Tests konnte die Geschwindigkeit erhöht, Kosten gesenkt und die Arbeitssicherheit und Qualität gesteigert werden. In Zahlen konnte bei der Behandlungsart „Eingangsprüfung“ ein Informationszugewinn um 30% gewonnen werden durch das Einbinden von Informationen, welche direkt vom Prozessor des BMS stammen. Die Behandlungsart „Aufbereitung prog“ konnte von einer ehemaligen Durchlaufzeit von ca. 7 Minuten pro Akkupack auf 3 Minuten pro Akkupack gesenkt werden. Dies entspricht einer Steigerung der Effizienz von 230% und erhöht die Kapazitäten pro Arbeitsplatz von ~1325 Akkus/Monat auf ~3080 Akkus/Monat. In wirtschaftlichen Zahlen, ausgehend von der Annahme eines 1000 € Bruttomonatsgehalt pro Arbeitsplatz, kostet die Stunde Arbeitszeit ca. 6,50 € (1000 € Gehalt / 38,5 Wochenstunden / 4 Wochen) bedeutet dies eine Verringerung der Kosten für die Behandlung von Akkupacks in Höhe von 0,431 €. Diese Verringerung berechnet sich aus dem Stundensatz des Arbeiters 6,50 € dividiert durch die bearbeitete Akkupackanzahl pro Stunde (früher: 8,6 Akkupacks/h und aktuell: 20 Akkupacks/h) und ergibt somit einen früheren Akkubehandlungssatz von 0,756 €/Akku (6,50 € / 8,6 Akkus/h) und einen aktuellen Behandlungssatz von 0,325 €/Akku (6,50 € / 20 Akkus/h). Oder in einfacheren Worten pro Stunde spart sich die Firma AccuPower pro Arbeitsplatz in Vollbetrieb 74,1 € im Vergleich zum alten System (20 Akkus/h - 8,6 Akkus /h = 11,4 Akkus/h * 6,50 € Stundensatz = 74,1 €). Da direkt die Ursache des Flaschenhalses, eben die Behandlungsart „Aufbereitung prog“ gezielt optimiert wurde, kann von einer wirklichen Verbesserung der Produktion der Firma AccuPower geschrieben werden.

6.1 Zusammenfassung

In diesem Kapitel wird eine kurze Zusammenfassung und der Schwerpunkt der in der Arbeit enthaltenen Kapitel gegeben:

Ausgehend vom Kapitel Einleitung, in welchem der Kontext der Arbeit, die Zielsetzung und der grundlegende Aufbau der Arbeit beschrieben werden und somit ein grober Überblick über das Projekt ACC gegeben wird, erweitert jedes nachfolgende Kapitel den Grad an Details, um stufenmäßig das benötigte Wissen und unternommen Schritte zur Umsetzung zu beleuchten. Des Weiteren wird Auskunft darüber gegeben, was von dieser Arbeit erwartet werden kann und was nicht.

Im Folgekapitel Grundlagen wird der Leser erstmalig mit einem Screenshot der Software konfrontiert, um anschließend in den Technologiestack des ACC einzutauchen. Dazu gehören die Programmiersprache, die Datenbank und verschiedene technische Protokolle. Jedes einzelne Unterkapitel vermittelt hierbei einen historischen Hintergrund und einen praktischen Bezug zum ACC.

Das Kapitel Planung und praktische Umsetzung beschreibt anhand der funktionalen und nicht-funktionalen Anforderungen an das ACC und deren praktischer Umsetzung auf der Metaebene die verwendete Hardware und Software. In diesem Kapitel findet sich auch ein SQL-Codeschnipsel und eine Einführung in die Planung des ACC und dessen Verwaltung. Abschließend gibt es im Unterkapitel Systemarchitektur einen Überblick über alle physischen Geräte, welche zum Instrumentarium des ACC gehören.

Der technische Einstieg befindet sich im Kapitel Implementierung. Dieses gibt einen Überblick über das Design des ACC und die Verwendung der jeweiligen Registerreiter, daran anknüpfend wird im Unterkapitel Schnittstellen anhand von Codeschnipseln dargestellt und erklärt, wie die Kommunikation der unterschiedlichen Geräte funktioniert. Dies beginnt bei der Datenbankverbindung, geht weiter zu den Multimetern und der Last und endet bei der Kommunikation zum Prozessor auf der BMS. Es werden unterschiedliche technische Eingriffe dargelegt, um die Effizienz zu steigern und gewünschte Kommunikationsergebnisse zu fördern. Anhand von Tabellen wird ein rascher Überblick über die Elemente des ACC gegeben und Unterschiede der Register der jeweiligen unterstützten BMS-Prozessoren.

Ein Softwareprojekt und sein Quellcode müssen verwaltet werden, dies wird im Kapitel Deploymentprozess beleuchtet. Es werden u. a. die Versionierung des ACC, die dazu verwendete Software und der Ausrollprozess beschrieben und anhand von Screenshots und eines Prozessbildes dargelegt. Ein Quellcodeschnipsel beschreibt die Möglichkeit, die automatische Versionierung wieder in einem Visual Studio Projekt freizuschalten.

Abschließend finden sich im Kapitel Fazit Zahlen, Daten und Fakten zum ACC betreffend der Wirtschaftlichkeit und Effizienzsteigerung. Diese Zusammenfassung und ein Ausblick, was in Zukunft mit diesem Projekt passieren könnte. Eine Reflexion gibt einen Einblick in den Projektalltag und setzt sich kritisch mit diesem Projekt auseinander.

6.2 Reflexion

Während eines Projektes gibt es unterschiedliche Phasen, so auch bei der Umsetzung des ACC. Der Anfang war sehr positiv, von der Anfrage bis zu den ersten Codezeilen und der anschließenden deutlichen Verbesserung. Zum Ende hin wurde es dann etwas anstrengend, was mehrere Gründe hatte. In diesem Kapitel werden unterschiedliche Themenblöcke innerhalb der Projektzeit reflektiert.

In Summe sind in etwa 240 Stunden an Arbeit angefallen, dies inkludiert sowohl die Treffen mit der Firma AccuPower, die Zeit des Programmierens und Debuggens und die Zeit zum Schreiben der Diplomarbeit. Davon entfallen ca. 80 h Stunden auf das Schreiben der Diplomarbeit und die restlichen Stunden sind direkt in die Weiterentwicklung der Software geflossen. Da ich noch nie eine Arbeit in diesem Umfang geschrieben habe, habe ich keinen Referenzwert, ob dies viel ist oder nicht. Bereits während der Entwicklung der Software sind interessante Softwareprojekte, Artikel und Handbücher mitnotiert worden, damit die Dokumentation bzw. das Verfassen der Diplomarbeit schneller vonstattengeht.

Bei Informatikprojekten ist es gefährlich, einen Pauschalpreis zu nennen und dementsprechend wird auf Stundenbasis verrechnet. Dies kann verglichen werden mit Bauprojekten neben Flüssen oder an anderen unsicheren Orten, wo man den Kräften der Natur ausgesetzt ist. Müssen die Bauträger einen Pauschalpreis nennen, kann dieser meist nicht berappt werden, da eine entsprechend hohe Sicherheit auf Grund von Hochwässern einberechnet werden muss und somit wird die tatsächliche Arbeit verrechnet. Daraus folgend wird auch das ACC auf Stundenbasis verrechnet, wobei hier angemerkt werden muss, dass weder die Dokumentation noch weitere unproduktive Zeiten (Anfahrt, Essen, Kaffeepausen, ...) verrechnet wurden.

Entsprechend der Stundenverrechnungsmethodik ist es natürlich frustrierend, wenn das Projekt nicht in der gewünschten Zeit einen entsprechenden Fortschritt zeigt, auch wenn ohne Probleme belegbar ist, woran es lag. Ein Beispiel hierfür ist der Wunsch, ein neues Multimeter seitens des ACC zu unterstützen. Dieser Featurewunsch war zu Projektbeginn nicht vorhanden. Beim Beginn der Umsetzung war noch nicht klar, dass das Testgerät umständlicher in der Handhabung ist als die bereits eingebundenen und unterstützten Multimeter und das die gekauften Geräte sich nochmals vom Testgerät unterscheiden.

Durch die Stundenverrechnungsmethode wird der Quellcode nicht zur Gänze fein justiert und auf Performance getrimmt, sondern ab einem gewissen Reifegrad einfach ausgeliefert. Dadurch wird ein gewisser technischer Ehrgeiz nicht befriedigt, aber dies ist leider notwendig, um wirtschaftlich zu sein. So könnte der Quellcode bei den genannten neuen Multimetern sicherlich deutlich optimiert werden, aber aufgrund der Zeitverzögerung und der dabei entstehenden Kosten, die nicht unbedingt getragen werden wollen, ist ein funktionierender Code ausreichend.

Ein weiterer Verzögerungspunkt bestand in der unzureichenden Dokumentation seitens der Firma TI. Zu Beginn des Projektes wechselte ein Mitarbeiter seitens der Firma AccuPower, der das größte Know-how bzgl. der eingesetzten TI Prozessoren hat, die Firma und dadurch

musste bei gewissen Themen von null begonnen werden. Die entsprechende Dokumentation, in der auch Fehler enthalten sind, wie dies bereits beschrieben wurde, machten die Sache noch deutlich komplizierter und komplexer und daher musste ein deutlich größerer Aufwand in die Recherche gesteckt werden, als dies zu Beginn des Projektes klar gewesen ist.

Als ehemaliger Mitarbeiter von AccuPower war es für viele alte Kollegen, die an dieser Diplomarbeit indirekt mitwirkten, schwierig zu unterscheiden, dass ich in einer neuen Rolle, eben als externe Firma, auftrete und beispielsweise Besprechungen mitprotokolliert wurden bzgl. der Informationsgewinnung zu neu umzusetzenden Punkten, der Abnahme von programmierten Features und dem Heraushören, was benötigt wird. Hier muss vom sozialen Aspekt früher direkt mitgeteilt werden, dass sich die Geschäftsbeziehung geändert hat, auch wenn es offenkundig ist.

Während des Projektes gab es auch längere Stillstandzeiten von bis zu einem Monat, da nicht ausreichend freie Ressourcen seitens der Firma AccuPower für das Projekt vorhanden waren. Dies besserte sich glücklicherweise zum Schluss hin und dadurch konnte in den letzten Monaten des Projektes auch der größte Fortschritt erzielt werden.

Positiv hervorzuheben sind die freien Entscheidungsmöglichkeiten während der Umsetzung des Projektes, so war der Zeitpunkt der Programmierzeiten frei einzuteilen und die Lösungsmöglichkeiten konnten selbstständig erdacht und umgesetzt werden. Durch diese Freiheit war Selbstdisziplin notwendig, um auch an harten Studien- und Arbeitstagen dennoch das Projekt fortzusetzen und zum Erfolg zu führen.

Das Reporting zur Geschäftsführung und zum Projektleiter von AccuPower war sowohl mittels Microsoft Teams möglich als auch durch physische Besprechungen. Die Besprechungen verliefen alle ausnahmslos zufriedenstellend, Probleme konnten angesprochen werden und Maßnahmen und Lösungen wurden rasch umgesetzt. Die flache Hierarchie und der unkomplizierte Kommunikationsweg machten auch Motivation, das Projekt weiterhin durchzuführen und die Probleme zu meistern.

Die Gespräche mit den Lehrlingen und ausführenden Anwendern des ACC führten auch immer zu positiven Rückmeldungen und weiteren Optimierungsideen, welche die Prozesse beschleunigten, die Arbeit erleichterten und zufriedener gestalteten. Entsprechend konnte auch der Ausrollprozess vereinheitlicht und optimiert werden und dadurch die Vermeidung von ungleich ausgerollten Softwareversionen verhindert werden. Eine schöne Rückmeldung war, dass das Abarbeiten bzw. Aufbereiten der Akkumulatoren mit dem neuen ACC Spaß macht. Durch solche Rückmeldungen war das Bemühen um weitere Fortschritte in der Vereinfachung der Prozesse umso leichter und größer.

Ein weiterer motivierender Faktor waren die gemeinsamen Mittagessen und Kaffeepausen, bei welchen Wehwehchen und Unzulänglichkeiten der Umsetzung meinerseits problemlos angesprochen werden konnten. Die entsprechenden Rückmeldungen wurden ernst genommen und in der nächsten Version berücksichtigt und umgesetzt.

Rückblickend ist man natürlich immer schlauer, aber in Summe bin ich sehr froh, das Projekt mit meinen ehemaligen Arbeitskollegen zum Erfolg geführt zu haben. Durch das ACC konnten unnötige Einzelschritte und das umständliche Wechseln zwischen Dateien aus dem Arbeitsalltag eliminiert werden und dadurch an Arbeitszufriedenheit dazugewonnen werden. Hier lässt sich erkennen, dass Automatisierung auch positive Aspekte für den Arbeitsalltag der Menschen bietet. Würde das Projekt allerdings Neustarten, würde ich folgenden Punkten mehr Bedeutung einräumen, um reibungsloser voranzuschreiten:

1. Klarstellung der neuen Geschäftsbeziehung und daraus resultierende Folgen (Mitloggen von Meetings).

2. Auf die Freistellung von Ressourcen behaaren, damit keine unnötig langen Stillstandzeiten entstehen.

3. Eine Ansprechperson bzgl. Wissen um TI-Prozessoren bekommen, dadurch könnte die Programmierung effizienter vonstattengehen.

Durch die genannten drei Punkte wäre eine reibungslosere Kommunikation ohne Fehlkommunikation möglich, die Stillstandzeiten wären eliminiert und die Recherchezeiten würden drastisch reduziert, dadurch wäre das Projekt deutlich schneller und effizienter umgesetzt worden. Allerdings kann hier nur festgehalten werden, dass trotz aller Widrigkeiten das Projekt sowohl in technischer als auch in wirtschaftlicher und sozialer Hinsicht eindeutig erfolgreich abgeschlossen ist.

Die genannten negativen Aspekte und Thematiken sind meist menschlicher Natur, und entsprechend kann man aus einem anderen Blickwinkel feststellen, dass man daran gewachsen ist und dadurch würde ich sie auch nicht missen wollen, auch wenn sie für Verzögerungen und Reibereien sorgten. Zum Schluss konnte alles zum Guten gewandt werden und menschliche Aspekte sorgen jedenfalls für keine Langeweile im Projektleben.

In einem Bild zusammengefasst entspricht das ACC einem Segelschiff, welches aus einem Sturm in sichere Gewässer gelaufen ist. Auf der Reise wurden manche Flauten passiert und Streitigkeiten beigelegt, um gemeinsam an Verbesserungen zu arbeiten und Reparaturen durchzuführen und dabei machte die gesamte Mannschaft mit. Dadurch konnte das Schiff für weitere Fahrten wieder fit gemacht und durch die Generalüberholung dürfte ein weiterer Sturm auch keine Probleme mehr bereiten.

Wenn man sich weder von den Höhen in irreale Gedanken verläuft und von den negativen Aspekten verschlucken lässt, sondern das Gleichmaß wahrt, dann kann wohl jedes Projekt zum Erfolg geführt werden, klar nach dem Ausspruch „In der Ruhe liegt die Kraft“.

6.3 Ausblick

Dank des großen Potenzials, welches die Automatisierung der technischen Prozesse bietet, wird das ACC voraussichtlich weiterentwickelt werden. Zu den nächsten Schritten gehört weiterhin die Weiterentwicklung der Grundform in der Programmiersprache C#. Dies bedeutet, dass in der nächsten Ausbaustufe der Prozessor auf der BMS-Platine direkt im Speicher beschrieben wird.

Als nächster Schritt empfiehlt es sich, den Akkumulator durch einen Klick auf die Checkbox „Akku gesperrt“, ersichtlich in Abbildung 4.6: ACC Behandlungsart „Aufbereitung prog“, wirklich zu sperren und nicht nur zu überprüfen, ob dieser gesperrt ist und somit für den Versand freigegeben werden kann.

```
byte[] close chip = { 0x55, 0x20 };
write reg addr length = 2;
AardvarkApi.aa i2c write(handle, device,
         AardvarkI2cFlags.AA I2C NO FLAGS,
         (ushort)write reg addr length, close chip);
AardvarkApi.aa sleep ms(200);
```

Quelltext 6.1: C#-Code zum Sperren des Prozessors

Der Quelltext 6.1 zeigt eine Möglichkeit, den bq34z100g1-Prozessor zu sperren. Dazu muss dieser auf seiner Adresse 0x55 und dem Register 0x20 angesprochen werden, anschließend sollte sich dieser in den gesperrten Zustand versetzen und es ist ohne Eingabe des Passworts keine Bearbeitung der Konfiguration und Firmware möglich. Vor Abschluss der Operation muss ein Delay von 200 ms absolviert werden, um einen gültigen Bearbeitungszyklus abzuwarten.

Anschließend sollten der Prozessor der Seriennummer- und der Flashdatumvergabe automatisiert werden. Dies sollte die Checkboxen „Seriennummer [HEX] + Flashdatum geflasht“ und „LT-Reset [Prozessor-Check]“ vereinen und somit einen Schreibvorgang auf den Prozessor auslösen und in die korrekten Register den Little-Endian-Wert übertragen und anschließend auf Korrektheit überprüfen. Analog verhält es sich mit dem Flashdatum, wobei hier auf einen Integerwert nach der entsprechenden Formel gerechnet wird.

Eine weitere lohnenswerte Thematik, die der Automatisierung unterworfen werden sollte, ist das Flashen des Goldenimages. Dazu wird die erstelle Firmware als *.srec-Datei, einer einfachen ASCII kodierten binären Datei abgespeichert und anschließend Zeile für Zeile iteriert, um in die entsprechenden Register die notwendigen Daten zu übertragen [68].

Bevor die Daten transferiert werden können, muss der Prozessor in den ROM-Mode gebracht werden, um Daten entsprechend schreibend übertragen zu können. Sobald dies geschehen ist, muss die alte Firmware im Speicher des Prozessors überschrieben werden, um fehlerhafte Einträge ausschließen zu können und damit anschließend die korrekte Übertragung der neuen Daten aus der *.srec-Datei in den Speicher des Prozessors sichergestellt ist. Im letzten Schritt muss überprüft werden, ob Errors bei der Übertragung aufgetreten sind und ein Neustart des Vorganges erforderlich ist.

Eine Möglichkeit, den genannten Prozess in Quellcode umzusetzen, findet sich nachfolgend:

```
    '// MASS ERASE DATA FLASH
     lError = WriteI2CByte(&H0, &HC, &H16)
     lError = WriteI2CByte(&H4, &H83, &H16)
     lError = WriteI2CByte(&H5, &HDE, &H16)
     iChecksum = (&HC + &H83 + &HDE) Mod &H10000
     lError = WriteI2CByte(&H64, iChecksum Mod &H100, &H16)
     lError = WriteI2CByte(&H65, iChecksum \ 256, &H16)
     DoDelay 0.5

     '// WRITE EACH ROW
     For iRow = 0 To iNumberOfRows - 1
         lError = WriteI2CByte(&H0, &HA, &H16) '//program row command

         '// WRITE TARGET ROW TO THE ROW LOW REGISTER
         lError = WriteI2CByte(&H1, iRow, &H16)
         iChecksum = (&HA + iRow) Mod &H10000

         '// COPY DATA FROM THE FULL ARRAY TO THE ROW ARRAY
         For iIndex = 0 To 31
         yRowData(iIndex) = yDataFlashImage((iRow * 32) + iIndex)
         iChecksum = (iChecksum + yRowData(iIndex)) Mod &H10000
         Next iIndex

         '// WRITE THE ROW DAYA REGISTERS
         lError = WriteI2CByteArray(&H4, yRowData, 32, &H16)

         '// WRITE THE ROW
         lError = WriteI2CByte(&H64, iChecksum Mod &H100, &H16)
         lError = WriteI2CByte(&H65, iChecksum \ 256, &H16)
         DoEvents '//allow Windows to catch up
         DoDelay 0.2
     Next iRow
     End With

      '// EXECUTE GAS GAUGE PROGRAM
     lError = WriteI2CByte(0, &HF, &H16)
     lError = WriteI2CByte(&H64, &HF, &H16)
     lError = WriteI2CByte(&H65, 0, &H16)

     '// RETURN OK OR ERROR
     Write DFI to Flash = 0

End Function
```

Quelltext 6.2: VB-Code zum Löschen der alten Firmware und zum Übertragen der neuen Firmware

In Quelltext 6.2 wird dieser Vorgang als Visual Basic Code aus einem alten Handbuch von TI abgebildet. Dieser VB-Code muss entsprechend in C# übersetzt und für alle Prozessoren angepasst werden. Im ersten Schritt wird der Speicher des Prozessors geleert und anschließend jede Zeile aus der *.srec-Datei korrekt an den Speicher übergeben [vgl. 69, S. 7].

Die letzte Quintessenz der Automatisierung liegt im Vorgang der Strom- und Spannungskalibrierung. Dazu ist es notwendig, analog zum Vorgang der Seriennummerübertragung, diese für die entsprechenden Strom- und Spannungswerte zu wiederholen. Neben den Anleitungen von TI existieren diverse Bibliotheken auf Github, welche als Vorlage dienen können [70].

Nach der Absolvierung der oben genannten Schritte können nur noch Gewinne in Sachen der Stabilität und Performance erzielt werden, aber die Prozesse sind bereits vollkommen digitalisiert und automatisiert.

Eine weitere Zukunftsperspektive spiegelt sich in der Neuprogrammierung des ACC in der Programmiersprache Python. Dies hätte den Vorteil, dass das ACC direkt in das ERP-System Odoo integriert werden könnte. Dadurch wären weitere kaufmännische Prozesse abbildbar und eine Cloudfunktionalität wäre gegeben.

Etwas genauer beschrieben wäre es möglich, das Fertigungsmodul von Odoo entsprechend um eine IoT-Komponente zu erweitern und dadurch die Prozesse des ACC abzubilden und zwar unabhängig vom Betriebssystem und dem Standort. Mit entsprechender Programmierung wäre es möglich, gänzlich ungeschultes Personal den Umgang mit entsprechenden einzelnen Teilschritten zu lehren und diese durchführen zu lassen.

Durch die genannte Cloudfunktionalität wäre es möglich, direkt beim Kunden den Akkumulator mit der neuesten und aktuellsten Firmware zu beschreiben, ohne den Versandweg beschreiten zu müssen. Entsprechend geringer würde die Umwelt belastet und Akkumulatoren würden langlebiger werden, durch die Optimierung für den jeweiligen Einsatzzweck.

Des Weiteren wäre die Etablierung eines neuen Geschäftsmodelles möglich, anstatt den Akkumulator zu verkaufen, könnte dieser vermietet werden. Dies bedeutet, die Firma AccuPower hätte direkt Einblick in das Lager des Kunden, sieht, welche Akkumulatoren demnächst getauscht werden müssen und sorgt stetig für den entsprechenden Nachschub und die vollständige Einsatzbereitschaft der Akkumulatorenflotte. Der Kunde müsste sich weder um den Kauf, um die Wartung noch um die Entsorgung der Akkupacks kümmern, sondern mietet einfach eine entsprechende Leistungsklasse an und AccuPower kümmert sich um die Umsetzung.

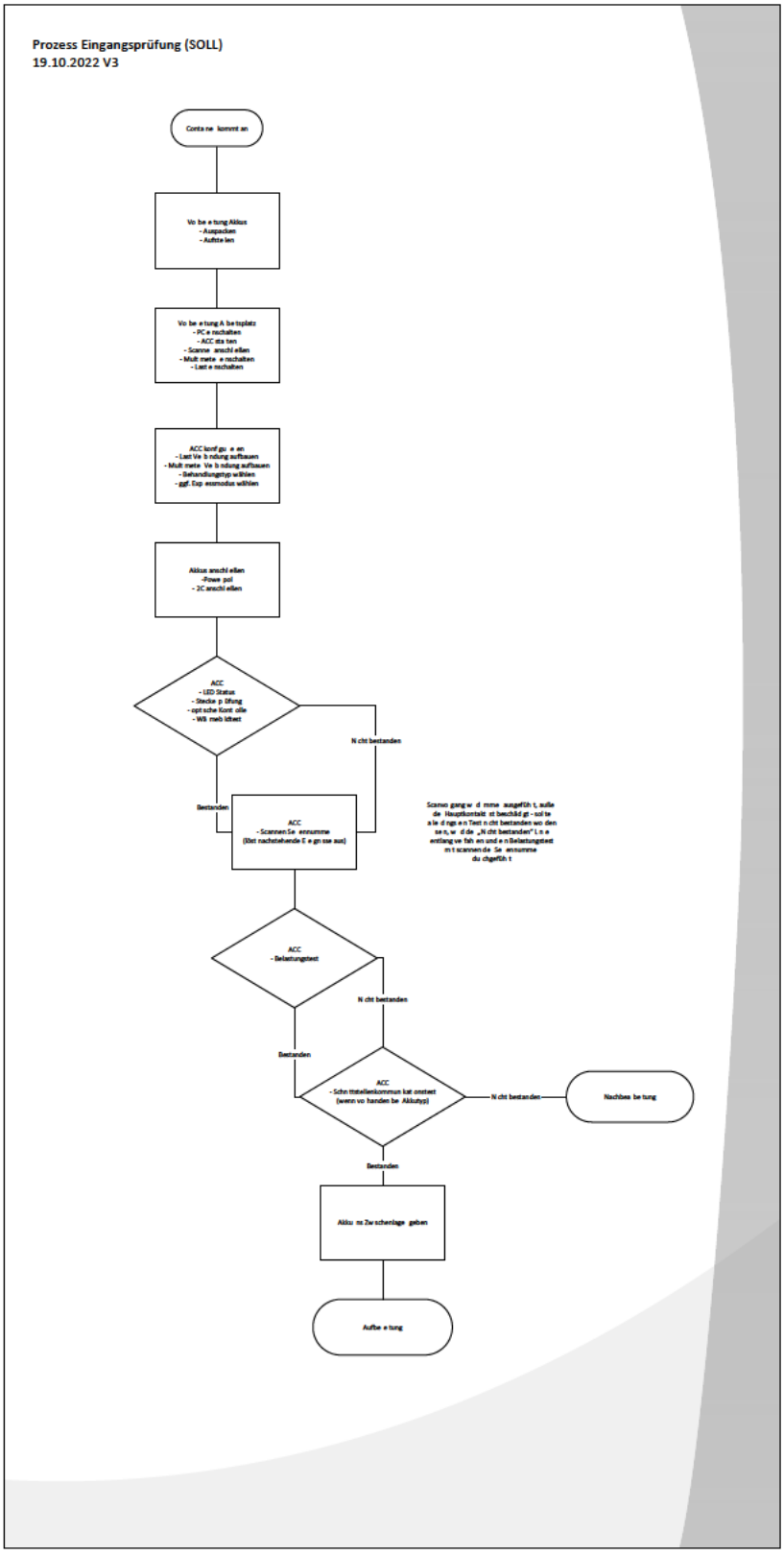
Das genannte Geschäftsmodell würde für einen kontinuierlichen Cashflow sorgen und die Kundenbindung an AccuPower deutlich erhöhen. Vorbild sind hierfür sogenannte „Pay-per-use“-Modelle wie dies beispielsweise die britische Firma Rolls Royce bereits umsetzt [71].

Dazu ist seitens von AccuPower eine Investition in die IT, OT und Entwicklung notwendig. Das einzuführende ERP-System Odoo würde um entsprechende Module, die aus der eigenen Entwicklung entstehen, erweitert. Und durch IoT-Hardware, die direkt beim Kunden im Lager im Einsatz ist, wäre es möglich, durch drahtlose Kommunikationswege kontinuierlich von den Akkumulatoren Daten zu beziehen und durch Algorithmen auswerten zulassen, wie es um die aktuellen Lagerbestände bestellt ist. Entsprechend muss ein erhöhtes Sicherheitsbewusstsein bei der Programmierung der benötigten Module zutage treten, um entsprechende Imageschäden aus Hackerangriffen zu vermeiden.

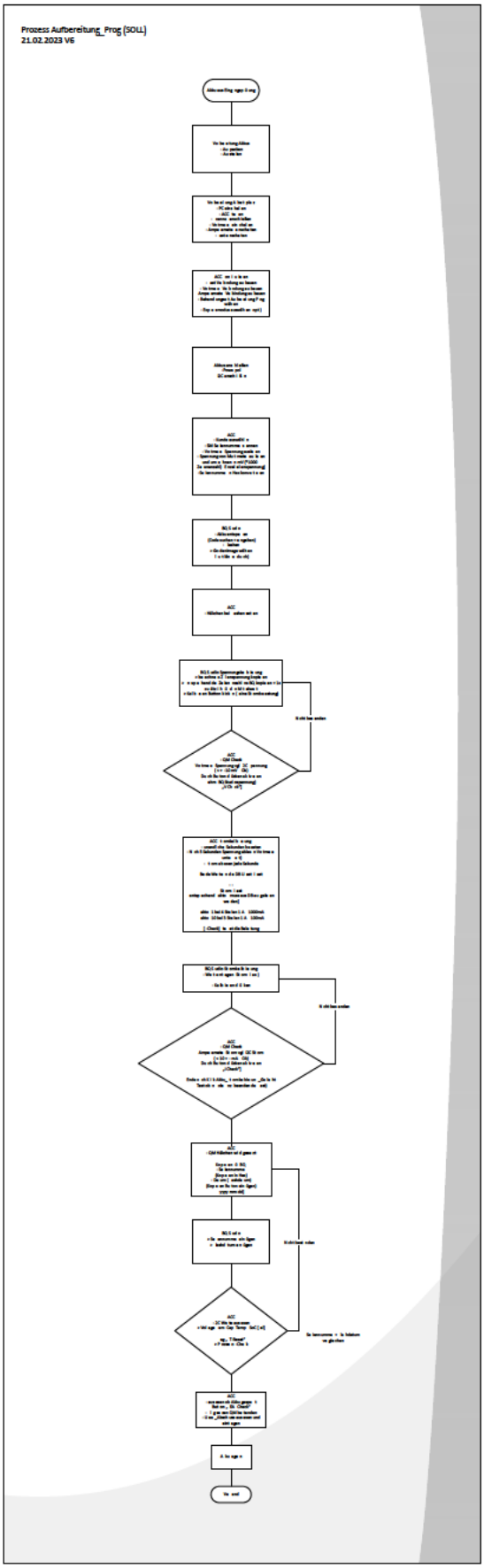
In die Zukunft blickend brechen voraussichtlich gute Zeiten für mobile Akkumulatoren an.

Anhang A: Prozessleitbilder

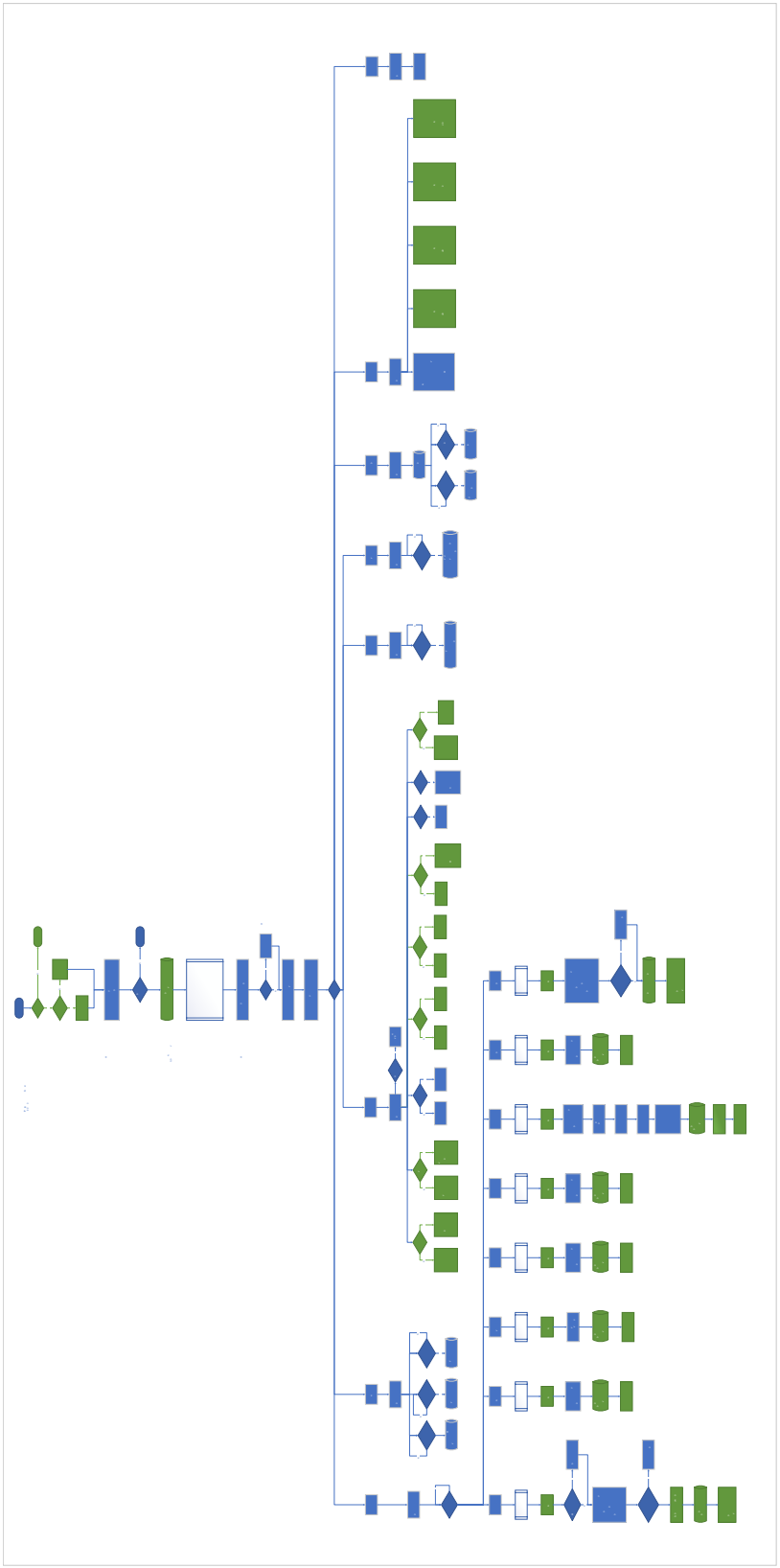
A.1 Prozessleitbild Eingangsprüfung



A.2 Prozessleitbild Aufbereitung Programmierung



A.3 ACC Programmatik



1

[1] Dieses Flussbild kann im digitalen Format angesehen werden.

Literaturverzeichnis

[1] M. Geyer, *Belegarbeit Datenbankaufbau AccuControlCenter*, Vorlesung Datenbanken in der AT, Bernhard Weinhappel, 2023.

[2] SFG. „Junger Steirer kauft AccuPower GmbH“. (2022), Adresse: https://www.sfg.at/n/junger-steirer-kauft-accupower-gmbh/ (besucht am 14. 12. 2022).

[3] AccuPower. „AT ACCUPOWER - Der Experte für mobile Energie“. (2023), Adresse: https://www.accupower.at (besucht am 19. 05. 2023).

[4] M. Geyer, *Belegarbeit Projektplanung AccuControlCenter*, Vorlesung Projektmanagement, Manfred Lach, 2023.

[5] Webgeyer. „Webgeyer“. (2023), Adresse: https://webgeyer.com/ (besucht am 19. 05. 2023).

[6] M. Minarik, T. Renjo und P. Otipka, *Bestandsaufnahme AccuControlCenter - Ausblick und Planung*, Meeting, AccuPower, 13.09.2022.

[7] Microsoft. „Windows Presentation Foundation, WPF, die .NET-Plattform und Visual Studio ermöglichen das Entwickeln moderner datenorientierter Geschäftsanwendungen.“ (2023), Adresse: https://visualstudio.microsoft.com/de/vs/features/wpf/ (besucht am 05. 03. 2023).

[8] J. Allen. „A WPF Q&A“. (2014), Adresse: https://www.infoq.com/news/2014/04/WPF-QA/ (besucht am 05. 03. 2023).

[9] E. Dietrich. „C# Version History: Examining the Language Past and Present“. (2017), Adresse: https://blog.ndepend.com/c-versions-look-language-history/ (besucht am 12. 04. 2023).

[10] Microsoft. „Die Geschichte von C#“. (2023), Adresse: https://learn.microsoft.com/de-de/dotnet/csharp/whats-new/csharp-version-history (besucht am 12. 04. 2023).

[11] AnshulAggarwal. (2023), Adresse: https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/ (besucht am 22. 05. 2023).

[12] Lukas. „Microsoft SQL Server – die Geschichte des Datenbankverwaltungssystems“. (2022), Adresse: https://best-software.de/blog/microsoft-sql-server-die-geschichte-des-datenbankverwaltungssystems/ (besucht am 12. 04. 2023).

[13] M. Schwartz und N. Abramson. „The Alohanet - surfing for wireless data [History of Communications]“. (2009), Adresse: https://ieeexplore.ieee.org/document/5350363 (besucht am 13. 04. 2023).

[14] Ionos. „Was ist Ethernet?“ (2022), Adresse: https://www.ionos.de/digitalguide/server/knowhow/ethernet-ieee-8023/ (besucht am 13. 04. 2023).

[15] P. Schnabel. „OSI-Schichtenmodell“. (2003), Adresse: https://www.elektronik-kompendium.de/sites/kom/0301201.htm (besucht am 13. 04. 2023).

[16] Dewetron. „SCPI - Die universelle Schnittstelle“. (2022), Adresse: https://www.dewetron.com/de/2022/09/scpi-die-universelle-schnittstelle/ (besucht am 13. 04. 2023).

[17] Keysight. „Einführung in die SCPI-Sprache“. (2015), Adresse: https://rfmw.em.keysight.com/spdhelpfiles/33500/webhelp/DE/Content/ I SCPI/00%20scpi introduction.htm (besucht am 13. 04. 2023).

[18] GWInstek. „GDM-906X Series User Manual“. (2023), Adresse: https://www.gwinstek.com/en-global/products/downloadSeriesDownNew/14348/1772 (besucht am 13. 04. 2023).

[19] NXP Semiconductors. „UM10204, I2C-bus specification and user manual“. (2023), Adresse: https://community.nxp.com/pwmxy87654/attachments/pwmxy87654/nxp-designs/931/1/UM10204.pdf (besucht am 15. 04. 2023).

[20] E. Buchmann. „I2C-Ansteuerung“. (2020), Adresse: https://www.ipd.kit.edu/mitarbeiter/buchmann/microcontroller/i2c.htm (besucht am 15. 04. 2023).

[21] SMIF. „SMBus“. (2013), Adresse: http://smbus.org/ (besucht am 16. 04. 2023).

[22] SMI. „System Management Bus (SMBus) Specification“. (2022), Adresse: http://smbus.org/specs/ (besucht am 16. 04. 2023).

[23] AccuPower. „ACCUPOWER Lithium Akku 4S1P 14,4V 3,35Ah 48,24Wh“. (2023), Adresse: https://www.accupower.at/produkt/accupower-lithium-akku-4s1p-144v-335ah-4824wh/ (besucht am 19. 05. 2023).

[24] Odoo. „Odoo: The New OpenERP, Moving into new territories, beyond ERP.“ (2023), Adresse: https://www.odoo.com/de DE/blog/odoo-news-5/odoo-the-new-openerp-156 (besucht am 22. 04. 2023).

[25] Odoo. „Hosting Types“. (2023), Adresse: https://www.odoo.com/de DE/page/hosting-types (besucht am 22. 04. 2023).

[26] debian. „debian, The Universal operating system“. (1997), Adresse: https://www.debian.org/ (besucht am 22. 04. 2023).

[27] docker. „Develop faster. Run anywhere. The most-loved Tool in Stack Overflow’s 2022 Developer Survey.“ (2004), Adresse: https://www.docker.com/ (besucht am 22. 04. 2023).

[28] nginx. „nginx“. (2007), Adresse: https://nginx.org/ (besucht am 22. 04. 2023).

[29] Odoo. „Agiles Projektmanagement, Schön. Einfach. Open Source.“ (2021), Adresse: https://www.odoo.com/de DE/app/project (besucht am 22. 04. 2023).

[30] AccuPower. „ACCUPOWER LiFePO4 Akku 8S2P 25,6V 8Ah mit LED Ladestandanzeige“. (2023), Adresse: https://www.accupower.at/produkt/accupower-lifepo4-akku-8s2p-256v-8ah-mit-led-ladestandanzeige/ (besucht am 22. 04. 2023).

[31] TI. „EV2400, USB-Based PC Interface Board for Battery Fuel (Gas) Gauge Evaluation Module“. (2012), Adresse: https://www.ti.com/tool/EV2400 (besucht am 22. 04. 2023).

[32] ICONS8. „Free Icons, Clipart Illustrations, Photos, and Music“. (2023), Adresse: https://icons8.com/ (besucht am 21. 05. 2023).

[33] flaticon. „Vector Icons and Stickers - PNG, SVG, EPS, PSD and CSS“. (2023), Adresse: https://www.flaticon.com/ (besucht am 21. 05. 2023).

[34] D. Gradwohl und AccuPower. „Transport von wiederaufladbaren Lithium-Ionen Batterien* Zellen*“. (2020), Adresse: https://www.accupower.de/wp-content/uploads/Informations/TRANSPORT VON UN3480 LIHTIUMIONEN BATTERIEN ADR RID IMDG IATA-ICAO DE.pdf (besucht am 27. 04. 2023).

[35] D. Engel, MashaMSFT, rothja, cheenamalhotra und yukiwongky. „Microsoft ADO.NET für SQL Server and Azure SQL-Datenbank“. (2023), Adresse: https://learn.microsoft.com/de-de/sql/connect/ado-net/microsoft-ado-net-sql-server?view=sql-server-ver16 (besucht am 29. 04. 2023).

[36] M. Mierke. „DHCP - was ist das?“ (2022), Adresse: https://www.heise.de/tipps-tricks/DHCP-was-ist-das-4476566.html (besucht am 30. 04. 2023).

[37] S. Luber und A. Donner. „Was ist Telnet?“ (2018), Adresse: https://www.ip-insider.de/was-ist-telnet-a-691219/ (besucht am 30. 04. 2023).

[38] Keysight. „SCPI Programming Reference“. (2019), Adresse: https://rfmw.em.keysight.com/bihelpfiles/Truevolt/WebHelp-Mobile/US/Advanced/Resources/Toc10.htm (besucht am 30. 04. 2023).

[39] GWInstek. „Dual Measurement Multimeter, GDM-9060/9061“. (2023), Adresse: https://www.gwinstek.com/en-global/products/downloadSeriesDownNew/14348/1772 (besucht am 30. 04. 2023).

[40] EA. „Programmieranleitung ModBus & SCPI, Für USB, GPIB, Ethernet und AnyBus-Module“. (2023), Adresse: https://elektroautomatik.com/shop/de/service/support-material/programmieranleitung-modbus/ (besucht am 30. 04. 2023).

[41] C. Möhring. „ASCII-Code und ASCII-Tabellen“. (2019), Adresse: https://www.heise.de/tipps-tricks/ASCII-Code-und-ASCII-Tabellen-4558152.html (besucht am 30. 04. 2023).

[42] ahnbizcad. „Difference between CR LF, LF and CR line break types“. (2023), Adresse: https://stackoverflow.com/questions/1552749/difference-between-cr-lf-lf-and-cr-line-break-types (besucht am 30. 04. 2023).

[43] Total Phase. „AARDVARK I2C/SPI HOST ADAPTER“. (2014), Adresse: https://www.totalphase.com/products/aardvark-i2cspi/ (besucht am 01. 05. 2023).

[44] Total Phase. „AARDVARK SOFTWARE API“. (2014), Adresse: https://www.totalphase.com/products/aardvark-software-api/ (besucht am 01. 05. 2023).

[45] Total Phase. „Aardvark I2C/SPI Host Adapter User Manual“. (2014), Adresse: https://www.totalphase.com/support/articles/200468316/ (besucht am 01. 05. 2023).

[46] batronix. „Rigol DS1054Z“. (2023), Adresse: https://www.batronix.com/versand/oszilloskope/Rigol-DS1054Z.html (besucht am 01. 05. 2023).

[47] lisong26. „Mini Saleae 16 Logic Analyzer USB 100M Max Probenrate Unterstützung 1.2.10 Software“. (2023), Adresse: https://www.ebay.at/itm/134462237586?hash=item1f4e92eb92:g:jy4AAOSweANgNL6K&amdata=enc%3AAQAIAAAA8IwMSzSTn3NxzL2DrTuwVODorHY9hmu3mlwnlDuuvjnJzBYarNUk2a6cx22iaawBlNcx2cSOe15UCTvmyQ2POpW0%2F7RIpKh5E6BvAL30wwaweKyxVFKLWAtawYv2uHTu238g%2F8AevhRWWn56Ns51lnmEBU0WakwB3rjCibCSf6c5IJ4uMC8bm1gjNSrO5ZdSQ%2BWPGrDeNGshKa91F2A%2BAMqP%2B%2B92MUw0VqdVNhNn84V8K5ZtNYw4jFaBZG3c8LfNxvRFZgk2xWj9kakjpmZpajJOfEDSDr6JUgw78ohB5iawI3taSKbPiYAxrf6LxztOjA%3D%3D%7Ctkp%3ABk9SR4Ca1Pz6YQ (besucht am 01. 05. 2023).

[48] saleae. „Debug happy“. (2008), Adresse: https://www.saleae.com/downloads/ (besucht am 01. 05. 2023).

[49] Total Phase. „7-bit, 8-bit, and 10-bit I2C Slave Addressing“. (2014), Adresse: https://www.totalphase.com/support/articles/200349176-7-bit-8-bit-and-10-bit-I2C-Slave-Addressing/ (besucht am 17. 05. 2023).

[50] TI. „BQ40Z80, Technical Reference Manual“. (2022), Adresse: https://www.ti.com/lit/ug/sluubt5c/sluubt5c.pdf?ts=1682962264079&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ40Z80 (besucht am 01. 05. 2023).

[51] TI. „bq40z50-R1, Technical Reference“. (2018), Adresse: https://www.ti.com/lit/ug/sluubc1d/sluubc1d.pdf?ts=1682962560940&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ40Z50-R1 (besucht am 01. 05. 2023).

[52] TI. „bq40z50-R2, Technical Reference“. (2018), Adresse: https://www.ti.com/lit/ug/sluubk0b/sluubk0b.pdf?ts=1684302196250&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ40Z50-R2%253Futm_source%253Dgoogle%2526utm_medium%253Dcpc%2526utm_campaign%253Dapp-null-null-gpn_en-cpc-pf-google-soas%2526utm_content%253Dbq40z50-r2%2526ds_k%253DBQ40Z50-R2%2526dcm%253Dyes%2526gclid%253DEAIaIQobChMIru3glNL7_gIVSFJ9Ch2s6Q83EAAYASAAEgKGKvD_BwE%2526gclsrc%253Daw.ds (besucht am 17. 05. 2023).

[53] TI. „bq78350 CEDV Li-Ion Gas Gauge and Battery Management Controller Companion to the bq769x0 Battery Monitoring AFE“. (2012), Adresse: https://www.ti.com/lit/ds/symlink/bq78350.pdf?ts=1682962790450&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ78350 (besucht am 01. 05. 2023).

[54] TI. „BQ78350-R1, Technical Reference“. (2020), Adresse: https://www.ti.com/lit/ug/sluubd3e/sluubd3e.pdf?ts=1682962809930&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ78350-R1 (besucht am 01. 05. 2023).

[55] TI. „bq78350-R1 TRM Addendum for the bq78350-R2 Device“. (2018), Adresse: https://www.ti.com/lit/ug/sluubm6/sluubm6.pdf?ts=1682962816897&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FBQ78350-R2-DEVICE-FW (besucht am 01. 05. 2023).

[56] TI. „BQ34Z100-G1 Wide Range Fuel Gauge with Impedance Track™ Technology“. (2021), Adresse: https://www.ti.com/lit/ds/symlink/bq34z100-g1.pdf?ts=1682963053886&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ34Z100-G1 (besucht am 01. 05. 2023).

[57] T. Cosby. „BQ34Z100 Manufacturer Date format“. (2013), Adresse: https://e2e.ti.com/support/power-management-group/power-management/f/power-management-forum/283058/bq34z100-manufacturer-date-format (besucht am 02. 05. 2023).

[58] TI. „Using I2C Communications With the bq34110 bq35100 and bq34z100-G1 Series of Gas“. (2023), Adresse: https://www.ti.com/lit/an/slua790/slua790.pdf?ts=1682943266303&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ34Z100-G1 (besucht am 02. 05. 2023).

[59] Egoloizos. „Was ist Deployment?, Definition Software-Verteilung“. (2021), Adresse: https://www.dev-insider.de/was-ist-deployment-a-0463e840e8b2b2372b3ba71865b875ca/ (besucht am 05. 05. 2023).

[60] S. Chacon und B. Straub. „1.2 Getting Started - A Short History of Git“. (2014), Adresse: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git (besucht am 06. 05. 2023).

[61] S. Chacon und B. Straub. „4.8 Git on the Server - GitLab“. (2014), Adresse: https://git-scm.com/book/en/v2/Git-on-the-Server-GitLab (besucht am 06. 05. 2023).

[62] GitLab. „GitLab Runner“. (2016), Adresse: https://docs.gitlab.com/runner/ (besucht am 06. 05. 2023).

[63] J. Vincent. „The lawsuit that could rewrite the rules of AI copyright“. (2022), Adresse: https://www.theverge.com/2022/11/8/23446821/microsoft-openai-github-copilot-class-action-lawsuit-ai-copyright-violation-training-data (besucht am 06. 05. 2023).

[64] GitHub. „GitHub Desktop“. (2015), Adresse: https://desktop.github.com/ (besucht am 06. 05. 2023).

[65] Microsoft. „MICROSOFT SOFTWARE LICENSE TERMS, MICROSOFT VISUAL STUDIO CODE“. (2015), Adresse: https://code.visualstudio.com/license (besucht am 06. 05. 2023).

[66] Schkn. „How To Add and Update Git Submodules“. (2019), Adresse: https://devconnected.com/how-to-add-and-update-git-submodules/ (besucht am 07. 05. 2023).

[67] M. van der Heijden. „The specified version string contains wildcards, which are not compatible with determinism“. (2018), Adresse: https://marinovdh.wordpress.com/2018/10/22/68/ (besucht am 07. 05. 2023).

[68] Old Unix manual. „srec (5) - Linux Manuals, srec: Motorola S-record record and file format“. (2023), Adresse: https://www.systutorials.com/docs/linux/man/5-srec/ (besucht am 12. 05. 2023).

[69] W. Doug. „Going to Production With the bq34z1 xx“. (2012), Adresse: https://www.ti.com/lit/an/slua665/slua665.pdf (besucht am 12. 05. 2023).

[70] Ralim. „BQ34Z100“. (2018), Adresse: https://github.com/Ralim/BQ34Z100 (besucht am 12. 05. 2023).

[71] Future Markets Magazine. „Was sind Pay-per-use-Modelle?“ (2021), Adresse: https://future-markets-magazine.com/de/markets-technology/was-sind-pay-per-use-modelle/ (besucht am 13. 05. 2023).

Eidesstattliche Erklärung

Hiermit versichere ich – Maximian Geyer – an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt oder anderweitig veröffentlicht.

Gutenberg, 22.05.2023

Ort, Datum