HAW HAMBURG

Bachelorarbeit

Jorge Ignacio Lozano Zehl

Entwicklung eines Vorschaltgeräts zur schonenden Ladung von Smartphones

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Computer Science and Engineering
Department of Information and Electrical Engineering

Jorge Ignacio Lozano Zehl

Entwicklung eines Vorschaltgeräts zur schonenden Ladung von Smartphones

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Elektro- und Informationstechnik*
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Robert Heß
Zweitgutachter: Prof. Dr. Ralf Wendel

Eingereicht am: 25.12.2020

Jorge Ignacio Lozano Zehl

Thema der Arbeit

Entwicklung eines Vorschaltgeräts zur schonenden Ladung von Smartphones

Stichworte

Software Engineering, Hardwareentwicklung, Mikrocontroller, Lithium-Akkumulatoren

Kurzzusammenfassung

Diese Bachelorarbeit bildet die zweite Hälfte eines Projekts zur Entwicklung eines Vorschaltgeräts für Ladegeräte von Smartphones. Um die Lebensdauer von in Smartphones verbauten Lithium-Ionen-Akkumulatoren zu verlängern, kann u.a. der Ladezustand begrenzt werden. Dies soll mithilfe des Vorschaltgeräts erreicht werden. Im ersten Teil des Projekts wurde eine Smartphone-Applikation erstellt. In dieser Arbeit wurde ein Vorschaltgerät entwickelt, das zwischen Ladegerät und Smartphone angeschlossen wird und über Bluetooth mit der App kommuniziert.

Jorge Ignacio Lozano Zehl

Title of Thesis

Development of an intermediate device to gently charge smartphones

Keywords

Software Engineering, Hardware development, microcontroller, lithium batteries

Abstract

This bachelor thesis forms the second half of a project to develop a device to turn on and off the charging process of smartphones. In order to extend the service life of lithium-ion batteries installed in smartphones, the state of charge can be limited. This is to be achieved with this device. In the first part of the project, a smartphone application was created. In this work, the device was developed. It is connected between the charger and the smartphone and communicates with the app via Bluetooth.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Problemstellung

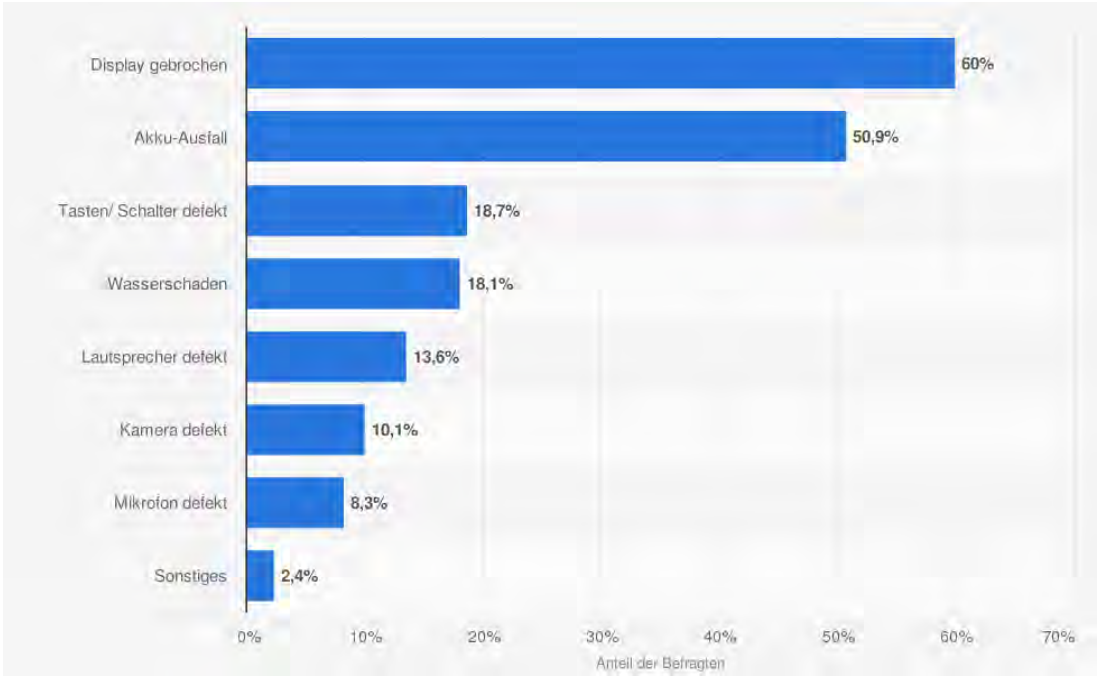
Smartphones sind aus unserem Alltag nicht mehr wegzudenken. Allerdings entstehen durch die stark angestiegene Verwendung mobiler Endgeräte auch Probleme:

Allein im Jahr 2012 wurden 1,75 Milliarden Handys verkauft. Das ergibt eine beträchtliche Summe an natürlichen Ressourcen, die für die Produktion verwendet wurden. Durch den Produktionsprozess und Materialtransporte entstehen weitere Umweltbelastungen. [23]

Handys begleiten uns den ganzen Tag und zum Teil auch nachts und müssen daher ständig einsatzbereit sein. Daher ist es nicht verwunderlich, dass die meisten Besitzer von Smartphones sehr viel Wert auf lange Akkulaufzeiten legen, wenn sie ein neues Smartphone kaufen. [1]

Das Problem bei vielen Smartphones ist jedoch die geringe durchschnittliche Lebensdauer des Akkus. Wie aus der folgenden Statistik hervorgeht, haben 50% der 375 Befragten in Deutschland schon einen Akku-Ausfall erlebt. [33]

Abb. 1.1: Arten von Schäden an Smartphones in Deutschland [33]



Das macht deutlich, dass die meisten Handys eine Schwachstelle beim Akku besitzen. Dementsprechend ist es wichtig, eine Lösung für dieses Problem zu finden. Gleichzeitig wäre diese Lösung auch eine Gelegenheit, unseren Umgang mit Smartphones nachhaltiger zu gestalten. Denn die Verlängerung der Lebensdauer eines Smartphone-Akkus würde mit einer Verlängerung der Nutzungsdauer der Geräte einhergehen und damit zum nachhaltigeren Umgang mit Ressourcen beitragen.

Die Lebensdauer eines Smartphone-Akkus kann durch die richtige Nutzung verbessert werden. Dazu gehört eine Begrenzung der maximalen Ladung [7].

Daher besteht die Herausforderung darin, den Ladevorgang von Smartphone-Akkus so zu beeinflussen, dass der Benutzer in der Lage ist, die maximale Ladung zu begrenzen. Dazu soll im Rahmen dieser Arbeit an einer funktionsfähigen Lösung gearbeitet werden.

1.2 Zielsetzung

Um das Smartphone nicht mechanisch/elektrisch zu beeinflussen, soll ein externes Vorschaltgerät gepaart mit einer geeigneten Applikation entwickelt werden. Die Kommunikation zwischen Applikation und Endgerät soll über Bluetooth stattfinden. Das Vorschaltgerät bildet eine Ergänzung zu dem jeweils vorhandenen Ladegerät des Smartphones und wird an der Niedrigstromseite an dieses angeschlossen.

Aufgrund des Umfangs wurde das Projekt in zwei Teile aufgeteilt: 1) Erstellung einer Applikation und 2) Entwicklung eines Vorschaltgeräts. Das Ziel dieser Arbeit ist die Entwicklung des Vorschaltgeräts auf der Grundlage der von Emrah Demir im Rahmen seiner Bachelorarbeit entwickelten Smartphone-Applikation .

1.3 Aufbau der Arbeit

Im folgenden zweiten Kapitel wird zunächst auf die theoretischen Grundlagen zu Lithium-Ionen-Akkus und deren Alterungsprozess eingegangen sowie die theoretische Seite von Smartphone-Applikationen erläutert. Anschließend wird in Kapitel 3 eine Anforderungsanalyse für das zu entwickelnde Vorschaltgerät durchgeführt, auf deren Grundlage dessen Design und Entwurf entwickelt wurden. Nach deren Darstellung in Kapitel 4 wird in Kapitel 5 die Umsetzung beschrieben. Nach der Evaluation des Projekts in Kapitel 6 wird in Kapitel 7 ein Fazit gezogen und ein Ausblick gegeben.

2 Grundlagen

Dieses Kapitel gibt eine Erklärung der Grundlagen, die zum Verständnis dieser Arbeit erforderlich sind.

Da in Smartphones i.d.R. Lithium-Ionen-Akkumulatoren verbaut werden, werden in diesem Kapitel als erstes die Funktionsweise und die Vor- und Nachteile dieser Energiespeicher erläutert sowie die Faktoren, die beim Alterungsprozess eine wichtige Rolle spielen.

Anschließend wird die Smartphone-Applikation, die den ersten und bereits abgeschlossenen Teil dieses Projektes bildet, erläutert. In diesem Abschnitt wird beschrieben, welche Eigenschaften diese besitzt, welche Einschränkungen sie hat und wie der Kommunikationsprozess abläuft.

2.1 Lithium-Akkumulatoren

Lithium-Akkumulatoren (im Folgenden auch Lithium-Akkus genannt) sind die im Bereich der mobilen Endgeräte meist eingesetzten Akkumulatoren. Dies ist auf mehrere Vorteile zurückzuführen, die diese bieten. In der folgenden Tabelle werden einige Vor- und Nachteile beschrieben. [18]

Tab. 2.1: Charakteristische Eigenschaften von Lithium-Akkus [17]

Vorteile	Nachteile
• Hohe Zellspannung	• Degradation bei hohen Temperaturen
• Schnelles Laden	• Strenge Transportbedingungen
• Über 1000 Lade-Entlade-Zyklen	• Brandgefahr und Kapazitätsverlust im Falle einer Überladung
• Geringe Selbstentladung	• Der Temperaturbereich liegt zwischen −20 und 60 Grad Celsius
• Leicht, klein und kompakt	
• Der Memory-Effekt ist zu vernachlässigen	

Eine entscheidende Eigenschaft von solchen Akkumulatoren ist, dass sie keinen ausgeprägten Memory-Effekt haben. Der Memory-Effekt reduziert die Kapazität und tritt bei einer wiederholten unvollständigen Ladung auf. Der Akku passt sich an den Energiebedarf an und liefert dementsprechend weniger Spannung. Da dieser Effekt bei Lithium-Akkus zu vernachlässigen ist, entsteht die Möglichkeit, diese teilweise zu laden, ohne dass ihre Leistungsfähigkeit in relevantem Maße verringert wird. [17]

Dennoch unterliegt ein Lithium-Akku Alterungsprozessen. Im folgenden Abschnitt wird der Alterungsprozess von Lithium-Zellen erläutert, um die Bedeutung dieses Projekts zu verdeutlichen.

2.2 Alterungsprozess von Lithium-Akkumulatoren

Lithium-Zellen verlieren mit der Zeit und im Gebrauch an Leistung und Kapazität. Während die Gesamtkapazität verringert wird, nimmt der innere Widerstand zu. In den meisten Fällen nimmt auch die Selbstentladung zu. [16]

Im Laufe der Lebensdauer einer Lithium-Zelle entstehen irreversible Nebenreaktionen. Dies wird als kalendarische Alterung bezeichnet. Dieser Alterungsprozess begrenzt die kalendarische Lebensdauer der Zelle. Er ist bestimmt von den verwendeten Materialien und den Lagerbedingungen sowie insbesondere der Temperatur und der Spannung. Für die Zellen, die zum Beispiel in Smartphones und Laptops verwendet werden, liegt die kalendarische Lebensdauer ungefähr bei 5 Jahren. [16]

Ein weiterer Faktor, der zu berücksichtigen ist, wenn über den Alterungsprozess gesprochen wird, ist die Zyklenalterung. Diese wird durch die Lade-Entlade-Vorgänge verursacht und führt zu einer Verringerung der Kapazität. Diese Kapazitätsverluste werden durch die Entstehung von Deckschichten erzeugt. Ein Beispiel für Deckschichten ist das Solid Electrolyte Interface (SEI). Dieses wird in der negativen Elektrode durch die Unterschreitung des in etwa $1\,V$ Potentials gegen Li /Li^+ gebildet. Es ist zu beachten, dass die Bildung dieser Schicht durch hohe Ströme und hohe Temperaturen beschleunigt wird. Diese Art von Deckschichten verursacht eine Widerstandserhöhung, die zu einem Leistungsverlust führt, was indirekt einen Kapazitätsverlust induziert. [16]

Beim Alterungsprozess von Akkumulatoren spielen der Ladezustand und die C-Rate eine Rolle. Deswegen werden an dieser Stelle die beiden Begriffe näher erläutert. Für den Ladezustand wird in den meisten Fällen der englische Begriff „state of charge“ (kurz SoC) verwendet. Dieser wird in Prozent angegeben, wobei 100% für volle Ladung steht. In den meisten Fällen wird der englische Begriff „state of charge“ (kurz SoC) verwendet. Das Komplement wird als „depth of discharge“ (kurz DoD) bezeichnet, wobei 100% für eine komplette Entladung steht. [17]

Im Folgenden werden die Formeln zur Berechnung des SoC und DoD angegeben:

$$SoC = \frac{Q(U)}{Q_0} \qquad DoD = 1 - SoC \tag{2.1}$$

$Q(U)$ entspricht der verfügbaren Ladung und Q_0 der Kapazität des voll geladenen Akkus.

Ein Maß für die Lade- und Entladeströme in Bezug auf die Nennkapazität ist der C-Faktor, auch als C-Rate bezeichnet. Dieser wird wie folgt berechnet [17]:

$$C = \frac{I_m}{Q_0} \tag{2.2}$$

I_m steht für den Strom und Q_0 für die Nennkapazität.

Im Folgenden wird auf die Faktoren eingegangen, die bei der Alterung eines Akkus eine Rolle spielen. Dies können Alter, Überladung, Balancierung, Überentladung, Produktionsstreuung, SoC, DoD, Strom und Spannung sein. Die Einflussfaktoren Temperatur, SoC und DoD sowie Strom und Spannung werden ausführlich dargestellt, weil diese im Kontext dieser Arbeit als besonders wichtig erachtet wurden. [16]

2.2.1 Abhängigkeit von der Temperatur

Die Temperatur ist vermutlich der einflussreichste Faktor im Alterungsprozess. Dies ist auf den Einfluss des arrheniusschen Gesetzes zurückzuführen. Das Gesetz besagt, dass chemische Reaktionen mit zunehmender Temperatur exponentiell wachsen. [16]

$$k = k_0 \exp\left(-\frac{E_A}{R \cdot T}\right) \tag{2.3}$$

In der Formel steht k für die Reaktionsgeschwindigkeit und k_0 für eine Konstante. E_A entspricht der Aktivierungsenergie und R der allgemeinen Gaskonstante sowie T der Temperatur in Kelvin. Ein Temperaturanstieg von 10 Grad Kelvin ergibt somit bereits eine Verdoppelung der Reaktionsgeschwindigkeit [16].

2.2.2 Abhängigkeit von der Spannung

Die Spannung wirkt sich ebenfalls deutlich auf dem Alterungsprozess aus. Generell kann festgestellt werden, dass gilt: je höher die Spannung, desto stärker die Alterung. Dies führt dazu, dass Lithiumzellen, die eine niedrigere absolute Zellspannung haben, wie z.B. Zellen mit Lithium-Eisenphosphat als Kathode, im Vergleich zu anderen Zellen mit höheren Spannungen, eine viel geringere Alterung aufweisen. Diese Art von Zellen überschreiten selten Potentiale über $3,8\,V$. Die Zellen mit Übergangsmetalloxiden als

Kathode erreichen eine Spannung von bis zu $4,2\,V$. Auf Grund von neuen Entwicklungen wird die verwendete Spannung in Zukunft steigen. In diesen Spannungsbereichen ist die Alterung wesentlich ausgeprägter als im Bereich zwischen $3,6$ und $3,8\,V$. [16]

Abbildung 2.1 zeigt den Einfluss der Ladespannung in Lithium-Zellen auf die Kapazität bei einem Anfangswert von 900 mAh in Abhängigkeit vom Ladezyklus. Die Testzellen wurden bei einem konstanten Strom mit einer Rate von $1\,C$ bis zur Abschaltspannung geladen. Anschließend wird dieser Spannung für $2,5\,h$ lang gehalten und dann bei $2,75\,V$ mit einer Rate von $1\,C$ entladen. [3]

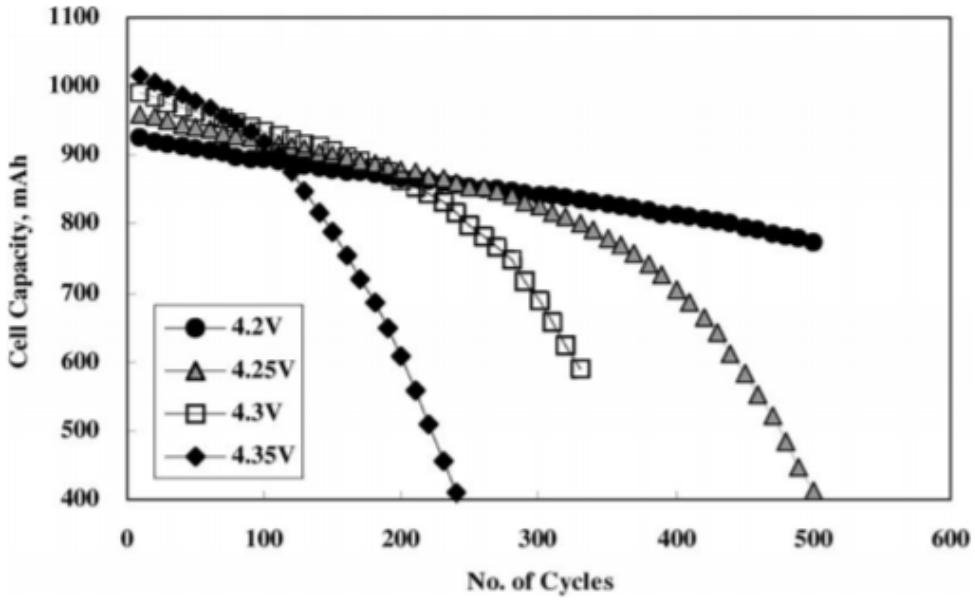


Abb. 2.1: Auswirkung der Ladespannung auf der Anzahl der Ladezyklen [3]

Zum Zweck einer optimierten Lebensdauer wird heute der maximale Spannungswert normalerweise auf $4,15\,V$ oder weniger begrenzt. Es ist dabei allerdings zu beachten, dass eine Absenkung der maximalen Spannung um 100mV eine Verringerung der verfügbaren Kapazität um ca. 15% bedeutet. [16]

2.2.3 Abhängigkeit vom Ladezustand

Ein einflussreicher Faktor im Alterungsprozess ist der SoC, der mit der Batteriespannung verknüpft ist. Ein erhöhter SoC erzeugt ein höheres Potential an der Kathode bzw. ein

niedrigeres an der Anode. Das hohe Potential der Kathode führt zur Oxidation des Elektrolyten und zur Zersetzung der Elektrode. Während das niedrige Potential der Anode die Bildung von Sekundärreaktionen erleichtert, die zur Entstehung von SEI-Film führt und damit zu einer Beschleunigung der Alterung. [7]

Abbildung 2.2 stellt den Verlauf eines dynamischen Stress-Tests (englisch: „Dynamic Stress Test, DST“) dar. Für den Test beginnt die Zelle zunächst bei einem maximal eingestellten SoC. Anschließend wird die Zelle bis zu einem minimal eingestellten SoC entladen und anschließend wieder bis zum eingestellten Maximum geladen. Der Effekt auf die Kapazität durch mehrere DST-Zyklen ist in Abbildung 2.3 zu sehen.

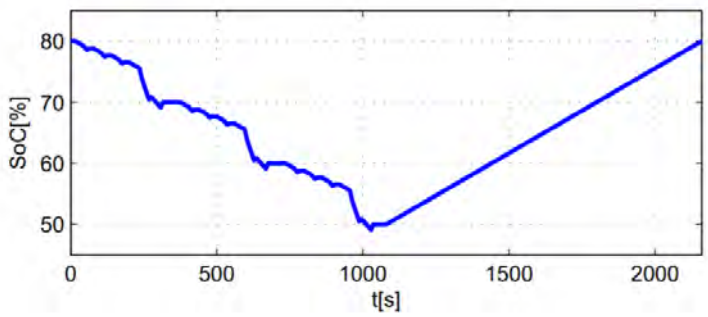


Abb. 2.2: Beispiel von einem dynamischen Stress-Test (DST) [43]

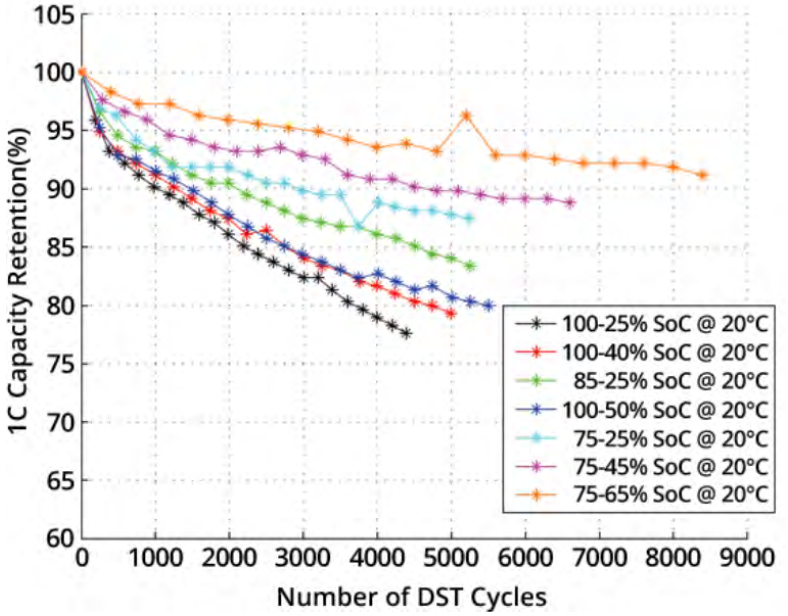


Abb. 2.3: Kapazitsveränderung als Funktion der Lade- und Entladespanne [43]

Zur Verbesserung der Lebensdauer kann daher die Gesamtkapazität eingeschränkt werden, sodass der SoC bei 80 bis 90 Prozent begrenzt werden kann [28]. Abbildung 2.4 zeigt ein Diagramm der Lebensdauer verschiedener Anwendungen. Aus dieser Grafik wird ersichtlich, dass durch die Beschränkung des maximalen Ladezustands eine spürbare Verbesserung erzielt wird.

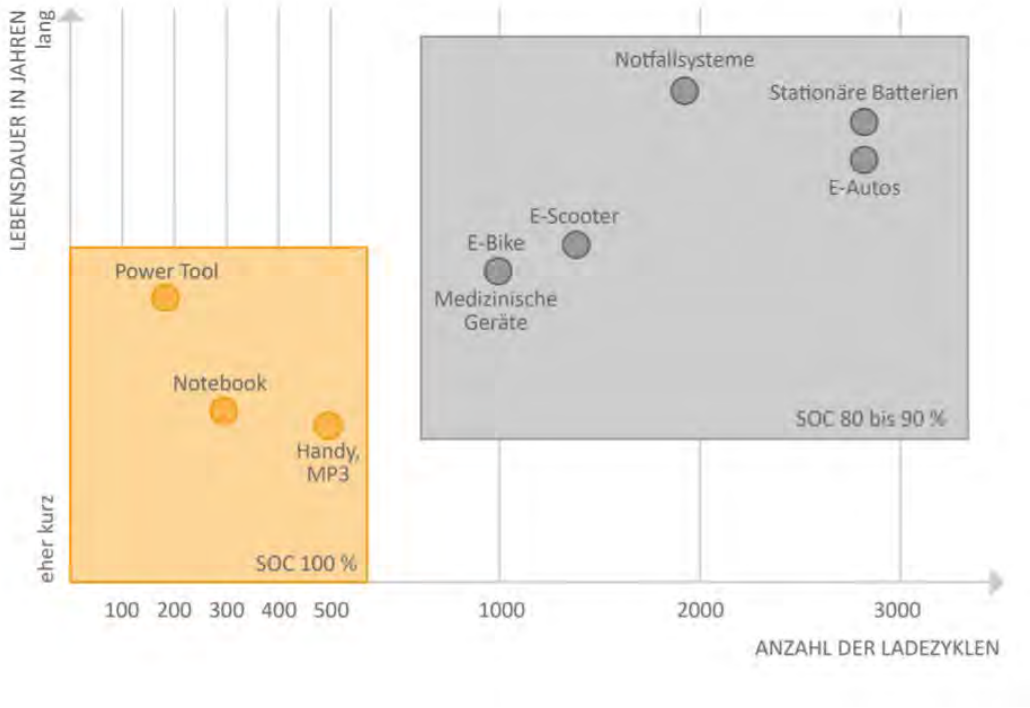


Abb. 2.4: Auswirkung der Ladebeschränkung auf die Lebensdauer [28]

2.2.4 Abhängigkeit vom Strom

Die Alterung wird auch vom Strom beeinflusst. Dieser Einfluss ist jedoch im Vergleich zu dem der Spannung gering, solange sich der Strom im Bereich der Akku-Spezifikation befindet. Die Lithium-Zellen, die für hohe Ströme nicht geeignet sind, altern deutlich schneller, falls sie mit diesen betrieben werden. Ein weiterer, zu berücksichtigender Faktor sind die großen Unterschiede, die zwischen den verschiedenen Zellen bestehen. Diese hängen vom Hersteller und den eingesetzten Materialien ab. [16]

Abbildung 2.5 zeigt den Kapazitätsverlust einer Zelle mit einem spezifizierten Dauerstrom von 5 C-Rate bei unterschiedlichen Entladeströmen.

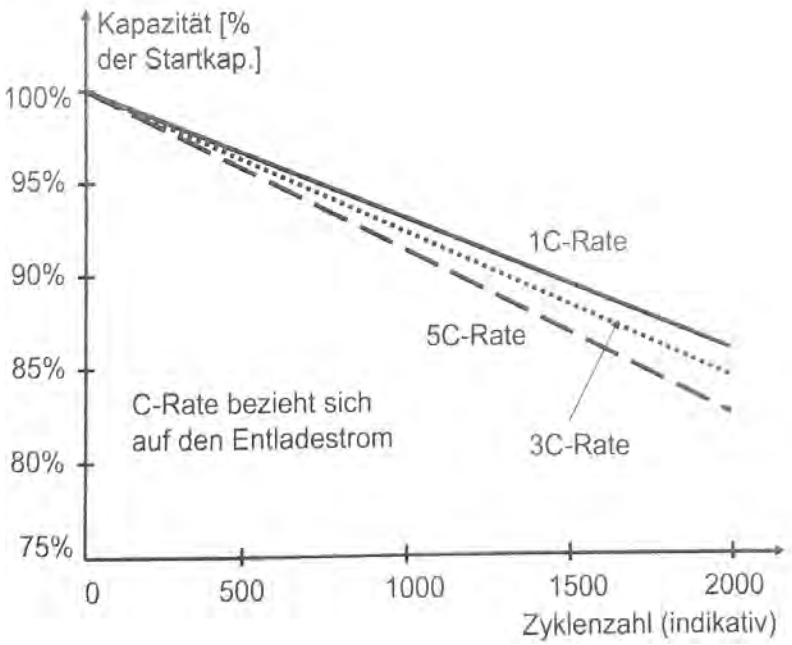


Abb. 2.5: Einfluss des Entladestroms auf die Kapazität [16]

Es ist wichtig zu erwähnen, dass der Einfluss des Stroms beim Laden deutlich höher als beim Entladen ist. Aus diesem Grund ist das schnelle Laden für eine erhöhte Lebensdauer zu vermeiden. Die auf dem Markt befindlichen Zellen, die speziell für Schnellladevorgänge ausgelegt sind, reduzieren diesen Effekt, können ihn aber nicht vollständig beheben. [16]

2.2.5 Abhängigkeit von der Zyklentiefe

Die Entladetiefe (DoD) hat einen wichtigen Einfluss auf die erreichbare Gesamtzahl der Ladezyklen. Die Anzahl von Zyklen steigt deutlich, wenn das DoD unter 100% liegt. Dies ist auf die minimale Beanspruchung zurückzuführen, die die Elektroden durch die Einlagerung von Lithium erfahren. [16]

Abbildung 2.6 zeigt den Einfluss des DoD mit der Einschränkung des SoC. Aus der Grafik geht hervor, dass die Anzahl von Zyklen mit abnehmendem DoD steigt.

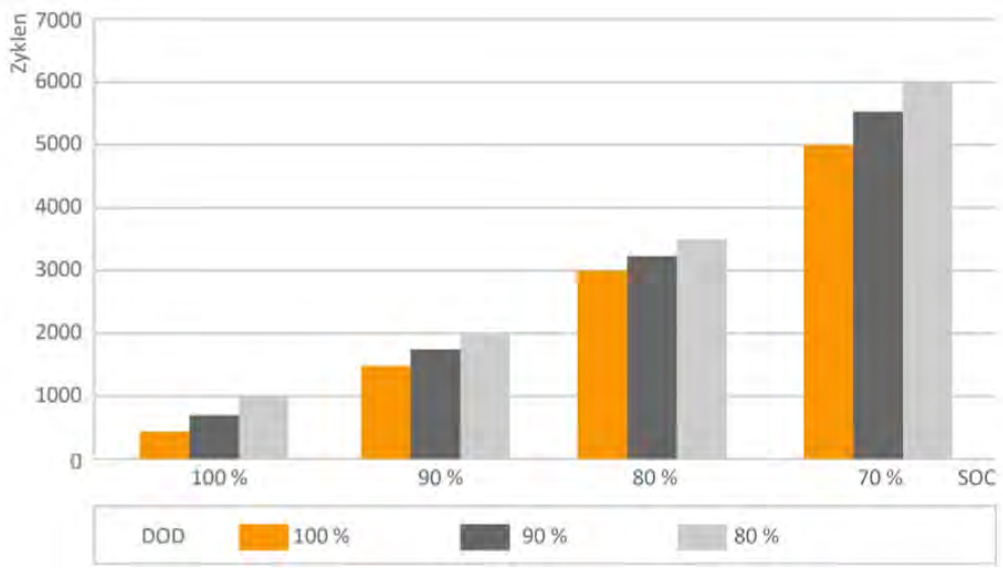


Abb. 2.6: Auswirkung der Beschränkung des SoC und DoD [28]

2.3 Smartphone-Applikation

In diesem Abschnitt wird die Android-Applikation, die von Emrah Demir im Rahmen seiner Bachelorarbeit entwickelten wurde. Die Applikation regelt den Ladevorgang durch eine Zweipunktregelung, indem sie je nach Fall ein Signal über Bluetooth zum Ein- oder Ausschalten sendet. Dafür misst sie unter anderem die Ladung des Akkus. Die Applikation ermöglicht es dem Benutzer auch, die Menge der Ladung sowie den Status der Bluetooth-Verbindung zu visualisieren [4], wie in Abbildung 2.7 zu erkennen ist.

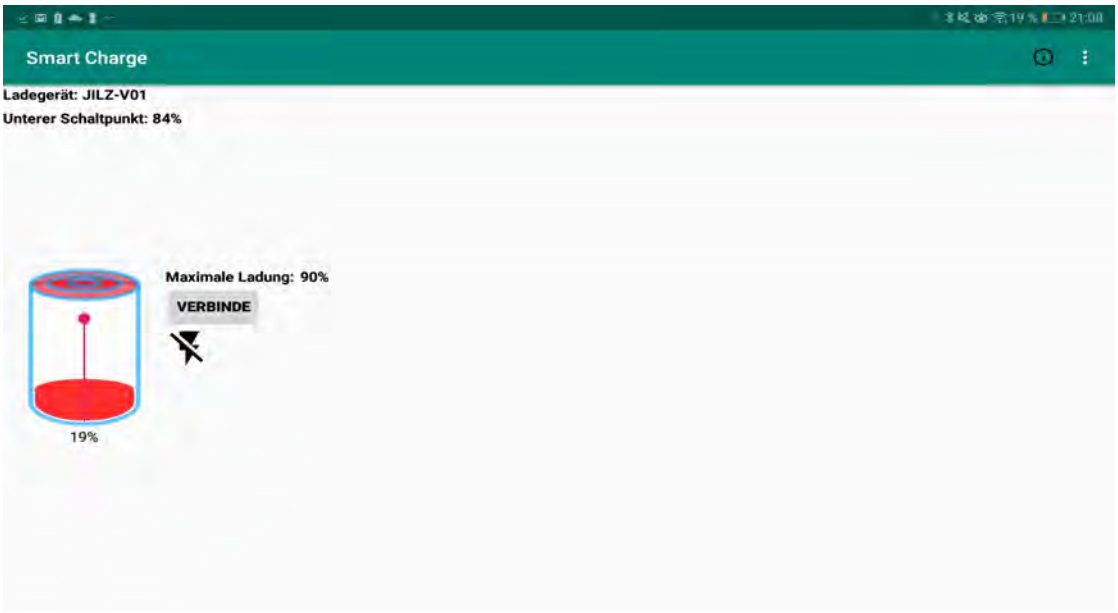


Abb. 2.7: Hauptbild der Applikation [4]

In den folgenden Abschnitten werden die Konfigurationsoptionen, die Einschränkungen, die diese Applikation hat, sowie dessen Kommunikationsprozess im Detail erläutert.

2.3.1 Konfigurationsoptionen

Die Anwendung ermöglicht es dem Benutzer, die folgenden Aktionen vorzunehmen [4].

Tab. 2.2: Vom Benutzer durchführbare Konfigurationen

Einstellung	Beschreibung
Bluetooth-Modul auswählen	Wahl des Bluetooth-Moduls, mit dem die Kommunikation aufgebaut werden soll.
Umstellung der Sprache	Der Benutzer kann die Anwendung sowohl auf Englisch als auch auf Deutsch benutzen.
Eingabe des unteren Schaltpunkts	Wahl des Mindestprozentpunktes für den Beginn der Aufladung des Akku.

2.3.2 Einschränkungen

Es handelt sich um eine Android-Applikation, die ein minimales API-Level von 19 verlangt. API-Level ist die Kennzeichnung, welche die Version eines Android-Systems angibt. In der folgende Tabelle stehen die Android-Versionen, mit denen die Applikation funktionieren sollte. Allerdings kann ab API-Level 26 die Anwendung von dem System beendet werden, wenn der Energiebedarf der Anwendung zu hoch ist. [4]

Tab. 2.3: Funktionale Android-Versionen für die Smartphone-Anwendung [4]

Codename	Version	API-Level
Android 10	10	API-Level 29
Pie	9	API-Level 28
Oreo	8.1.0	API-Level 27
Oreo	8.0.0	API-Level 26
Nougat	7.1	API-Level 25
Nougat	7.0	API-Level 24
Marshmallow	6.0	API-Level 23
Lollipop	5.1	API-Level 22
Lollipop	5.0	API-Level 21
Kitkat	4.4-4.4.4	API-Level 19

2.3.3 Kommunikationsprozess

Um festzustellen, ob die Batterie geladen werden muss, vergleicht die Applikation die aktuelle Ladung mit 2 Werten. Zuerst mit dem Wert der maximalen Ladung. Wenn die Ladung diesen Punkt erreicht, wird ein Signal gesendet, um den Ladevorgang zu stoppen. Der zweite Wert ist der untere Schaltpunkt. Wenn die Ladung unter diesem Punkt liegt, wird ein Signal zur Aktivierung des Ladevorgangs gesendet. Sollte sich die Ladung in der Mitte dieser Punkte befinden, wird kein Signal gesendet. Ein Beispiel für diesen Fall ist in Abbildung 2.8 zu erkennen. [4]

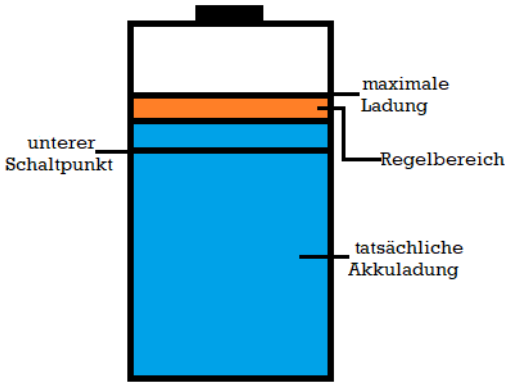


Abb. 2.8: Akkuladung im Regelbereich (modifiziert aus [4])

In den folgenden Abschnitten werden der Aus- und Einschaltvorgang im Detail erläutert.

Einschaltvorgang

Damit die Applikation das Signal senden kann, muss der Ladezustand des Akku unter dem festgelegten Schaltpunkt liegen [4].

Der Kommunikationsprozess der Einschaltvorgänge wird wie folgt durchgeführt:

1. Senden des Buchstabens e (ASCII = 0x65) über Bluetooth.

2. Auf den Bestätigungsbuchstaben X (ASCII = 0x58) warten.

Falls die Anwendung keine Bestätigung erhält, wird der Buchstabe e erneut gesendet. Falls die Bestätigung weiterhin ausbleibt, erhält der Benutzer eine Fehlerbenachrichtigung. [4]

Ausschaltvorgänge

Der Ladevorgang wird beendet, sobald die maximale Lademenge erreicht ist.

Der Abschaltvorgang verläuft wie folgt:

1. Senden des Buchstabens a (ASCII = 0x61) über Bluetooth.
2. Auf den Bestätigungsbuchstaben Y (ASCII = 0x59) warten.

Sollte die Applikation den Bestätigungsbuchstaben Y nicht erhalten, wird der Buchstabe a erneut gesendet. Falls die Bestätigung weiterhin ausbleibt, wird der Benutzer über das bestehende Verbindungsproblem informiert. [4]

3 Anforderungsanalyse

Dieses Kapitel beschäftigt sich mit der Ermittlung der Anforderungen an das Vorschaltgerät. Um die Anforderungen besser feststellen zu können, wird zunächst das Umfeld des Systems analysiert. Danach werden die verschiedenen Use Cases des Systems dargestellt. Anschließend werden darauf aufbauend die Anforderungen bestimmt.

3.1 Kontextabgrenzung

Dieser Abschnitt enthält eine Beschreibung, wie das System mit benachbarten Systemen interagiert. Mit dieser soll eine Vorstellung vom Datenfluss und der allgemeinen Funktionsweise vermittelt werden. Der Zweck dieses Abschnitts ist außerdem, die Bedeutung der Schnittstellen zu verdeutlichen und die Bestimmung der Anforderungen zu erleichtern.

Um die Schnittstelle grafisch darzustellen, wird in Abbildung 3.1 das Verteilungsdiagramm von der Unified Modeling Language (UML) verwendet [32].

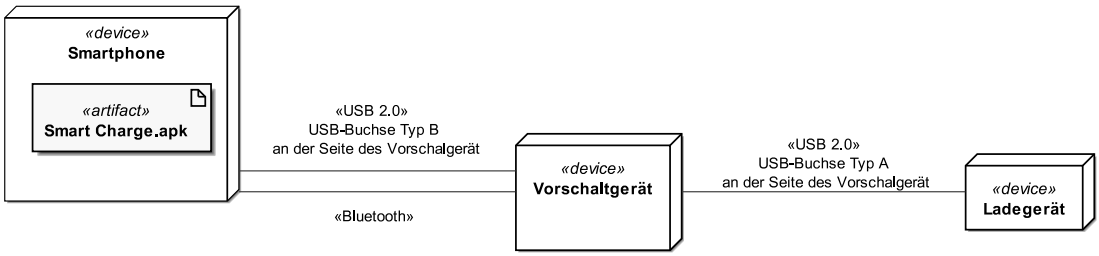


Abb. 3.1: Kontext des Vorschaltgeräts

Hierbei sind drei technischen Schnittstellen sowie eine ausführbare Datei auf dem Smartphone, in diesem Fall „Smart Charge.apk“, zu erkennen. Diese Datei erzeugt die Applikation, die in Abschnitt 2.3 bereits beschrieben wurde.

In Tabelle 3.1 werden die Aufgaben der Schnittstellen erläutert.

Tab. 3.1: Technische Zuordnung der Schnittstellen des Vorschaltgeräts

Schnittstelle	**Aufgabe**
USB-Buchse Typ A	Liefert den Ladestrom an das Smartphone
USB-Buchse Typ B	Versorgt das Vorschaltgerät mit Spannung
Bluetooth	Sendet Befehle aus der Smartphone-Anwendung an das Vorschaltgerät

3.2 Use Cases

Die Verwendung von Use Cases zielt darauf ab, die Funktionalität eines Systems aus der Nutzungsperspektive zu betrachten und zu dokumentieren. Hierdurch kann die Bestimmung von funktionalen und nicht-funktionalen Anforderungen erleichtert werden. Die Anwendung dieses Modells besteht aus der Erstellung eines Use-Case-Diagramms und Use-Case-Spezifikationen. [27]

3.2.1 Use-Case-Diagramm

In der Abbildung 3.2 wird das Use-Case-Diagramm des Systems gezeigt. In der Grafik sind unterschiedliche Reaktionen des Systems auf verschiedene Ereignisse in dessen Umgebung abgebildet. [14]

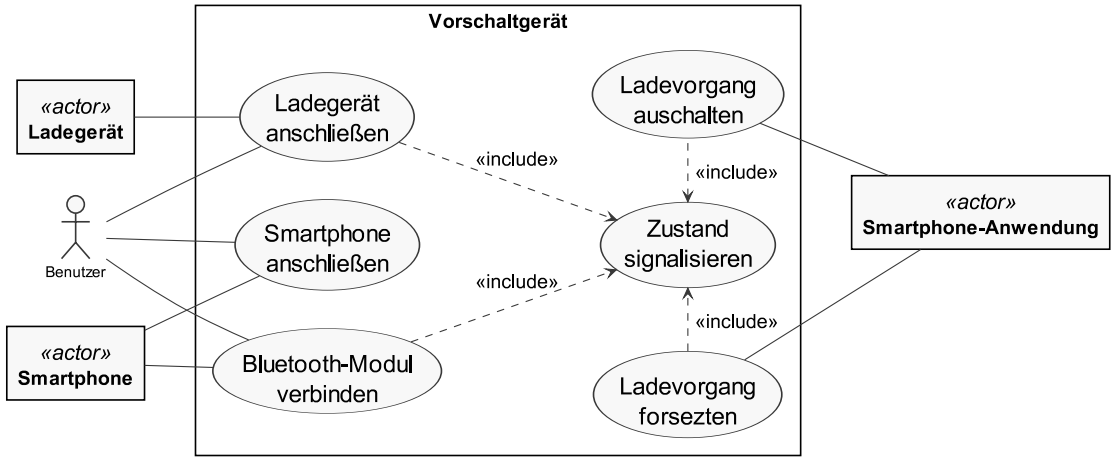


Abb. 3.2: Use-Case-Diagramm des Vorschaltgeräts

3.2.2 Use-Case-Spezifikationen

Jeder Use Case wird anhand der Schablone in Tabelle 3.2 in einer Tabelle ausformuliert, um mehr Informationen über den Anwendungsfall zu vermitteln.

Tab. 3.2: Schablone für die Use-Case-Spezifikationen (modifiziert aus [26])

Nr.	Abschnitt	Inhalt/Erläuterung
1	Bezeichner	Eindeutiger Bezeichner des Use Case
2	Name	Eindeutiger Name für den Use Case
3	Priorität	Die Priorität jedes Use Case wird entsprechend seiner Wichtigkeit für den Erfolg des Systems gemessen.
4	Kurzbeschreibung	Komprimierte Beschreibung des Use Case
5	Auslösendes Ereignis	Angabe des Ereignisses, das den Use Case auslöst
6	Akteure	Auflistung der Akteure, die mit dem Use Case in Beziehung stehen.
7	Vorbedingung	Eine Liste notwendiger Voraussetzungen, die erfüllt sein müssen, bevor die Ausführung des Use Case beginnen kann
8	Nachbedingung	Eine Liste von Zuständen, in denen sich das System unmittelbar nach der Ausführung des Hauptszenarios befindet.
9	Ergebnis	Beschreibung der Ausgaben, die während der Ausführung des Use Case erzeugt werden
10	Hauptszenario	Beschreibung des Hauptszenarios eines Use Case
11	Ausnahmeszenarien	Beschreibung von Ausnahmeszenarien des Use Case oder lediglich Angabe der auslösenden Ereignisse

Tab. 3.3: Dokumentation des Use Case „Zustand signalisieren“

Abschnitt	**Inhalt**
Bezeichner	U0
Name	Zustand signalisieren
Priorität	Wichtigkeit für Systemerfolg „niedrig“
Kurzbeschreibung	Das Vorschaltgerät signalisiert folgende Zustände: ■ Bluetooth-Verbindung inaktiv: In diesem Zustand ist das Gerät an kein Endgerät angeschlossen. Die Ladespannung ist eingeschaltet, was das Laden des Smartphones ermöglicht. ■ Ladevorgang aktiv: In diesem Zustand sind sowohl die Ladespannung als auch die Bluetooth-Verbindung aktiv. ■ Ladevorgang abgeschlossen: In diesem Zustand ist die Bluetooth-Verbindung aktiv, aber die Ladespannung ist abgeschaltet, was den Ladevorgang des Smartphones stoppt.
Auslösendes Ereignis	Änderung des Zustands
Vorbedingung	Das Vorschaltgerät muss eingeschaltet sein.
Nachbedingung	Die Status-Zustände werden der Definition entsprechend angezeigt.
Ergebnis	Anzeige des Zustands

Für eine detailliertere Erklärung des Verhaltens von dem Use Case „Zustand signalisieren“ wird ein UML-Zustandsdiagramm verwendet, das in Abbildung 4 zu sehen ist [14].

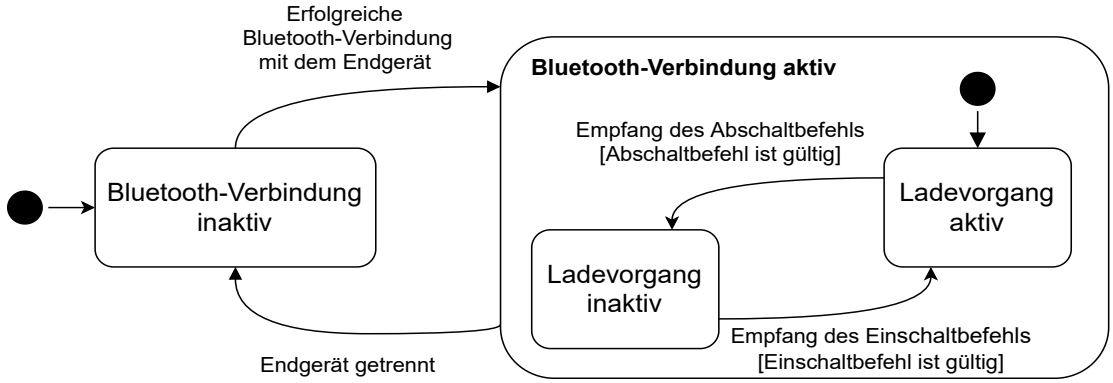


Abb. 3.3: Ablaufbeschreibung des Use Case „Zustand signalisieren“

Es gibt drei Zustände, einen, in dem die Bluetooth-Kommunikation deaktiviert ist, und zwei andere, in denen die Bluetooth-Verbindung aktiv ist. Der Unterschied zwischen diesen beiden Zuständen besteht darin, dass in einem Zustand das Smartphone aufgeladen wird und im anderen nicht. Der Initialzustand „Bluetooth-Verbindung inaktiv“ ist der Ausgangspunkt, wobei der erste Zustandswechsel eintritt, wenn eine Bluetooth-Verbindung zwischen Vorschaltgerät und Smartphone hergestellt wird. Danach geht das Gerät automatisch in den Zustand „Ladevorgang aktiv“ über, ein Zustandswechsel nach „Ladevorgang inaktiv“ erfolgt, wenn die Smartphone-Anwendung den Befehl zum Stoppen des Stromflusses an das Smartphone sendet. Es besteht hier ebenfalls die Möglichkeit, in den Zustand „Ladevorgang aktiv“ zurückzukehren, wenn die Anwendung den entsprechenden Befehl sendet. Eine Unterbrechung der Bluetooth-Verbindung führt zu einem Statuswechsel auf „Bluetooth-Verbindung inaktiv“.

Tab. 3.4: Dokumentation des Use Case „Ladegerät anschließen“

Abschnitt	**Inhalt**
Bezeichner	U1
Name	Ladegerät anschließen.
Priorität	Wichtigkeit für Systemerfolg „hoch“
Kurzbeschreibung	Der Benutzer schließt das Vorschaltgerät mittels eines Anschlusskabels an das entsprechende Ladegerät seines Smartphones an.
Auslösendes Ereignis	Der Benutzer möchte sein Smartphone laden.
Akteure	Benutzer, Ladegerät
Vorbedingung	Es muss ein Anschlusskabel mit einem USB-Type-A-Stecker und einem USB-Type-B-Stecker vorhanden sein.
Nachbedingung	Das Vorschaltgerät ist unter Spannung und geht in den Zustand „Bluetooth-Verbindung inaktiv“ über.
Ergebnis	Betriebsbereitschaft
Hauptszenario	1. Der Benutzer schließt das Handy-Ladegerät an die Steckdose an. 2. Der Benutzer steckt den USB-Type-A-Stecker des Kabels in die USB-Type-A-Buchse des Ladegerätes. 3. Der Benutzer steckt den USB-Type-B-Stecker des Kabels in die USB-Type-B-Buchse des Vorschaltgerätes. 4. Das Vorschaltgerät schaltet die Ladespannung ein. 5. Das Vorschaltgerät signalisiert dem Benutzer den Zustand „Bluetooth-Verbindung inaktiv“.

Tab. 3.5: Dokumentation des Use Case „Smartphone anschließen“

Abschnitt	**Inhalt**
Bezeichner	U2
Name	Smartphone anschließen
Priorität	Wichtigkeit für Systemerfolg „hoch“
Akteure	Benutzer, Smartphone
Kurzbeschreibung	Der Benutzer verbindet das Gerät und seinem Smartphone mit Hilfe seines jeweiligen Ladekabels.
Auslösendes Ereignis	Der Benutzer möchte sein Smartphone laden.
Vorbedingung	Ein Anschlusskabel muss mit einem USB-Type-A-Stecker sowie mit dem entsprechenden Stecker für den Ladeanschluss des Smartphones vorhanden sein.
Nachbedingung	Das Smartphone ist zum Aufladen bereit.
Ergebnis	USB-Verknüpfung zwischen Vorschaltgerät und Smartphone
Hauptszenario	1. Der Benutzer steckt den USB-Type-A-Stecker vom Kabel in die USB-Type-A-Buchse des Vorschaltgeräts. 2. Der Benutzer steckt den passenden Stecker für den Ladeanschluss des Smartphones ein.

Tab. 3.6: Dokumentation des Use Case „Bluetooth-Modul verbinden“

Abschnitt	**Inhalt**
Bezeichner	U3
Name	Bluetooth-Modul verbinden
Priorität	Wichtigkeit für Systemerfolg „hoch“
Akteure	Benutzer, Smartphone
Kurzbeschreibung	Es wird eine Bluetooth-Verbindung zwischen dem Vorschaltgerät und dem Smartphone hergestellt.
Auslösendes Ereignis	Der Benutzer möchte, dass die Smartphone-Anwendung den Ladevorgang regelt.
Vorbedingung	Das Vorschaltgerät muss mit Spannung versorgt sein.
Nachbedingung	Das Vorschaltgerät kann über Bluetooth kommunizieren und ist im Zustand „Ladevorgang aktiv“.
Ergebnis	Erfolgreiche Bluetooth-Verbindung
Hauptszenario	1. Der Benutzer wählt das Vorschaltgerät als das zu verbindende Gerät. 2. Das Smartphone sendet eine Verbindungsanfrage an das Vorschaltgerät. 3. Das Smartphone verbindet sich mit dem Vorschaltgerät. 4. Das Vorschaltgerät signalisiert dem Benutzer den Zustand „Ladevorgang aktiv“.

Tab. 3.7: Dokumentation des Use Case „Ladevorgang ausschalten“

Abschnitt	**Inhalt**
Bezeichner	U4
Name	Ladevorgang ausschalten
Priorität	Wichtigkeit für Systemerfolg „hoch“
Akteure	Smartphone-Anwendung
Kurzbeschreibung	Die Smartphone-Anwendung sendet einen Befehl an das Vorschaltgerät, damit es den Ladevorgang des Smartphones stoppt.
Auslösendes Ereignis	Die Smartphone-Anwendung sendet den Befehl zum Stoppen des Ladevorgangs.
Vorbedingung	Die Verfügbarkeit einer Bluetooth-Verbindung zwischen Vorschaltgerät und Smartphone.
Nachbedingung	Das Vorschaltgerät geht in dem Zustand „Ladevorgang inaktiv“ über.
Ergebnis	Empfang des Abschaltbefehls
Hauptszenario	1. Das Vorschaltgerät erhält die Anweisung, den Ladevorgang zu stoppen. 2. Das Vorschaltgerät beendet den Ladevorgang durch Abschalten der Ladespannung. 4. Das Vorschaltgerät signalisiert dem Benutzer den Zustand „Ladevorgang inaktiv“.
Ausnahmeszenarien	Auslösendes Ereignis: Der empfangene Befehl entspricht nicht dem gesendeten Befehl zum Stoppen des Ladevorgangs.

Tab. 3.8: Dokumentation des Use Case „Ladevorgang fortsetzen“

Abschnitt	**Inhalt**
Bezeichner	U5
Name	Ladevorgang fortsetzen
Priorität	Wichtigkeit für Systemerfolg „Hoch“
Akteure	Smartphone-Anwendung
Kurzbeschreibung	Das Vorschaltgerät schaltet den Ladevorgang ein, nachdem es den entsprechenden Befehl von der Smartphone-Anwendung erhalten hat.
Auslösendes Ereignis	Die Smartphone-Anwendung sendet den Befehl zum Starten des Ladevorgangs.
Vorbedingung	Der Ladevorgang wurde gestoppt und das Vorschaltgerät verfügt über eine Bluetooth-Verbindung.
Nachbedingung	Das Vorschaltgerät kehrt in dem Zustand „Ladevorgang aktiv“ zurück.
Ergebnis	Empfang des Einschaltbefehls
Hauptszenario	1. Das Vorschaltgerät empfängt den Befehl zur Aktivierung des Ladevorgangs. 2. Das Vorschaltgerät schaltet den Ladevorgang durch Einschalten der Ladespannung ein. 4. Das Vorschaltgerät signalisiert dem Benutzer den Zustand „Ladevorgang aktiv“.
Ausnahmeszenarien	Auslösendes Ereignis: Der empfangene Befehl entspricht nicht dem gesendeten Befehl zum Starten des Ladevorgangs.

3.3 Anforderungen an das Vorschaltgerät

Das Projekt wird zunächst als Prototyp entwickelt, aber die Idee ist es, ein Produkt zu entwickeln, das auf dem Markt gebracht werden kann. Daher ist es notwendig, die Wahl des Endverbrauchers sowie des zukünftigen Herstellers des Vorschaltgeräts zu berücksichtigen. Sowohl der Entwickler der Smartphone-Applikation als auch der des Vorschaltgeräts spielen eine wichtige Rolle, um ein Projekt schnell und möglichst fehlerfrei durchführen zu können. Die Anforderungen an das Vorschaltgerät werden in Tabelle 3.9 dargestellt, wobei jede Anforderung mit einer ID versehen wird.

Tab. 3.9: Liste der Anforderungen

ID	Beschreibung
Req-1	Falls das Vorschaltgerät mit Spannung versorgt wird und nicht an ein Endgerät angeschlossen ist, muss das Vorschaltgerät fähig sein, eine Verbindungsanfrage eines Endgeräts anzunehmen.
Req-2	Falls der Ladevorgang bereits durch einen vorherigen Befehl von der Smartphone-Anwendung gestoppt wurde, muss das Vorschaltgerät fähig sein, den Befehl zum Einschalten des Ladevorgangs von der Smartphone-Anwendung zu interpretieren und auszuführen.
Req-3	Falls das Vorschaltgerät über Bluetooth mit der Smartphone-Applikation verbunden ist, muss das Vorschaltgerät fähig sein, den Befehl zum Abschalten des Ladevorgangs von der Smartphone-Anwendung zu interpretieren und auszuführen.
Req-4	Sobald das Vorschaltgerät mit Spannung versorgt wird, sollte das Vorschaltgerät in der Lage sein, dem Benutzer die 3 Zustände „Bluetooth-Verbindung inaktiv“, „Ladevorgang aktiv“ und „Ladevorgang inaktiv“ zu signalisieren.
Req-5	Das Vorschaltgerät sollte dem Benutzer die Möglichkeit bieten, sich mit einer PIN zu identifizieren.
Req-6	Das Vorschaltgerät muss über eine USB-Buchse vom Typ A verfügen.
Req-7	Das Vorschaltgerät muss über eine USB-Buchse vom Typ B überstehen.
Req-8	Das Vorschaltgerät sollte die Abmessungen 7cm x 5 cm x 4cm nicht überschreiten.
Req-9	Das Vorschaltgerät sollte Stürze aus einer Höhe von 60 cm aushalten.
Req-10	Die Zustandsanzeige sollte dem Benutzer die Möglichkeit bieten, die drei verschiedenen Zustände des Vorschaltgeräts intuitiv zu erkennen.
Req-11	Die Software muss in der Programmiersprache C implementiert werden.
Req-12	Die Hardware Komponenten des Vorschaltgeräts sollten einen Gesamtpreis von 10 Euro nicht überschreiten.

Die Anforderungen werden mit Attributen versehen, um die Möglichkeit zu haben, die Anforderungen zu sortieren und somit in der Lage zu sein, die Anforderungen in den gleichen Kontext zu bringen und die Kontrolle über die Anforderungen zu behalten [6].

In der Tabelle 3.10 werden die Anforderungen aus der Tabelle 3.9 wie folgt eingeordnet [27]:

- Name:
  Die Anforderung wird mit einem Namen versehen, um das allgemeine Verständnis zu erleichtern.
- Anforderungstyp:
  Die Anforderungen werden in Qualitätsanforderungen, Funktionsanforderungen, Rahmenbedingungen unterteilt.
- Use Case:
  Das Attribut erwähnt den Use Case, der an der Anforderung beteiligt ist.
- Querbezüge:
  Dieses Attribut gibt die Anforderungen an, die zur Erfüllung der betreffenden Anforderung erforderlich sind.
- Stakeholder:
  Die Anforderung wird mit dem jeweiligen Stakeholder, der beteiligt ist, versehen.

Tab. 3.10: Attribuierung der Anforderungen (Vorschaltgerät als VSG abgekürzt)

ID	Name	Anforderungstyp	Use Case	Querbezüge	Stakeholder
Req-1	Bluetooth-Verbindung	Funktionale Anforderung	U3	Req-7	Entwickler App. Entwickler VSG
Req-2	Fortsetzen des Ladevorgangs	Funktionale Anforderung	U5	Req-1 Req-3	Entwickler App. Entwickler VSG
Req-3	Abbrechen des Ladevorgangs	Funktionale Anforderung	U3	Req-1 Req-6	Entwickler App. Entwickler VSG
Req-4	Zustandsanzeige	Funktionale Anforderung	U0	Req-1 Req-2 Req-3	Endverbraucher Entwickler VSG
Req-5	PIN-Identifizierung	Funktionale Anforderung	U3	Req-1	Entwickler VSG Endverbraucher
Req-6	Ausgangs-schnittstelle	Randbedingung	U2	-	Endverbraucher
Req-7	Eingangs-schnittstelle	Randbedingung	U1	-	Endverbraucher
Req-8	Abmessung	Qualitäts-anforderung	-	-	Endverbraucher
Req-9	Sturz-verträglichkeit	Qualitäts-anforderung	U1 U2	-	Endverbraucher
Req-10	Verständliche Signalgebung	Qualitäts-anforderung	U0	Req-4	Endverbraucher
Req-11	Programmier-sprache	Randbedingung	-	-	Entwickler VSG
Req-12	Umsetzungs-kosten	Qualitäts-anforderung	-	-	Hersteller

4 Design und Entwurf

In diesem Kapitel wird die Struktur des Vorschaltgeräts auf der Grundlage der zuvor in Kapitel 3 dargelegten Anforderungen entworfen.

4.1 Komponentenauswahl

Im Folgenden werden die nötigen Komponenten aufgeführt. Im Anschluss folgt die spezifische Auswahl der Bauteile.

Zuerst wird eine Element benötigt, das die Steuerungs- und Kommunikationsaufgaben des Vorschaltgeräts bewältigt. Dabei besteht die Herausforderung darin, zwischen einem Mikrocontroller oder einem Mikroprozessor zu wählen. Dies soll unter Berücksichtigung der folgenden Aspekte stattfinden:

- Größe (Req-8):
  Die Komponente sollte möglichst klein sein.
- Programmiersprache (Req-11):
  Es muss möglich sein, die Software in der Programmiersprache C zu implementieren.
- Funktion:
  Die Komponente muss per Bluetooth empfangene Befehle der Smartphone-Anwendung interpretieren können.
- Kosten (Req-12):
  Die Kosten sollen möglichst gering gehalten werden.

Da sowohl Mikrocontroller als auch Mikroprozessor in C programmierbar sind und beide die Funktion erfüllen, wird die Entscheidung auf Basis von Kosten und Größe gefällt. In diesen Aspekten schneidet der Mikrocontroller besser ab. Deswegen wird in dieser Arbeit mit einem Mikrocontroller gearbeitet.

Weiterhin werden folgende Komponenten benötigt:

- Bluetooth-Modul (Req-5 und Req-1):
  Diese Komponente empfängt und sendet Daten per Bluetooth.
- Elektronischer Schalter (Req-2 und Req-3):
  Die Komponente ist für das Ein- und Abschalten der Ladespannung zuständig.
- Zustandsanzeige (Req-4 und Req-10):
  Die Zustandsanzeige stellt die folgenden drei Zustände dar: „Bluetooth inaktiv“, „Ladevorgang inaktiv“ und „Ladevorgang aktiv“.
- USB-Buchse vom Typ A (Req-6):
  Hierbei handelt es sich um die Schnittstelle zwischen Smartphone und Gerät, die die Aufgabe hat, die Ladespannung an das Smartphone zu liefern.
- USB-Buchse vom Typ B (Req-7):
  Diese bildet die Schnittstelle zwischen Vorschalt- und Ladegerät.
- Gehäuse (Req-9):
  Diese Komponente ist für die Gewährleistung der Berührungssicherheit notwendig.
- Spannungsregler:
  Ein Spannungsregler ist nötig, da die Ausgangsspannung des Ladegeräts je nach Hersteller variieren kann. Es wird ein $3,3\,V$ Spannungsregler ausgewählt, da die meisten Bluetooth-Module mit einer Spannung von $3,3\,V$ arbeiten.

Mit diesen Komponenten können alle Use Cases abgebildet werden. In den folgenden Abschnitten wird eine konkrete Auswahl der Komponenten getroffen.

4.1.1 Bluetooth-Modul

Die folgenden Kriterien werden bei der Auswahl eines Bluetooth-Moduls berücksichtigt:

- Niedriger Preis
- Geringe Größe
- Hohe Marktverfügbarkeit:
  Die Verfügbarkeit des Produkts in den örtlichen Geschäften ist von großer Bedeutung. Dieser Faktor erhält besondere Aufmerksamkeit, da durch die COVID-19-Pandemie Lieferzeiten z.T. stark angestiegen sind.
- Niedriger Stromverbrauch:
  Der Stromverbrauch spielt eine Rolle, da das Gerät ein Zwischengerät zwischen dem Smartphone und dem Ladegerät ist. Je niedriger der Energieverbrauch ist, desto geringer ist die Belastung des Ladegeräts.
- Gute Dokumentation:
  Dokumentation bezieht sich auf die Erläuterungen zu der Bedienung des Moduls und die verfügbaren Anwendungsbeispiele. Eine gute Dokumentation führt zur Einsparung von Zeit.

Zur näheren Betrachtung werden an dieser Stelle die drei meist gekauften Module der Firma „HC Information Technology“ HC-06, HC-08 und HC-09 aufgrund ihrer hohe Verfügbarkeit herangezogen und miteinander verglichen. Auch verfügen sie über eine Benutzeridentifizierung mittels einer PIN (Req-5). Um eine Entscheidung zu treffen, wird eine Pugh-Matrix erstellt [15]. Der erste Schritt dieser Methode ist die Ermittlung der Bewertungskriterien, dafür werden die oben genannten Kriterien genommen.

Jedes Kriterium erhält eine Gewichtung zwischen 1 als niedrigste und n als höchste Gewichtung, wobei n für die Gesamtzahl der Kriterien steht. Für dieses Projekt ergibt sich die folgende Reihenfolge:

1. Gute Dokumentation
2. Geringe Größe
3. Hohe Verfügbarkeit
4. Niedriger Preis

5. Stromverbrauch

Der Stromverbrauch wird für diese Arbeit am stärksten gewichtet, da er die Funktionalität des Vorschaltgeräts beeinflusst, indem er mit dem zu ladenden Smartphone um den verfügbaren Strom konkurriert. Ein niedriger Preis wird als zweitwichtigstes Kriterium gewertet, da er direkt mit den aufgestellten Anforderungen zusammenhängt (Req-12). Die geringe Größe, eine gute Dokumentation und eine hohe Verfügbarkeit sind weniger wichtig, weil sie lediglich den Aufwand bei diesem Projekt beeinflussen, darüber hinaus aber kaum Auswirkungen zeigen.

Als zweiter Schritt wird eine Basislösung gewählt, die als Standard definiert wird, an dem alle anderen Lösungen gemessen werden. In diesem Fall wird das HC-08 aufgrund des niedrigen Preises und des geringen Energieverbrauchs als Basislösung gewählt.

Die letzten Schritte bestehen in der Erstellung einer Pugh-Matrix und deren Auswertung. In der Pugh-Matrix, die in Tabelle 4.1 zu sehen ist, werden die Lösungen mit der Basislösung in Bezug auf die zuvor definierten Kriterien abgeglichen. Ein besseres Ergebnis als die Basislösung wird mit einem (+) und ein schlechteres Ergebnis mit einem (-) gekennzeichnet. Für jede Lösung wird die Anzahl gleicher Bewertungen addiert und nach den Kriterien gewichtet. Die Bewertungen in der Pugh-Matrix wurden aufgestellt, nachdem die Spezifikationen aus den Datenblättern entnommen wurden.

Tab. 4.1: Pugh-Matrix: Bluetooth-Modul

Bluetooth-Modul / **Kriterien**	**HC-06**	**HC-09**	**HC-08**	**Gewichtung**
Dokumentation	+	-	0	1
Geringe Größe	-	+	0	2
Hohe Verfügbarkeit	+	-	0	3
Niedriger Preis	+	-	0	4
Stromverbrauch	-	+	0	5
Summe +	3	2	0	
Summe -	2	3	0	
Summe 0	0	0	0	
Gewichtete Summe +	8	7	0	
Gewichtete Summe -	7	8	0	

Der Pugh-Matrix 4.1 nach ist, das bestgeeignete Modul das HC-06. Ausschlaggebend waren die hohe Verfügbarkeit und der Preis.

Die wichtigsten Merkmale des HC-06 Bluetooth-Moduls im Bezug auf das Projekt sind die folgenden [9]:

- Verfügt über eine eingebaute 2,4 GHz-Antenne
- Temperaturbereich zwischen -25 ℃ und +75℃
- Arbeitsspannung zwischen 3.1 V und 4.2 V
- Abmessungen: 27mm×13mm×2mm
- Kommunikationsprotokoll: UART
- Der Stromverbrauch im verbundenen Zustand beträgt $8\,mA$

- Baudrate: 1200, 2400, 4800, 9600 (Standard), 19200, 38400, 57600, 115200, 230400
- Verfügt über einen Status-Pin, der ein niederfrequentes Rechtecksignal liefert, wenn das Modul mit keinem Endgerät verbunden ist, und einen ständigen High-Pegel, wenn das Modul verbunden ist.

Das Bluetooth-Modul hat 34 Pins mit unterschiedlichen Funktionalitäten, wie in der Abbildung 4.1 der Pinbelegung des Moduls zu sehen ist.

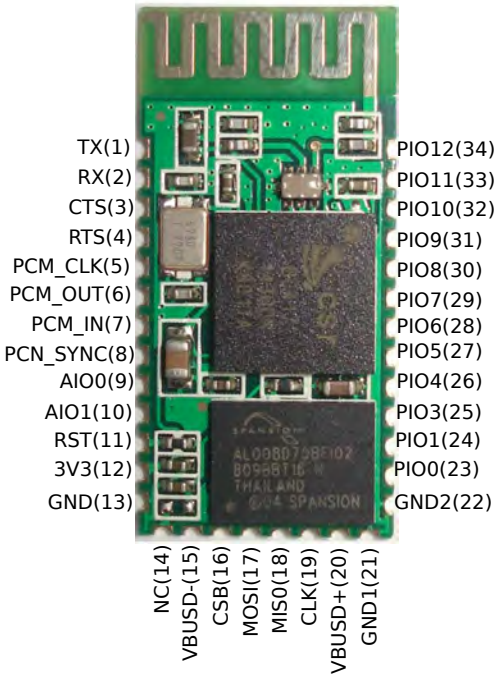


Abb. 4.1: Pinbelegung des HC-06 Bluetooth-Moduls [9]

Für die Realisierung des Projekts sind die folgenden Pins notwendig:

- Tx (1):
  Dieser Pin ist der TX-Pin der seriellen Kommunikation. Die vom Modul über Bluetooth empfangenen Daten werden später mittels dieses Pins an den Mikrocontroller weiter gesendet.
- Rx (2):
  Dieser Pin ist der RX-Pin der seriellen Kommunikation. Die vom Pin empfangenen Daten werden anschließend via Bluetooth an das Smartphone gesendet.

- RST (11):
  Dies ist der Pin, der das Bluetooth-Modul zurücksetzt, wenn der Pin einen Low-Pegel hat. Falls dieser Pin nicht angeschlossen ist, wird der Pegel des Pins High und das Bluetooth-Modul wird nicht zurückgesetzt.
- PIO1 (24):
  Dieser Pin ist der Status-Pin, der signalisiert, ob eine Verbindung zwischen dem Endgerät und dem Bluetooth-Modul besteht.
- GND (12)
- 3V3 (13)

4.1.2 Elektronischer Schalter

Bei der Auswahl eines elektronischen Schalters, der den Anforderungen des Geräts entspricht, werden die folgenden Kriterien beachtet:

- Geringe Größe
- Niedriger Preis
- Niedrige Verluste
- Einfache Treiberschaltung:
  Elektronische Schalter erfordern normalerweise eine Treiberschaltung, die den nötigen Strom zur Verfügung stellt. Je komplexer sie sind, desto größer sind die Anforderungen an die Hardware, was zu erhöhten Kosten und Platzbedarf führt. Daher ist es wichtig, dass der Schalter keine komplexe Treiberschaltung benötigt.

Um die Pugh-Matrix zu erstellen, werden die oben genannte Kriterien wie folgt gewichtet:

1. Niedrige Verluste
2. Einfache Treiberschaltung
3. Niedriger Preis
4. Geringe Größe

Die geringe Größe wird mit der höchsten Gewichtung versehen, weil die Wahl des HC-06 einen größeren Platzverbrauch mit sich brachte. Deshalb muss sichergestellt werden, dass die anderen Komponenten möglichst platzsparend sind (Req-8).

Die folgenden elektronischen Schalter sind in der Pugh-Matrix von Tabelle 4.2 zu erkennen: MOSFET, Relais, PhotoMOS-Relais. Wobei als Basiskonzept das Relais gewählt wurde, da die in Abschnitt 2.3 beschriebene Smartphone-Anwendung mit einem Prototyp des Vorschaltgeräts getestet wurde, in der ein Relais verbaut war [4]. Die Bewertungen in der Pugh-Matrix wurden vorgenommen, nachdem Beispiele analysiert wurden, in denen diese Schalter verwendet wurden [25, 29, 41].

Tab. 4.2: Pugh-Matrix: Elektronischer Schalter

Elektronischer Schalter / **Kriterien**	**PhotoMOS-Relais**	**MOSFET**	**Relais**	**Gewichtung**
Niedrige Verluste	0	+	0	1
Einfache Treiberschaltung	+	0	0	2
Niedriger Preis	-	+	0	3
Geringe Größe	+	+	0	4
Summe +	2	3	0	
Summe -	1	0	0	
Summe 0	1	1	0	
Gewichtete Summe +	6	8	0	
Gewichtete Summe -	3	0	0	

Das Relais hat den Vorteil, mit einer einfachen Treiberschaltung gesteuert zu werden, die aus einem Transistor, Widerständen und einer Diode gebildet wird, aber ein schwerwiegender Nachteil besteht in seiner Größe [29]. Das PhotoMOS-Relais wird mit dem Licht einer LED gesteuert [25]. Das bringt einen großen Vorteil, weil eine LED nur einen Vorwiderstand benötigt. Allerdings spricht der hohe Preis gegen diesen Schalter. Der

MOSFET wird wegen seines niedrigen Preises und der großen Auswahl von verschiedenen Modellen interessant.

Aus der Tabelle 4.2 ist zu entnehmen, dass sowohl das PhotoMOS-Relais als auch der MOSFET bessere Optionen als ein Relais sind. In diesem Projekt wird jedoch mit einem MOSFET gearbeitet, da dieser im Vergleich zu einem PhotoMOS-Relais am günstigsten ist.

4.1.3 Spannungsregler

Die Parameter, welche die Entscheidung für den zu wählenden Spannungsregler beeinflussen, sind:

- Niedriger Preis
- Geringe Größe
- Niedrige Verluste
- Hohe Marktverfügbarkeit

Es werden die folgenden zwei Spannungsregler von der Firma „Texas-Instruments“ verglichen: „LP2985-33DBVR“ und „TPS76933DBVR“. Der Hersteller wurde gewählt, weil diese Firma für ihre gute Dokumentation und die Erstellung von Spannungsreglern mit geringem Stromverbrauch bekannt ist [35]. Beide Regler sind vom Typ Low-Dropout (LDO), dadurch kommt es nur zu geringen Verlusten. Außerdem verfügen diese über ein SOT-23-5-Gehäuse, das viel Platz spart [36, 37]. Da die Größe und die Verluste beider Optionen gering sind, wird die Entscheidung nur auf Basis von Preis und Marktverfügbarkeit getroffen. In beiden Bereichen übertrifft der „LP2985-33DBRV“-Spannungsregler das andere Modelle, sodass er für dieses Projekt verwendet wird.

Wichtige Features dieses Spannungsreglers sind [36]:

- Maximale Stromlieferung von $150\,mA$
- Überstrom- und Wärmeschutz
- Großer Eingangsspannungsbereich: 16 V (maximal)

4.1.4 Zustandsanzeige

Diese Komponente hat die Aufgabe, dem Benutzer die Zustände des Vorschaltgeräts mitzuteilen (Req-10). Dafür ist wegen ihrer Größe und der einfachen Steuerung mit einem Mikrocontroller die Nutzung von LEDs sinnvoll. Es bietet sich an, mit einer RGB-LED zu arbeiten, damit verschiedene Farben generiert werden können und somit weniger Platz als bei der Verwendung mehrerer LEDs verbraucht wird. Es gibt zwei verschieden Arten von RGB-LED: eine mit einer gemeinsamen Anode und eine mit einer gemeinsamen Kathode. Beide Varianten werden in der Abbildung 4.2 dargestellt.

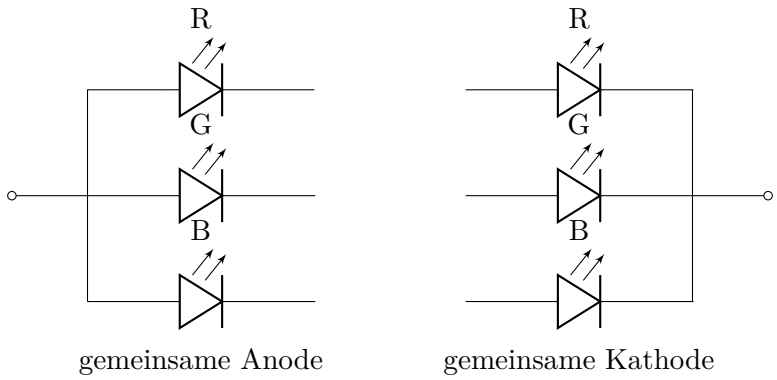


Abb. 4.2: RGB-LED Schaltbilder

Mit einer gemeinsamen Kathode zu arbeiten, hat den Nachteil, dass der Strom von dem Mikrocontroller geliefert werden muss. Da die Pins des Mikrocontrollers eine begrenzte Stromausgabe haben, kann die LED nicht voll beleuchtet werden. Im Gegensatz dazu hat eine gemeinsame Anode dieses Problem nicht, da der Strom vom Ladegerät bereitgestellt werden kann. Aus diesem Grund wird eine RGB-LED mit einer gemeinsamen Anode eingesetzt.

4.1.5 USB-Buchse vom Typ A und vom Typ B

Es ist wichtig, dass die Buchsen über eine gute Qualität verfügen, um trotz häufiger Verwendung eine lange Lebensdauer zu gewährleisten.

4.1.6 Gehäuse

Das Gehäuse sollte an die verbauten Komponenten angepasst sein, um so möglichst viel Platz sparen zu können. Weiterhin ist ein ansprechendes Design vorteilhaft. Es bietet sich daher an, das Gehäuse mit einem 3D-Drucker anzufertigen.

4.1.7 Mikrocontroller

Der Mikrocontroller steuert die RGB-LED und den MOSFET je nach vom Bluetooth-Modul erhaltenen Befehl. Das Bluetooth-Modul muss per UART mit dem Mikrocontroller kommunizieren können. Da die Frequenz der gesendeten Daten nicht hoch ist und der Datenaustausch nicht zeitkritisch ist, kann die UART-Schnittstelle mittels Software implementiert werden. Um ein „Software-UART“ umzusetzen, wird zunächst dargestellt, wie der UART-Kommunikationsprozess funktioniert.

Die asynchrone serielle Datenkommunikation wird durch einfache Regeln bestimmt. Jedes Byte wird zwischen einem Start-Bit und einem Stop-Bit gesetzt, um den Empfänger über die Ankunft eines neuen Byte zu informieren. Das Wort „Frame“ wird für den Namen des gesendeten Pakets verwendet. Das Format kann durch Hinzufügen eines Paritätsbits zur Erkennung von Fehlern sowie zusätzlicher Stop-Bits erweitert werden. Abbildung 4 zeigt ein grafisches Beispiel für dieses Kommunikationsprotokoll. [19]

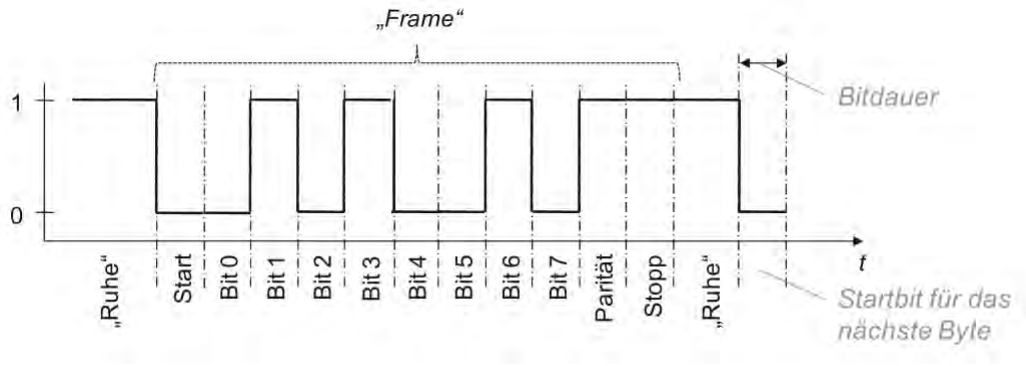


Abb. 4.3: Zeitdiagramm des UART-Protokolls [38]

Die Kommunikation findet statt, ohne dass ein Taktsignal zur Synchronisation erforderlich ist. Der korrekte Empfang der Daten wird durch die Beibehaltung der Bitdauer gewährleistet. Der Empfänger synchronisiert sich, indem er eine fallende Flanke des Startbits erkennt

und dann die nächste Abtastzeit mit seinem eigenen Timer ermittelt. Die Bitdauer wird durch die verwendete Baudrate bestimmt. Im Falle des UART-Kommunikationsprotokolls entspricht die Baudrate der Anzahl der pro Sekunde übertragenen Bits. Damit dies funktioniert, müssen sowohl der Sender als auch der Empfänger mit der gleichen Baudrate konfiguriert sein. Unter Berücksichtigung dieser Aspekte kommt man zu dem Schluss, dass ein Mikrocontroller einen Pin Interrupt für die Erkennung des Start-Bits und einen Timer für die Abtastzeit benötigt, um ein UART mit Hilfe von Software implementieren zu können. [19]

Das Bluetooth-Modul hat noch zwei weitere Pins, die mit dem Microcontroller verbunden werden. Zum Einen den RST-Pin, der dafür sorgt, dass der Mikrocontroller die Möglichkeit hat, das Bluetooth-Modul neuzustarten. Zum anderen der PIO1-Pin, der dem Mikrocontroller den Verbindungsstatus zwischen Bluetooth-Modul und Smartphone mitteilt. Die Steuerung der RGB-LED kann mit nur einem Pin des Mikrocontrollers realisiert werden, indem man den PIO1-Pin des Bluetooth-Moduls mit einer Farbe der RGB-LED verbindet, um die Zustände „Bluetooth-Verbindung inaktiv“ und „Ladevorgang aktiv“ zu signalisieren. Dies führt dazu, dass eine RGB-LED-Farbe blinkt, wenn keine Bluetooth-Verbindung besteht und die gleiche Farbe dauernd leuchtet, wenn die Verbindung erfolgreich hergestellt wurde. Es fehlt daher nur die Anzeige des Zustands „Ladevorgang inaktiv“. Die entsprechende Signalisierung wird durch die Kombination der Farbe, an die der PIO1-Pin des Bluetooth-Moduls angeschlossen wurde, und der restlichen Farbe(n) erzeugt, welche vom Mikrocontroller gesteuert werden. Zur Steuerung des MOSFETs ist nur ein Mikrocontroller-Pin erforderlich.

Der zu wählende Mikrocontroller muss folgende Eigenschaften aufweisen:

- Er muss einen Timer besitzen.
- Er muss über einen Pin Interrupt zur Erkennung einer fallenden Flanke verfügen.
- Er muss eine Mindestanzahl von 6 I/O-Pins besitzen.

Da diese Eigenschaften auf eine große Auswahl an Mikrocontrollern zutreffen, wird, um die Auswahl zu erleichtern, für diese Arbeit nur ein Hersteller in Betracht gezogen. Als Hersteller wurde „Microchip“ aufgrund seiner übersichtlichen Dokumentation und der angebotenen Tools ausgewählt, die eine einfache und schnelle Kompilierung des Codes ermöglichen. [31]

Im nächsten Schritt wurden drei 8-Bit-Mikrocontroller aus verschiedenen Familien gewählt, die einen niedrigen Preis und einen Flash-Speicher von mehr als 5 und weniger als 30 Kilobyte aufweisen. Tabelle 4.3 bietet einen Vergleich der Spezifikationen der Mikrocontroller.

Tab. 4.3: Spezifikationsvergleich der Mikrocontroller [20, 21, 22]

	PIC16LF1938	**ATmega8L**	**ATtiny85**
Preis (€)	1.73	1.99	0.96
Pin Anzahl (I/O)	25	32	8
Frequenz (MHz)	32	8	8
Flash (Kb)	28	8	8
EEPROM (bytes)	256	512	512
RAM (bytes)	1024	1024	512
Timers	4 x 8-bit 1 x 16-bit	2 x 8-bit 1 x 16-bit	2 x 8-bit
Versorgungspannung (V)	1.8 - 3.6	2.7 - 5.5	1.8 - 5.5
Standby-Strom (nA)	60 @ 1.8V	500 @ 3V	100 @ 1.8V
Betriebsstrom (μA)	43 @ 1 MHz, 1.8V	3600 @ 4 MHz, 3V	300 @ 1 MHz, 1.8V
Watchdog Timer	Verfügbar	Verfügbar	Verfügbar

Um eine Entscheidung zwischen den drei Mikrocontrollern zu treffen, wird eine Pugh-Matrix erstellt. Dabei werden die folgenden Kriterien bewertet[15]:

- Dokumentation:
  Mit Dokumentation sind Anwendungsbeispiele sowie ein gutes Datasheet gemeint. Diese helfen bei einem schnellen Aufbau der Software.
- Preis:
  Wie bei den anderen Komponenten wird auch hier aufgrund von Req-12 ein niedriger Preis gefordert.
- Pin-Anzahl:
  Eine hohe Anzahl von Pins erhöht die Schwierigkeit beim Löten des Mikrocontrollers, was zu einem Zeitverlust führt.

- Stromverbrauch:
  Um Verluste zu reduzieren, ist es wichtig, dass der Mikrocontroller eine geringe Leistungsaufnahme hat. Der zu berücksichtigende Parameter ist der Standby-Strom, da der Mikrocontroller die meiste Zeit im Sleep-Modus bleibt.

Für dieses Projekt ergibt sich die folgende Reihenfolge der Bewertungskriterien:

1. Pin-Anzahl
2. Dokumentation
3. Stromverbrauch

Der Stromverbrauch wird für diese Arbeit am stärksten gewichtet, da er die Funktionalität des Vorschaltgeräts beeinflusst, indem er mit dem zu ladenden Smartphone um den verfügbaren Strom konkurriert. Der Preis wird als zweitwichtigstes Kriterium gewertet, da er direkt mit den aufgestellten Anforderungen zusammenhängt. Dokumentation und Pin-Anzahl sind weniger wichtig, weil sie lediglich den Aufwand bei diesem Projekt beeinflussen, darüber hinaus aber kaum Auswirkungen zeigen.

In der Pugh-Matrix von Tabelle 4.4 wird ATtiny85 aufgrund des niedrigen Preises und des geringen Energieverbrauchs als Basislösung gewählt [20].

Tab. 4.4: Pugh-Matrix: Mikrocontrollern

Mikrocontroller / **Kriterien**	**PIC16LF1938**	**ATmega8L**	**ATtiny85**	**Gewichtung**
Pin-Anzahl	-	-	0	1
Dokumentation	-	0	0	2
Preis	-	-	0	3
Niedriger Stromverbrauch	+	-	0	4
Summe +	1	0	0	
Summe -	3	3	0	
Summe 0	0	1	0	
Gewichtete Summe +	4	0	0	
Gewichtete Summe -	6	8	0	

Der ATmega8L schneidet im Vergleich zum ATtiny85 im allen Aspekte schlechter oder gleich gut ab (gewichte Summe -8). Auch das Ergebnis des PIC16LF1938 ist schlechter (gewichte Summe -2). Daher wird für dieses Projekt der ATtiny85 als Mikrocontroller gewählt.

4.2 Hardware Architektur

Nachdem die benötigten Hardware-Komponenten beschrieben und die entsprechenden Modelle gewählt wurden, wird in diesem Abschnitt gezeigt, wie die Hardware-Komponenten miteinander verbunden sind. Dafür wird ein Blockdiagramm erstellt, auf dem die Verbindungen zwischen den Komponenten mit Pfeilen gekennzeichnet werden. Abbildung 4.4 zeigt das Blockdiagramm des Vorschaltgeräts.

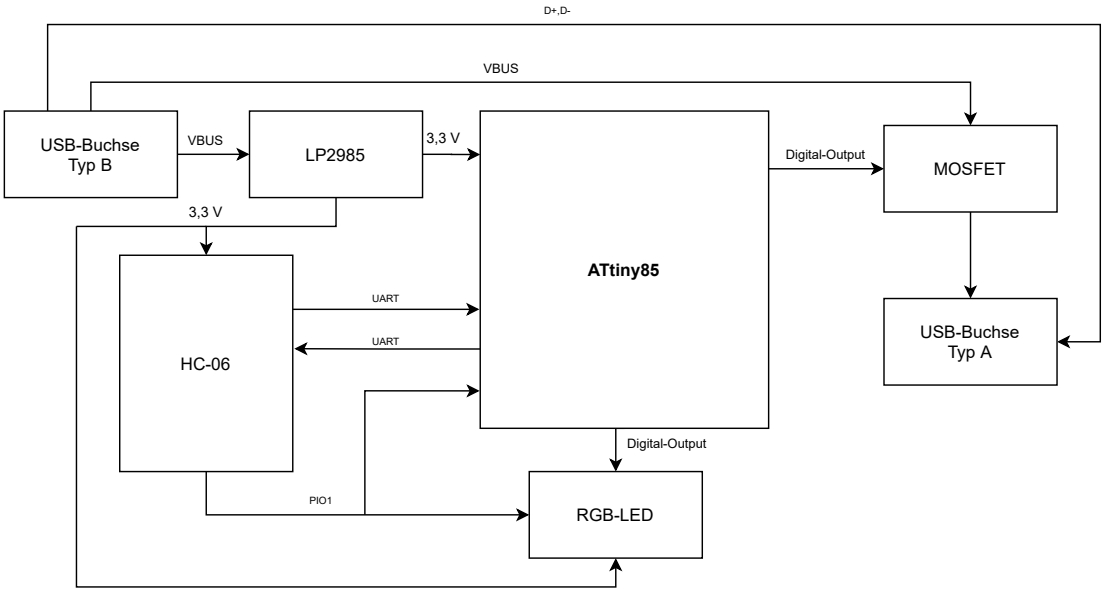


Abb. 4.4: Blockdiagramm der Hardware-Komponenten

Das Hardware-Design basiert auf den Anforderungen Req-8 und Req-12. Diese erfordern die Schaffung einer effizienten Architektur, die es ermöglicht, Größe und Kosten zu sparen. Dazu wurde u.a. die Verbindung zwischen dem Spannungsregler und dem Mikrocontroller so gestaltet, dass sie es erlaubt, den Mikrocontroller mit 3,3 V zu versorgen, um die Verwendung eines zusätzlichen Pegelwandlers zwischen Mikrocontroller und Bluetooth-Modul zu vermeiden. Außerdem wird die RGB-LED durch das HC-06-Modul angesteuert, um die Zustände „Ladevorgang aktiv“ und „Ladevorgang inaktiv“ zu signalisieren. Dadurch vereinfacht man die notwendige Software des Mikrocontrollers, der sonst vollständig für die Steuerung der RGB-LED verantwortlich wäre.

4.3 Softwarearchitektur und -design

Das folgende Unterkapitel beschäftigt sich damit, wie die Software für den Mikrocontroller erstellt wurde. Die Arbeit der Software besteht in der Interpretation der Anweisungen der Smartphone-Anwendung sowie in der Steuerung des MOSFETs und zum Teil der RGB-LED. Um die Funktionsweise der Software besser zu verstehen, wird die Software aus drei Perspektiven betrachtet [32]:

- Bausteinsicht:
  Hier wird die statische Struktur des Systems gezeigt. Die Software wird in Module unterteilt, von denen jedes eine spezifische Aufgabe und definierte Schnittstellen hat.

- Laufzeitsicht:
  Diese Sichtweise zeigt, wie die Softwarefunktionen ausgeführt werden. Dadurch wird eine klare Vorstellung vom Verhalten der Software vermittelt.

- Entwicklungssicht:
  Diese Perspektive zeigt den Prozess der Software-Generierung und ihre anschließende Übertragung auf den Mikrocontroller.

4.3.1 Bausteinsicht

Um eine gute Softwarestruktur zu schaffen, wird die Software in drei Module unterteilt, die in der Abbildung 4.5 mit ihren jeweiligen Verbindungen zueinander zu sehen sind.

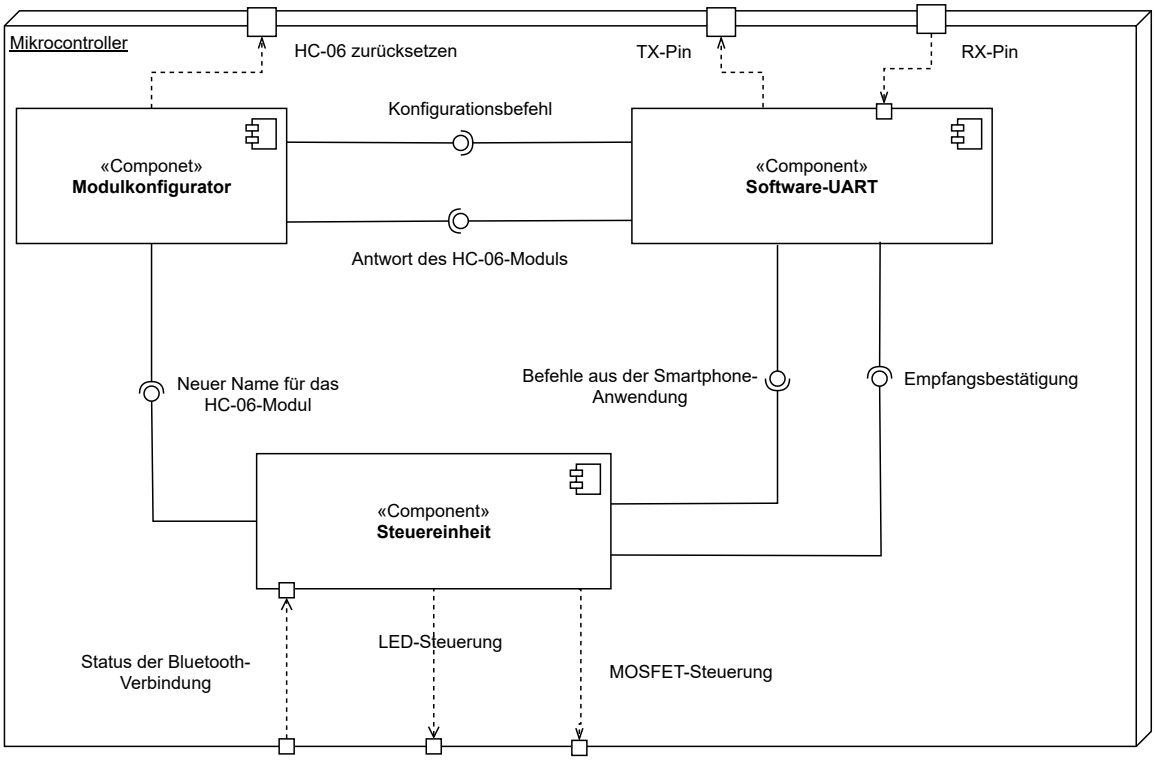


Abb. 4.5: Komponentendiagramm der Software

Bei diesen Modulen handelt es sich um Softwarekomponenten, die für die Ausführung einer bestimmten Aufgabe konzipiert sind. Das Diagramm veranschaulicht den Kommunikationsfluss zwischen den drei Modulen. Jedes Modul nutzt die Peripheriegeräte und Pins des Mikrocontrollers, um seine Aufgabe zu erfüllen. Tabelle 4.5 enthält die Kurzbeschreibungen der einzelnen Module.

Tab. 4.5: Überblick über die Software-Module des Vorschaltgeräts

Modul	Kurzbeschreibung
Modulkonfigurator	Beinhaltet den Konfigurationsprozess des Bluetooth-Moduls
Software-UART	Realisiert das UART-Kommunikationsprotokoll
Steuereinheit	Enthält den RGB-LED- und MOSFET-Steuerungsprozess

In den folgenden Abschnitten wird ausführlich erklärt, wie die einzelnen Module funktionieren und welche Schnittstellen ihnen zur Verfügung stehen.

Modulkonfigurator

Verantwortlichkeit Die HC-06-Bluetooth-Kommunikationsmodule sind ab Werk mit einem Namen versehen. Die Hauptaufgabe des Modulkonfigurators besteht darin, diesen auf den Standardnamen „JILZ-V01“ zu ändern. Dieser ist für den Benutzer bei der Suche nach dem Bluetooth-Gerät sichtbar. Das Modul bietet ebenfalls die Möglichkeit, diesen Standardnamen zu ändern.

Schnittstellen Das Modul bietet eine Funktion zur Konfiguration des Bluetooth-Moduls und eine zur Änderung des Standardnamens. Die Abbildung 4.6 zeigt den Aufbau dieser Funktionen.

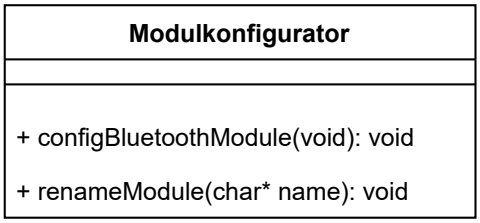


Abb. 4.6: Blackbox-Ansicht des Modulkonfigurator-Moduls

Tabelle 4.6 enthält eine detaillierte Erklärung dieser Funktionen.

Tab. 4.6: Beschreibung der Funktionen des Modulkonfigurator-Moduls

Funktion	Kurzbeschreibung
configBluetoothModule	Stellt sicher, dass das Bluetooth-Modul einen Namen sowie eine PIN hat. Der Name kann der Standardname „JILZ-V01“ oder der Name sein, der beim Ausführen der Funktion renameModule gespeichert wurde. Bei der PIN wird nur die Standardkombination „1234“ konfiguriert.
renameModule	Erlaubt die Änderung des Standardnamens.

Tabelle 4.6 enthält eine detaillierte Erklärung dieser Funktionen.

Software-UART

Verantwortlichkeit Dieses Modul ist für die Implementierung des UART-Kommunikationsprotokolls zuständig. Dies ist notwendig, um Daten vom Bluetooth-Modul zu empfangen und zu senden.

Schnittstellen Die Funktionen, die das Modul zur Verfügung stellt, sind in Abbildung 4.7 dargestellt.

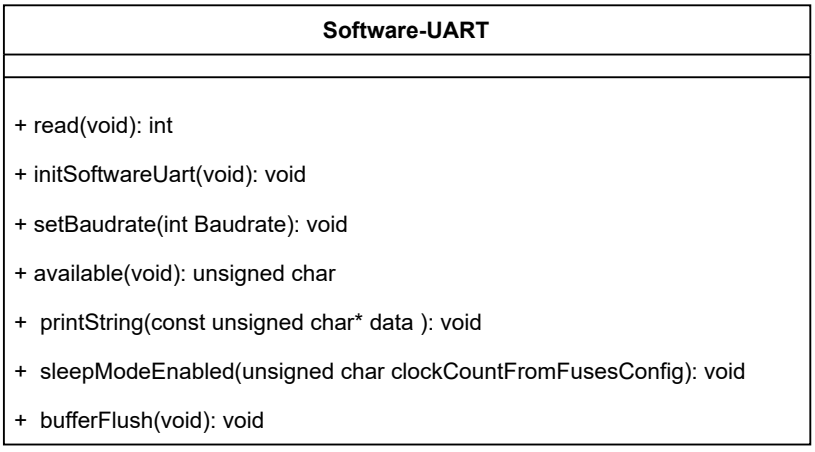


Abb. 4.7: Blackbox-Ansicht des Software-UART-Moduls

Tabelle 4.7 enthält relevante Erläuterungen zu den Funktionen, die ein besseres Verständnis der Funktionsweise des Moduls ermöglichen.

Tab. 4.7: Beschreibung der Funktionen des Software-UART-Moduls

Funktion	Kurzbeschreibung
read	Gibt ein von UART angekommenes Zeichen zurück. Falls kein Zeichen vorhanden ist, gibt die Funktion eine negative Zahl zurück.
initSoftwareUart	Aktiviert die notwendige Mikrocontroller-Peripherie.
setBaudrate	Stellt die Baudrate ein. In diesem Fall wird die Standard-Baudrate des HC-06 verwendet, die 9600 entspricht.
available	Liefert die Anzahl der ungelesenen Zeichen.
printString	Sendet eine Zeichenfolge mit dem UART-Protokoll.
sleepModeEnabled	Verringert den anfänglichen Timer-Zähler, da der Mikrocontroller in einen Sleep-Zustand gesendet wird.
bufferFlush	Löscht alle Zeichen, die empfangen wurden.

Steuereinheit

Verantwortlichkeit Dieses Modul interpretiert die über Bluetooth empfangenen Telegramme, um später zu beurteilen, welche Maßnahmen zu ergreifen sind. Die Steuerung der Hardware-Komponenten variiert je nach empfangenem Telegramm.

Schnittstellen Die von diesem Modul angebotene Funktion ist in Abbildung 4.8 dargestellt.

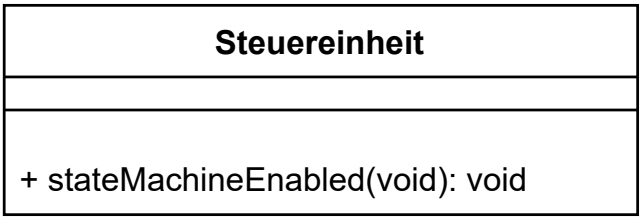


Abb. 4.8: Blackbox-Ansicht des Steuereinheit-Moduls

Die Funktion stateMachineEnabled aktiviert den zustandsbasierten Steuerungsprozess.

4.3.2 Laufzeitsicht

In diesem Abschnitt wird das Zusammenspiel der Module zur Durchführung der drei, im Folgenden erläuterten Aufgaben gezeigt. Diese Aufgaben sind:

1. Konfiguration des HC-06-Moduls
2. Empfangen und Senden über das UART-Kommunikationsprotokoll
3. Interpretation von empfangenen Telegrammen

Konfiguration des HC-06-Moduls

Die Aufgabe der Konfiguration des HC-06-Moduls besteht darin, das HC-06-Modul zu benennen, damit der Benutzer das Vorschaltgerät unter möglichen anderen Bluetooth-Geräten leicht erkennen kann.

Um die Kommunikation zwischen den Modulen darzustellen, wird ein UML-Sequenzdiagramm verwendet. In Abbildung 4.9 befindet sich das Sequenzdiagramm, in dem das Verfahren zur Namensvergabe des HC-06-Moduls gezeigt wird kann [14].

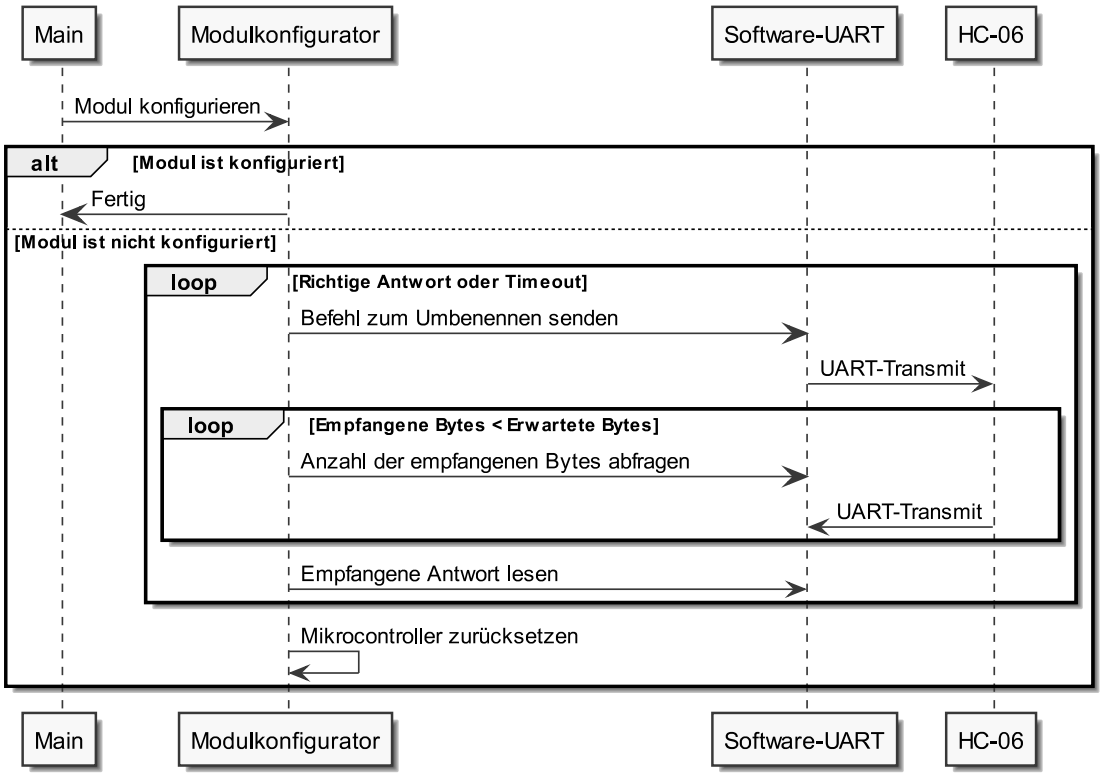


Abb. 4.9: Zusammenspiel der Module zur Konfiguration des Bluetooth-Moduls

Das Diagramm zeigt den Prozess aus der Sicht der Main-Funktion. Main ist für die Ausführung des Programms verantwortlich. Das Modul HC-06 bietet eine Reihe von Befehlen, die als „AT-Befehle“ bezeichnet werden und es ermöglichen, seine Eigenschaften zu verändern [9]. Im Falle der Namensänderung wäre der entsprechende AT-Befehl „AT+NAMEJILZ-V01“, wobei „JILZ-V01“ den gewünschten Namen darstellt.

Nach dem Senden dieses Befehls, stellt das Modulkonfigurator-Modul sicher, dass die Bestätigung des HC-06 korrekt. Gleichzeitig wird vermieden, dass die Software in eine Endlosschleife gerät. Dies geschieht, indem der Watchdog-Timer des Mikrocontrollers verwendet wird. Eine Zeitüberschreitung verursacht einen Neustart [20].

Damit das HC-06-Modul nur einmal konfiguriert werden muss, ist es wichtig, zu erkennen, ob das Modul bereits konfiguriert wurde oder nicht. Um dieses Ziel zu erreichen, ist ein nichtflüchtiger Speicher erforderlich. Der ATtiny-85 bietet zwei Arten von nichtflüchtigem Speicher, den FLASH-Speicher und den EEPROM-Speicher. Die Verwendung

des EEPROM-Speichers ist die praktikabelste Methode, da der Flash-Speicher eine vollständige Sektorlöschung erfordert, um neue Daten zu schreiben [20].

Empfangen und Senden über das UART-Kommunikationsprotokoll

Die Aufgabe, über das UART-Kommunikationsprotokoll Daten zu empfangen und zu senden, besteht darin, die UART-Kommunikation mithilfe von Software umzusetzen, sodass der Mikrocontroller die Möglichkeit hat, mit dem HC-06-Modul zu kommunizieren.

Zunächst werden die von dem HC-06-Modul empfangenen Daten analysiert. Die Software muss dazu den Frame des UART-Kommunikationsprotokolls interpretieren, um das übertragene Byte zu erkennen. Der erste Schritt ist die Identifizierung des Startbits, das den Beginn der Übertragung anzeigt. Dann muss die Software den Timer des Mikrocontrollers so einstellen, dass eine Unterbrechungsanfrage (englisch Interrupt Request, IRQ) nach der Zeit, die benötigt wird, um ein Bit zu identifizieren, erfolgt. Das Bild 4.10 zeigt, wie dies umgesetzt wird.

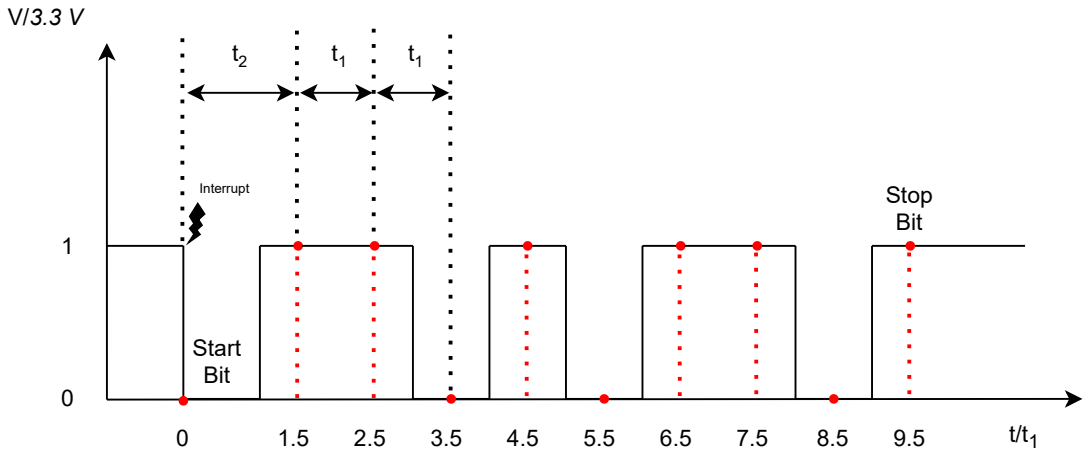


Abb. 4.10: Empfangsprozess eines Bytes per UART

Der Empfangs-Pin des Mikrocontrollers wird mit dem Rx-Pin des HC-06-Moduls verbunden. Das HC-06-Modul hält diesen Pin auf $3,3\,V$, bis Daten gesendet werden. Wenn das Modul mit dem Senden von Daten beginnt, senkt es als erstes den Spannungspegel auf $0\,V$ ab, um das Start-Bit zu erzeugen. Die Software des Mikrocontrollers erkennt die Spannungsänderung des Pins und erzeugt einen IRQ, dann muss die Software den Timer

des Mikrocontrollers mit einer Zeit entsprechend t_2 einstellen. Wenn die Zeit erreicht ist, liest die Software den Spannungswert des Pins und interpretiert ihn binär, d.h. $3,3\,V$ wird als „1“ interpretiert und $0\,V$ als „0“. Sobald das erste Bit erkannt wird, wird der Timer auf t_1 gesetzt. Der Timer bleibt konstant auf t_1, bis das Byte vollständig empfangen wurde. [19]

Sowohl die Zeit t_1 als auch die Zeit t_2 hängen von der verwendeten Baudrate ab. Das HC-06-Modul verwendet eine Standard-Baudrate von 9600. Für die Berechnung von t_1 und t_2 ist es notwendig, die Übertragungszeit eines Bits (t_{Bit}) zu kennen. t_1 wird als t_{bit} festgelegt, um jedes einzelne Bit messen zu können. Es wird wie folgt berechnet [19]:

$$\begin{aligned} t_{Bit} &= \frac{1}{Baudrate}\,Bit \\ t_{Bit} &= \frac{1}{9600\,\frac{Bit}{s}}\,Bit \\ t_{Bit} &= 104,167\,\mu s \end{aligned} \tag{4.1}$$

Damit die Messung möglichst wenig fehleranfällig ist, wird nach der halben Übertragungszeit eines Bits gemessen. Um diesen Zeitpunkt für das erste den Bit nach dem Starbit zu erreichen, ist $t_2 = t_{bit} \cdot \frac{3}{2}$.

Also gelten die folgenden Beziehungen:

$$t_1 = t_{Bit} \tag{4.2}$$

$$t_2 = \frac{3}{2} \cdot t_{Bit} \tag{4.3}$$

Das Senden von Daten durch die Software des Mikrocontrollers ist einfacher als der Empfang, da die Zeit des Timers nicht variiert. Abbildung 4.11 zeigt, wie die Übertragung eines Bytes funktioniert.

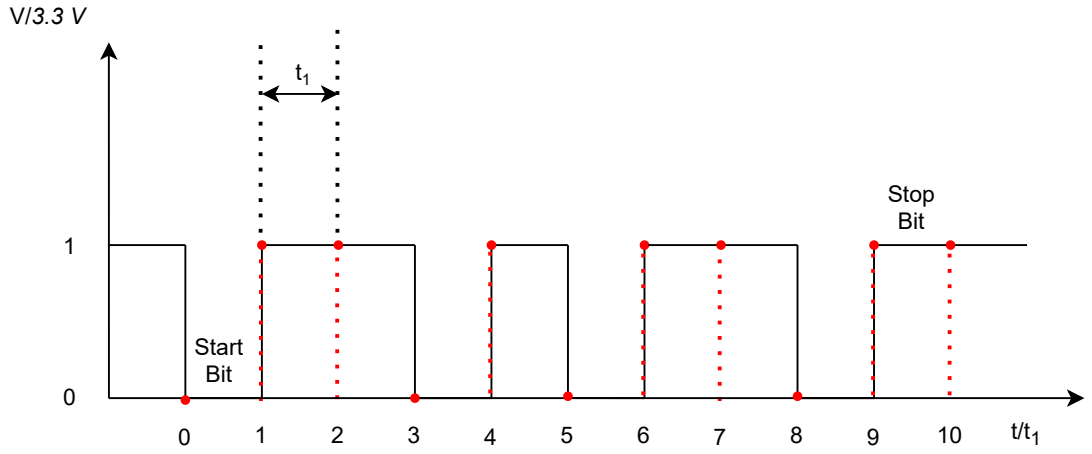


Abb. 4.11: Sendeprozess eines Bytes per UART

Der Übertragungsprozess erfolgt über einen Mikrocontroller-Pin, dieser Pin wird mit dem TX-Pin des HC-06-Moduls verbunden. Die Software hält diesen Pin zunächst auf $3,3\,V$ und senkt die Spannung beim Start der Übertragung auf $0\,V$ ab, um das Start-Bit zu erzeugen. Danach wird der Timer auf die Zeit t_1 eingestellt. Jedes Mal, wenn der Timer diese Zeit erreicht, nimmt der Pin den Wert des zu übertragenden Bits an, wobei 1 gleich 3,3 Volt und 0 gleich 0 Volt ist. Der Vorgang wird so lange wiederholt, bis alle Bits gesendet sind, beginnend vom Least Significant Bit (LSB) , bis zum Most Significant Bit (MSB) und dann schließlich das Stop-Bit. [19, 38]

Die Software modelliert den Sende- und Empfangsprozess durch einen Zustandsautomat mit folgenden Zuständen:

- Idle
- Transmit
- Transmit Stop Bit
- Receive

Abbildung 4.12 zeigt eine grafische Darstellung des Zustandsautomaten. Hierfür wird ein UML-Zustandsdiagramm verwendet [14].

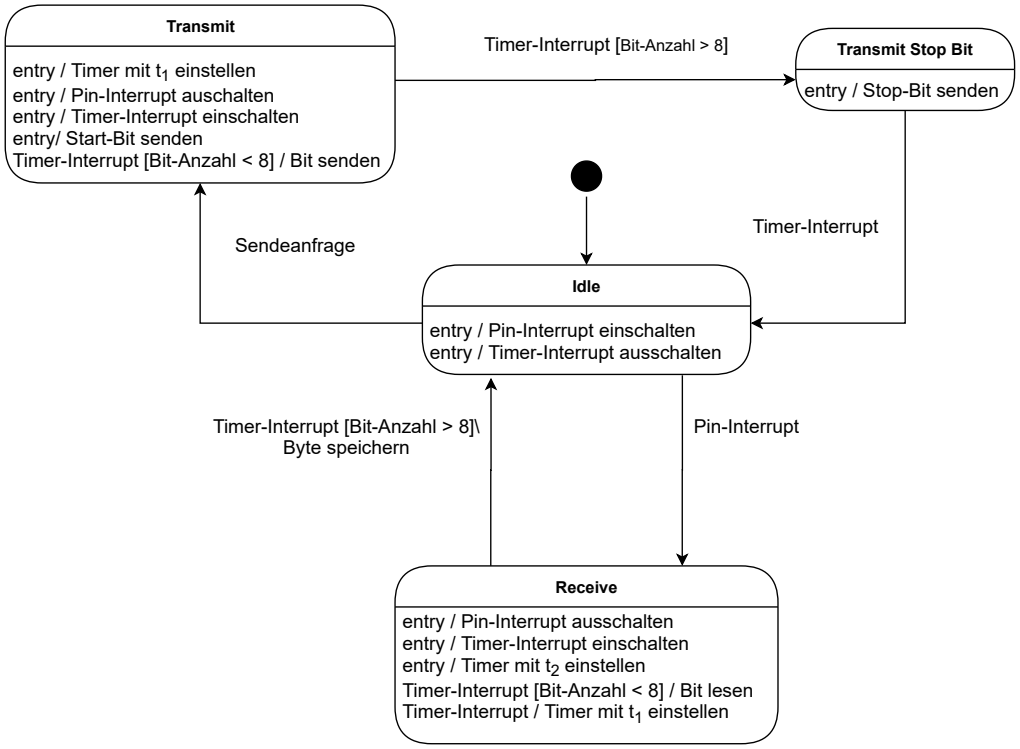


Abb. 4.12: Zustandsdiagramm zur Erzeugung des UART-Kommunikationsprotokolls

Der Zustandsautomat erzeugt das in Abbildung 4.11 und in Abbildung 4.10 gezeigte Verhalten. Der Automat arbeitet mit zwei Arten von Interrupts: Der erste ist der Timer-Interrupt, der ausgelöst wird, wenn eine bestimmte Zeit erreicht ist. Der zweite ist ein Pin-Interrupt, der es ermöglicht, eine fallende Flanke zu identifizieren. Es ist außerdem festzuhalten, dass die Kommunikation im Halbduplex-Verfahren erfolgt, da Empfang und Übertragung nicht gleichzeitig erfolgen können.

Interpretation der empfangenen Telegramme

Um das Design der Software zu erleichtern, werden Zustände zur Steuerung der Hardware-Komponenten wie folgt definiert:

- Aktiver Ladevorgang:
  In diesem Zustand muss der MOSFET angesteuert werden und die RGB-LED darf nur vom HC-06-Modul gesteuert werden.

- Inaktiver Ladevorgang:
  Der MOSFET muss den Stromfluss zum Smartphone blockieren und die RGB-LED sollte aufleuchten, damit der Benutzer sieht, dass das Smartphone über genügend Akkuladung verfügt.

Die Abbildung 4.13 zeigt das UML-Zustandsdiagramm, in dem man erkennen kann, wie die Zustandsänderungen auftreten.

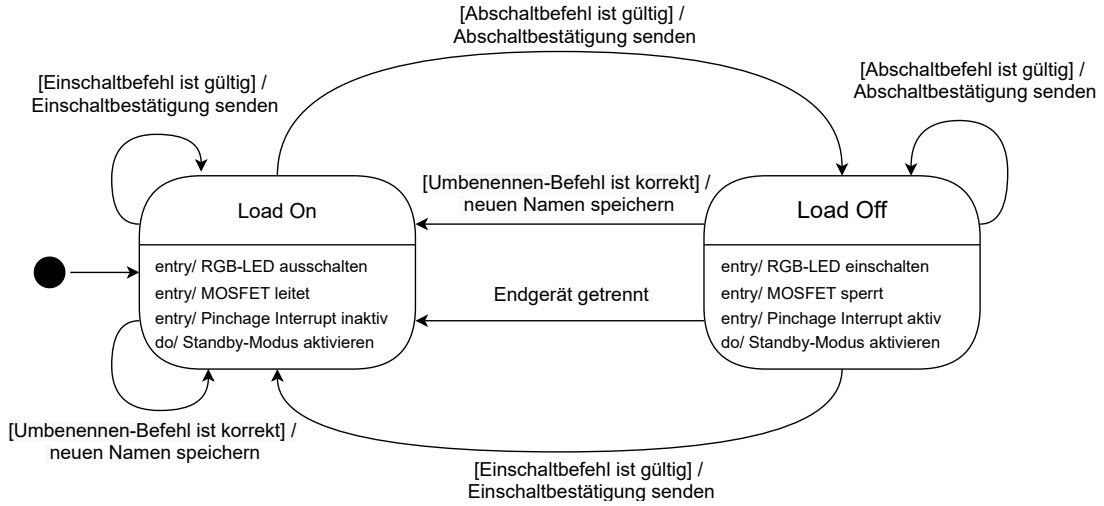


Abb. 4.13: Zustandsdiagramm zur Steuerung der Hardware-Komponenten

Die Software interpretiert die über Bluetooth empfangenen Befehle, um zu prüfen, ob es eine Zustandsänderung gibt. Damit dies möglich ist, muss das Vorschaltgerät an ein Endgerät angeschlossen sein. Falls ein korrekter Abschaltbefehl erfolgt, wechselt der Zustand zu „Load Off“, wobei die Stromzufuhr zum Smartphone unterbrochen wird. In diesem Zustand muss die Software den Status der Bluetooth-Verbindung überprüfen. Falls diese verloren geht, kehrt sie in den Zustand „Load On“ zurück. Um diese Aufgabe zu erfüllen, muss der PIO1-Pin des HC-06-Moduls überwacht werden, der den Status der Bluetooth-Verbindung anzeigt. Wenn keine Bluetooth-Verbindung besteht, wechselt das Signal zwischen $0\,V$ und $3,3\,V$. Wenn das Modul mit einem Gerät verbunden ist, besteht ein konstantes Signal von $3,3\,V$. Die Software muss den Pinchage-Interrupt so konfigurieren, dass es möglich ist, nach einer erfolgreichen Verbindung den Wechsel auf $0\,V$ zu identifizieren, damit der Zustand von „Load Off“ auf „Load On“ geändert wird. In beiden Zuständen geht der Mikrocontroller in einen Standby-Modus über, der nur durch

einen Interrupt beendet werden kann. Dies hilft, unerwünschtes Verhalten zu vermeiden und trägt gleichzeitig zur Energieeinsparung bei.

4.3.3 Entwicklungssicht

In diesem Abschnitt wird der Erstellungsprozess der Software dargestellt. Der erste Schritt besteht darin, die Elemente für die Programmierung eines Mikrocontrollers zu identifizieren [30]. Im Falle dieser Arbeit werden folgende Elemente benötigt:

- Texteditor
- Compiler
- Linker
- Programmiersoftware:
  Die Programmiersoftware ermöglicht die Übertragung des Programmcodes in den Flash-Speicher des Mikrocontrollers.
- Programmierhardware:
  Die Programmierhardware dient als physikalische Schnittstelle zwischen Mikrocontroller und Programmiersoftware.

Für diese Arbeit wurde eine Entwicklungsumgebung verwendet, die den Texteditor, den Compiler und den Linker in einem einzigen Element vereint und mit dem ATtiny85-Mikrocontroller kompatibel ist. Aufgrund des leistungsfähigen Simulators wurde die integrierte Entwicklungsumgebung („integrated development environment“, IDE) der Firma „IAR Systems“ gewählt [12]. Dieses verfügt über eine kostenlose KickStart-Lizenz, die eine begrenzte Codegröße von 4 Kilobyte ermöglicht [11]. Daher darf der Code diese Grenze nicht überschreiten.

Die IDE erzeugt aus dem in den Modulen vorhandenen Code eine Hex-Datei, in diesem Fall BA_Lozano.hex. Die Software „avrdude“ wird zum Herunterladen und Hochladen vom und auf den Flash-Speicher bzw. EEPROM des AVR-Mikrocontrollers verwendet [42]. Sie schreibt die Datei BA_Lozano.hex in den Flash-Speicher. Dazu nutzt „avrdude“ die USBasp-Hardware zur Verbindung über ein In-System-Programming Interface (ISP), wobei die MOSI-, MISO-, SCK- und RESET-Pins des ATtiny85 zum Hochladen des Programms verwendet werden [8, 34]. In Abbildung 4.14 wird dieser Prozess grafisch dargestellt.

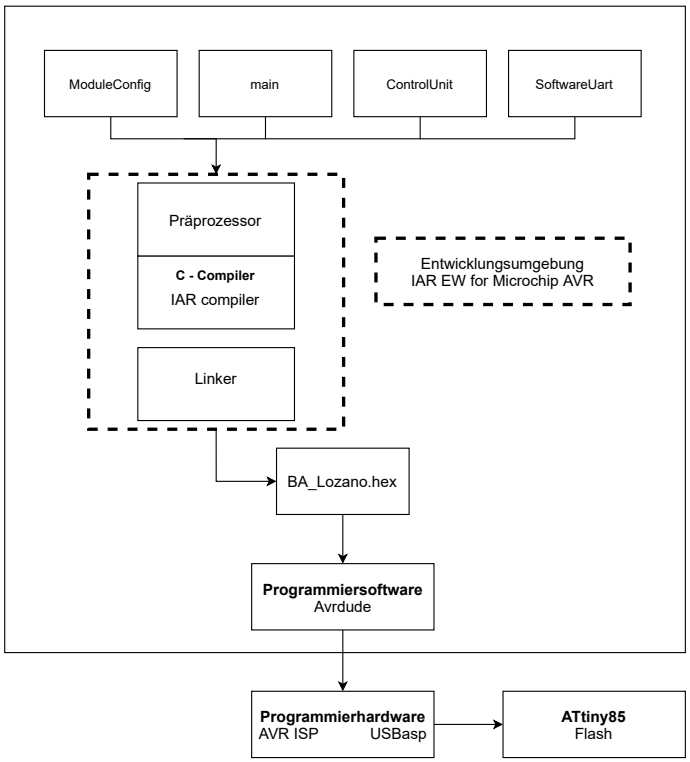


Abb. 4.14: Blockdiagramm der Softwareentwicklung für den ATtiny85

4.3.4 Konventionen

In diesem Abschnitt werden die Konventionen erwähnt, die bei der Implementierung der Software berücksichtigt werden.

Dokumentation

Das Tool Doxygen wird zur Generierung der Dokumentation der Software verwendet. Doxygen generiert eine Latex- sowie eine Html-Dokumentation [39]. Diese befinden sich in Abschnitt A.3 dieser Arbeit.

Sprache

Kommentare zum Code werden auf Englisch geschrieben, um die Software für möglichst viele verständlich zu gestalten. Aus dem selben Grund ist die von Doxygen erstellte Dokumentation ebenfalls auf Englisch.

Namenskonventionen

Die Verwendung einer Namenskonvention zielt darauf ab, die Verständlichkeit der Software zu verbessern. Tabelle 4.8 zeigt die Namenskonventionen, an denen sich die Software orientiert [10].

Tab. 4.8: Namenskonventionen für die Software-Implementierung [10]

Kategorie	**Notation**	**Beispiel**
Variable	Snake case	error_count
Funktion	Camel case	setBaudrate
Makro	Uppercase	CHANGE_NAME_COMMAND
Aufzählungstyp	Pascal case	UartStates
Datenstrukturen	Pascal case	BaudrateConfig

5 Umsetzung und Entwicklung

In diesem Kapitel wird die Entwicklung des Vorschaltgeräts, die auf den in Kapitel 4 getroffenen Entscheidungen basiert, beschrieben. Zunächst wird auf die Implementierung der Software für den ATtiny85-Mikrocontroller eingegangen und schließlich wird die Hardware des Vorschaltgeräts erläutert.

5.1 Entwicklung der Software

Dieses Kapitel beschreibt die Entwicklung der Software für das Vorschaltgerät. Es ist nach den drei Modulen der Software: Modulkonfigurator, Software-UART und Steuereinheit gegliedert, auf die in den jeweiligen Unterkapiteln näher eingegangen wird.

5.1.1 Software-UART

Zuerst wird die Entwicklung des Moduls Software-UART geschildert. Das Modul besteht aus den Funktionen „read“, „initSoftwareUart“, „setBaudrate“, „available“, „printString“, „sleepModeEnabled“ und „flushBuffer“.

Der erste Schritt bei der Erstellung dieses Moduls ist die Definition der „initSoftwareUart“ Funktion, die für die Initialisierung der Mikrocontroller-Pins, des Timers und des Interrupt-Pins zuständig ist. Das Aktivitätsdiagramm in Abbildung 5.1 zeigt den Ablauf dieser Funktion.

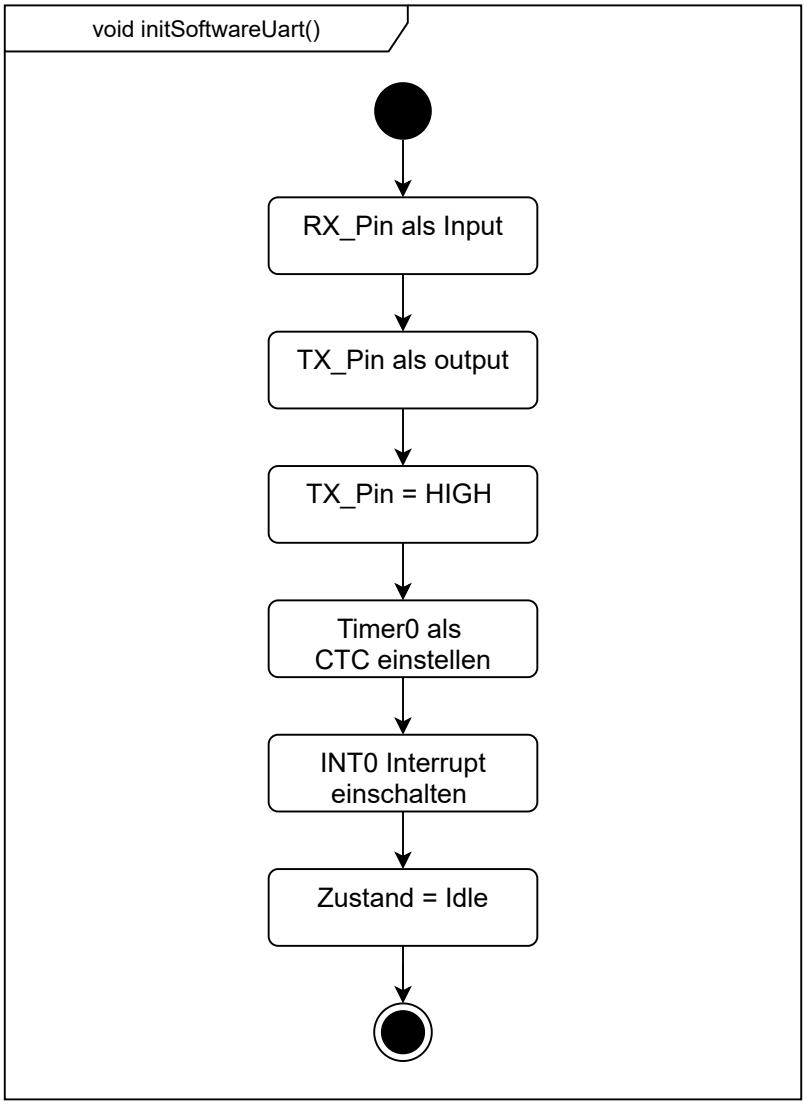


Abb. 5.1: Aktivitätsdiagramm der Funktion „initSoftwareUart“

Wie sich aus der Grafik ablesen lässt, konfiguriert die Funktion zuerst den Mikrocontroller-Pin RX_PIN als Eingang und den TX_PIN als Ausgang. Die TX-Leitung muss sich im Zustand HIGH befinden, damit der Empfänger das Start-Bit der seriellen Kommunikation erkennt, wenn die Übertragung beginnt. Deswegen setzt die Funktion „initSoftwareUart“ den TX_PIN auf HIGH. Dann wird der Timer0 auf den Clear Timer on Compare Modus (CTC) eingestellt, was bedeutet, dass der Timer nach Erreichen des eingestellten Grenzwertes zurückgesetzt wird. Der Interrupt INT0 wird benötigt, um die fallende

Flanke der RX-Leitung, die vom HC-06 ausgeht, zu erkennen. Er ist so eingestellt, dass es möglich ist, den Mikrocontroller aus dem Standby-Modus aufzuwecken.

Die Funktion „printString“ lässt sich wie folgt darstellen:

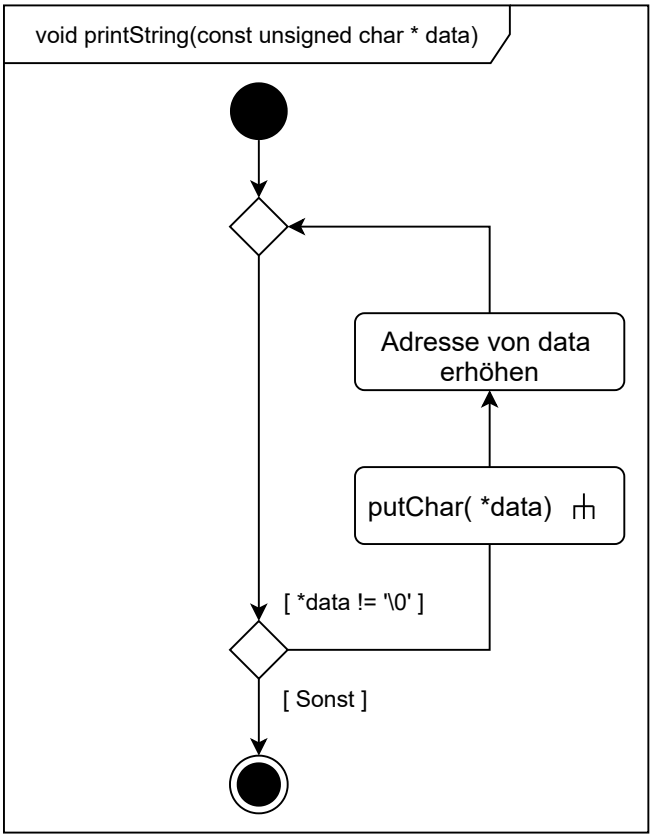


Abb. 5.2: Aktivitätsdiagramm der Funktion „printString“

Um Daten mit der Funktion „printString“ zu übertragen, muss das Senden eines Bytes möglich sein. Zu diesem Zweck wurde die Funktion „putChar“ entwickelt.

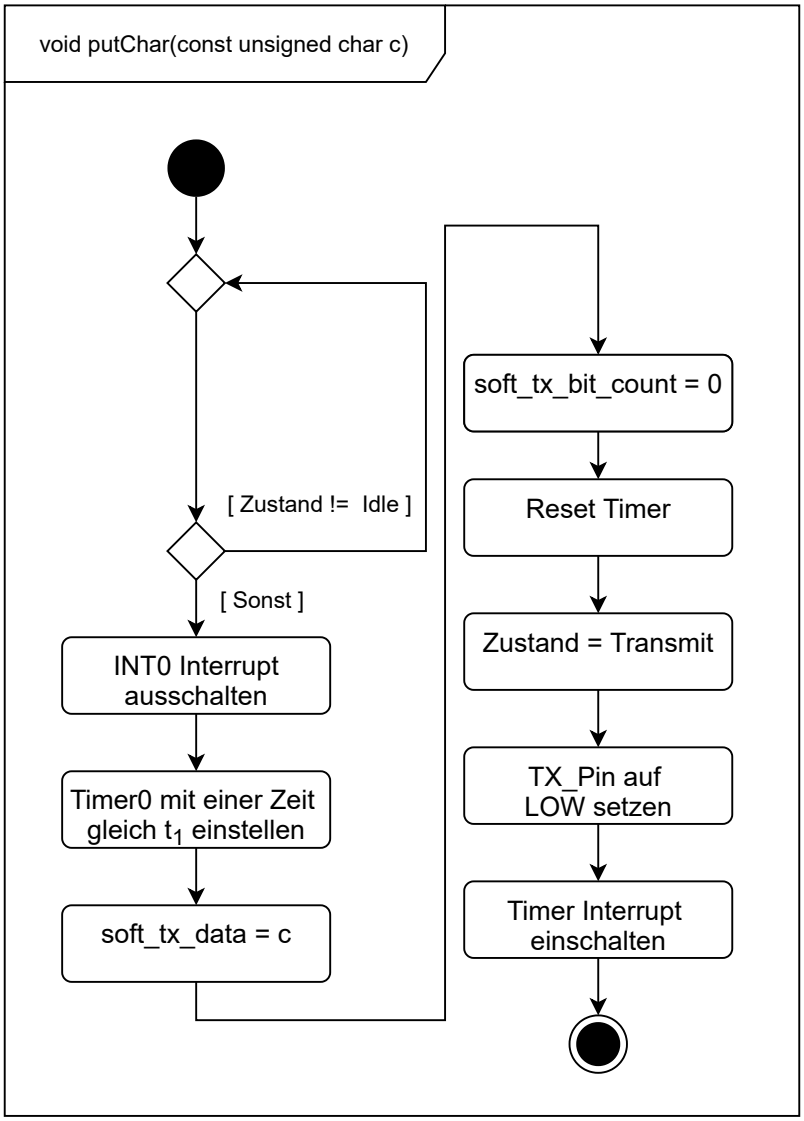


Abb. 5.3: Aktivitätsdiagramm der Funktion „putChar“

Abbildung 5.3 zeigt das Aktivitätsdiagramm dieser Funktion. Der Ablauf ist wie folgt: Zuerst wird gewartet, bis der Zustand „Idle“ erreicht ist, um sicherzustellen, dass es keine Überschneidung der Sende- und Empfangsprozesse gibt. Im nächsten Schritt werden „INT0 Interrupt ausschalten“ und „Timer0“ mit einer Zeit gleich t_1 einstellen ausgeführt. Diese sind die Eingangsbedingungen für den Zustand „Transmit“ des Zustandsautomats von Abbildung 4.12. Der Timer misst die Zeit mit den Zyklen seiner internen Uhr. Die Zeit t_1 wurde in Gleichung (4.2) berechnet und beträgt $104\,\mu s$. Der interne Takt des

Mikrocontrollers läuft mit einer Frequenz von $1\,MHz$. Es muss also die Anzahl der Takte ($n_{Takte\,t1}$) gefunden werden, die der Zeit t_1 entspricht.

$$\begin{aligned} T_{Clock} &= \frac{1}{f_{Clock}} \\ T_{Clock} &= 1\,\mu s \\ n_{Takte\,t1} &= \frac{t_1}{T_{Clock}} \\ n_{Takte\,t1} &= 104 \end{aligned} \tag{5.1}$$

Gleichung (5.1) ergibt eine Taktanzahl von 104, um t_1 zu erreichen. Anschließend wird das zu sendende Byte in der „sodt_tx_data“ Variable und die Anzahl der gesendeten Bits in der „soft_tx_bit_count“ Variable gespeichert. Der Timer wird zurückgesetzt und der Zustand wechselt von Idle zu Transmit. Schließlich wird der TX_Pin auf LOW gesetzt und der Timer Interrupt eingeschaltet, damit er nach 104 Takten einen Interrupt einleitet.

Wenn die Funktion „putChar“ endet, beginnt der Zustandsautomat aus Abbildung 4.12 zu arbeiten, indem er bei jedem Interrupt des Timer0 ein Bit sendet. Das Senden einer Zeichenfolge ist ein mehrfaches Senden von Zeichen, wie im Aktivitätsdiagramm aus Abbildung 5.2 zu sehen ist. Es werden Zeichen so lange gesendet, bis der Zeiger auf Null steht, was bedeutet, dass die Zeichenfolge beendet ist.

Die Funktion „read“ ist für den Datenempfang verantwortlich. Der Datenempfang beginnt, wenn eine fallende Flanke erkannt wird, wodurch der INT0-Interrupt ausgelöst wird. Die Aktionen, die in der Interrupt-Service-Routine (ISR) auftreten, werden im Aktivitätsdiagramm von Abbildung 5.4 beschrieben.

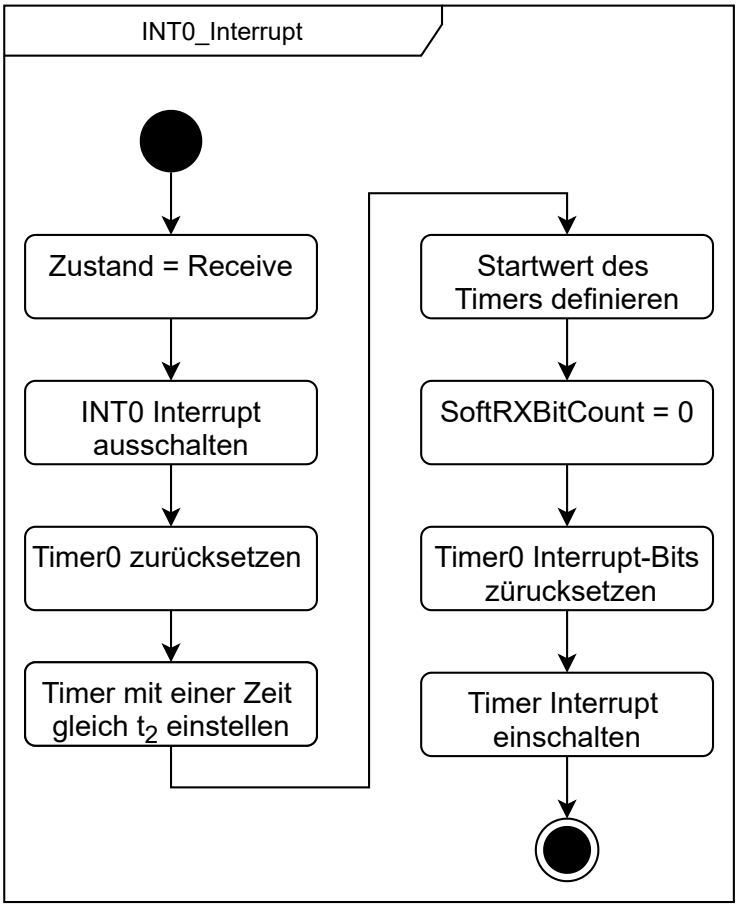


Abb. 5.4: Aktivitätsdiagramm der ISR vom INT0-Interrupt

Der Prozess beginnt mit den Eingangsbedingungen des Zustands „Receive“ des Zustandsautomatens der Abbildung 4.12: „Pin-Interrupt ausschalten“, „Timer-Interrupt einschalten“ und „Timer mit t2 einstellen“. Es ist zu beachten, dass der Timer0 zurückgesetzt werden muss, um die Zählzeit einzustellen. In diesem Fall ist die Zeit, die der Timer zählen soll, gleich t_2. Diese wurde in Gleichung (4.3) mit $t_2 = 1,5 \cdot t_1$ beschrieben. Die Berechnung der Takte ($n_{Takte\,t2}$) zum Erreichen der Zeit t_2 wird wie folgt durchgeführt:

$$\begin{aligned} T_{Clock} &= 1\,\mu s \\ n_{Takte\,t2} &= \frac{1.5 \cdot 104\,\mu s}{1\,\mu s} \\ n_{Takte\,t2} &= 156 \end{aligned} \tag{5.2}$$

Der Timer-Zähler muss jedoch mit einer Anfangszeit gestartet werden, die alle in der ISR des Timers und der ISR des INT0-Interrrupts verlorene Zeit einbezieht. Diese Zeit wird mit dem von der Entwicklungsumgebung „IAR EW“ angebotenen Simulator ermittelt. Mit diesem Simulator lässt sich die Anzahl der von einer ISR benötigten Taktzyklen ermitteln. Die Simulation ergibt in diesem Fall eine Gesamtzahl von 108 Taktzyklen ($n_{Takte\ ISR}$) für beide ISR. Die Zeit (t_3) für das Auftreten eines Timer-Interrupt und die entsprechende Anzahl von Takten ($n_{Takte\ t3}$) werden wie folgt berechnet:

$$\begin{aligned} n_{Takte\ t3} &= n_{Takte\ t2} - n_{Takte\ ISR} \\ n_{Takte\ t3} &= 156 - 108 \\ n_{Takte\ t3} &= 48 \end{aligned} \tag{5.3}$$

$$\begin{aligned} t_3 &= n_{Takte\ t3} \cdot T_{Clock} \\ t_3 &= 48 \cdot 1\,\mu s \\ t_3 &= 48\,\mu s \end{aligned} \tag{5.4}$$

Gleichung (5.3) gilt nicht, wenn sich der Mikrocontroller im Standby-Modus befindet: Der ATtiny85 verfügt über einen „Power-down“-Modus, in dem der interne Oszillator abgeschaltet wird, um Energie zu sparen. Der Mikrocontroller kann durch den INT0-Interrupt wieder „geweckt“ werden. Um wieder in den Betriebsmodus zu gelangen, benötigt er eine definierte Anzahl von Takten. Diese beträgt laut Datenblatt 6 ($n_{Takte\ sleep}$) Taktzyklen [20].

Die Funktion „sleepModeEnabled“ muss aufgerufen werden, bevor der Mikrocontroller in einen Sleep-Modus übergeht. Sie hat die Aufgabe, die Anzahl der Zyklen von $n_{Takte\ ISR}$ zu erhöhen, um beim Erwachen die richtige Zeitmessung fortsetzen zu können. Wenn der Mikrocontroller aufwacht, sehen nach dem Aufrufen der Funktion „sleepModeEnabled“

die Werte von t_3 und $n_{Takte\,t3}$ wie folgt aus:

$$\begin{aligned} n_{Takte\,ISR} &= 108 + n_{Takte\,sleep} \\ n_{Takte\,ISR} &= 108 + 6 \\ n_{Takte\,ISR} &= 114 \\ n_{Takte\,t3} &= n_{Takte\,t2} - n_{Takte\,ISR} \\ n_{Takte\,t3} &= 156 - 114 \\ n_{Takte\,t3} &= 42 \end{aligned} \tag{5.5}$$

$$\begin{aligned} t_3 &= n_{Takte\,t3} \cdot T_{Clock} \\ t_3 &= 42 \cdot 1\,\mu s \\ t_3 &= 42\,\mu s \end{aligned} \tag{5.6}$$

Die Funktion „setBaudrate“ initialisiert die Werte von $n_{Takte\,t1}$ und $n_{Takte\,t2}$. Wie aus Gleichung (4.1) ersichtlich wird, hängen $n_{Takte\,t2}$ und $n_{Takte\,t1}$ von der verwendeten Baudrate ab. Die Funktion ßetBaudrate“ akzeptiert nur eine Baudrate von 9600, wodurch $n_{Takte\,t1}$ und $n_{Takte\,t2}$ einen Wert von 104 bzw. 156 annehmen. Die Baudrate von 9600 ist der Standardwert für das HC-06-Modul und wird auch von den meisten anderen Bluetooth-Modulen unterstützt, weswegen sie für dieses Projekt gewählt wurde. Es besteht die Möglichkeit, weitere Baudraten zu implementieren, dies wird hier jedoch nicht näher betrachtet.

Wenn ein Byte vom UART empfangen wird, muss die Software dieses Byte in einem Puffer speichern, damit es später gelesen werden kann. Dieser Prozess wird durch einen „Circular buffer“ umgesetzt. Der „Circular buffer“ basiert auf einer FIFO-Datenstruktur (First In - First Out), bei der die Speicher ringförmig angeordnet sind [2]. Es gibt zwei Indizes: einen „Head“-Index, der die Position angibt, auf die das nächste Byte geschrieben werden muss, und einen „Tail“-Index, der die Position des Bytes angibt, das zuerst gelesen werden muss. Die Indizes werden auf Null zurückgesetzt, wenn die Pufferlänge erreicht ist. Abbildung 5.5 stellt diesen Prozess grafisch dar.

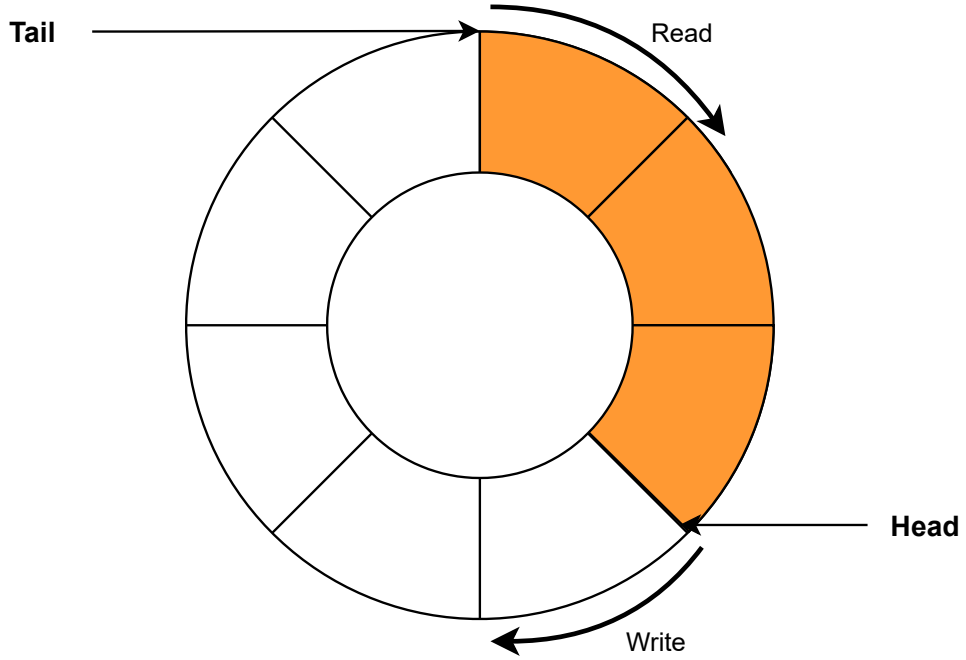


Abb. 5.5: Funktionsweise des „Circular buffer“

Die Funktion „available“ gibt die Anzahl der Bytes zurück, die sich im Puffer befinden. Der Ablauf dieser Funktion ist aus dem Aktivitätsdiagramm in Abbildung 5.6 ersichtlich.

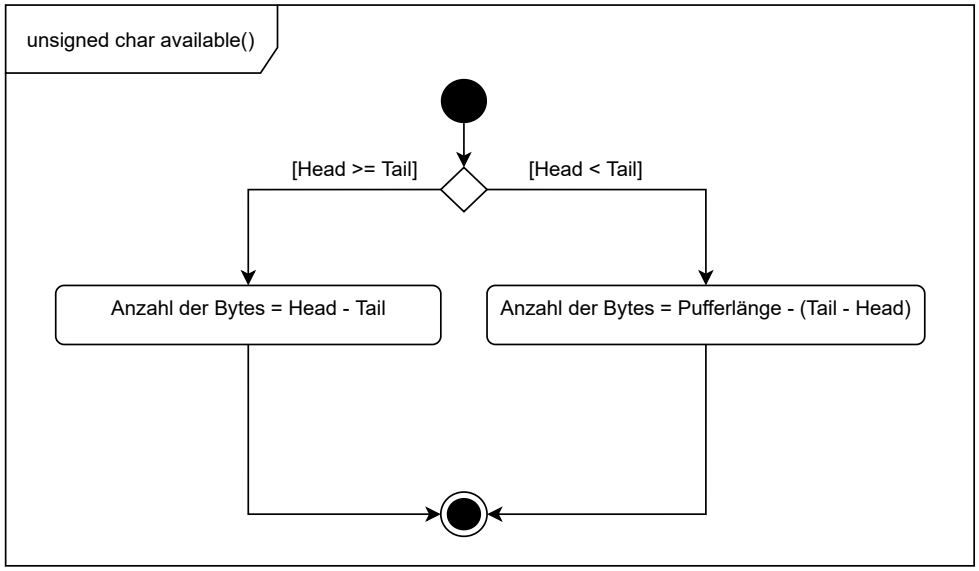


Abb. 5.6: Aktivitätsdiagramm der Funktion „available“

Der Prozess muss ermitteln, wo sich die Indizes befinden, um die Anzahl der im Puffer vorhandenen Bytes zu berechnen. Ein Pufferüberlauf muss ebenfalls berücksichtigt werden. In diesem Fall wird der gesamte Inhalt des Puffers gelöscht und eine Flag gesetzt. Diese Flag kann durch Aufruf der Funktion „overflow“ gelesen werden, die im Falle eines Pufferüberlaufs 1 zurückgibt.

Die Funktion „read“ bietet die Möglichkeit, ein Byte aus dem Puffer unter Berücksichtigung der Reihenfolge zu lesen. Das zuerst angekommene Byte wird auch zuerst gelesen. Das Diagramm in Abbildung 5.7 zeigt, wie dies funktioniert.

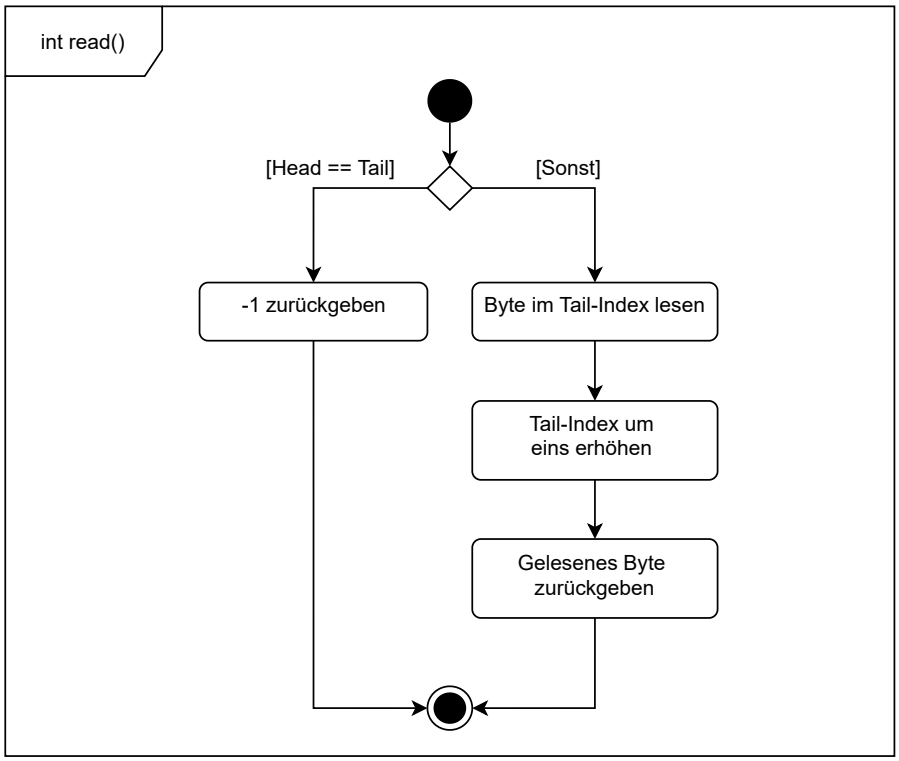


Abb. 5.7: Aktivitätsdiagramm der Funktion „read“

Sobald der „Head“- und der „Tail“-Index in der gleichen Position sind, bedeutet es, dass kein neues Byte vorhanden ist. Die Funktion identifiziert diese Situation und gibt -1 zurück. Andernfalls wird die Anzahl der im Puffer vorhandenen Bytes zurückgegeben.

5.1.2 Modulkonfigurator

Der Modulkonfigurator besteht aus den Funktionen „configBluetoothModule“ und „renameModule“. Im Folgenden werden diese beiden Funktionen näher erläutert. Die Funktion „configBluetoothModule“ wird verwendet, um den Namen und PIN des HC-06 zu konfigurieren. Diese Funktion aktiviert auch das UART-Kommunikationsprotokoll mit der Standard-Buadrate für das Bluetooth-Modul, die 9600 entspricht. Das Verfahren wird mit dem Aktivitätsdiagramm in Abbildung 5.8 beschrieben.

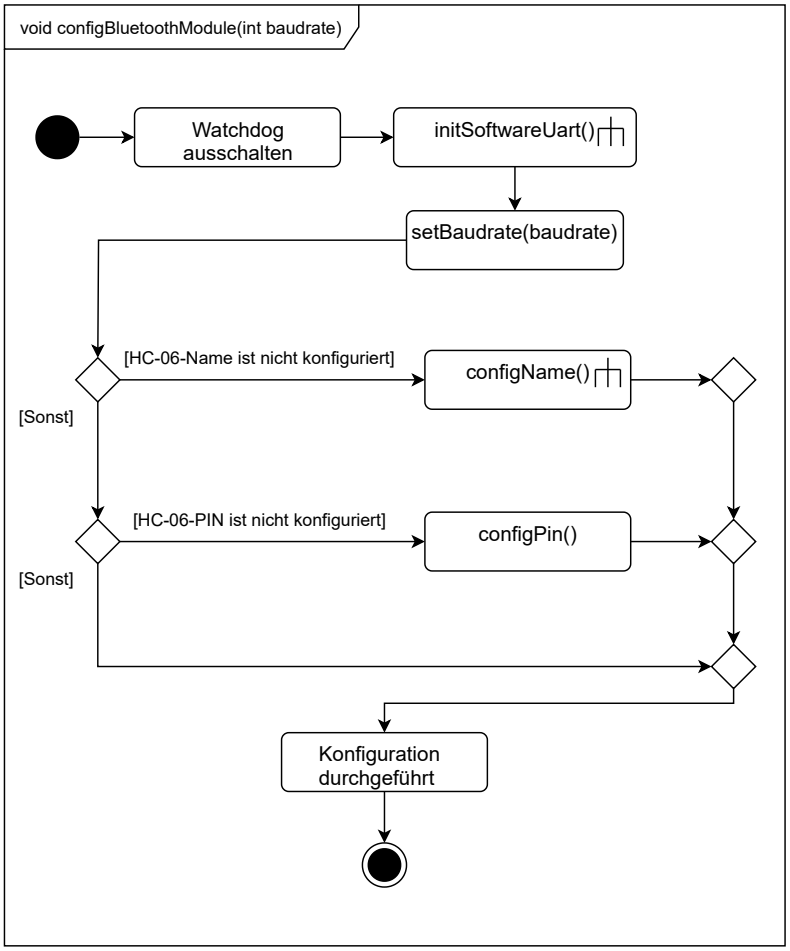


Abb. 5.8: Aktivitätsdiagramm der Funktion „configBluetoothModul“

Der erste Schritt ist die Deaktivierung des Watchdog, um ein ungewolltes Zurücksetzen des Mikrocontrollers zu vermeiden. Dann wird die UART-Kommunikation durch die Funktionen „initSoftwareUart“ aktiviert, die zuvor in Absatz 5.1.1 beschrieben wurde.

Am effektivsten ist es, den EEPROM-Speicher des ATtiny85 zu nutzen, um eine Variable zu speichern, die signalisiert, ob die Konfiguration bereits vorgenommen wurde. Der Mikrocontroller wird nach der Beendigung der „configPin“- und „configName“-Funktionen unter Verwendung des Watchdog neu gestartet. Das Verfahren zur Einstellung der PIN ist dasselbe wie das zur Einstellung des Namens, nur dass eine andere Adresse im EEPROM verwendet wird, um die Konfiguration zu speichern.

Die Funktion „configName“ kümmert sich um die Konfiguration des zu verwendenden Namens. Falls ein benutzerdefinierter Name mit der Funktion „renameModule“ gespeichert wurde, wird dieser verwendet, andernfalls der Standardname „JILZ-V01“. Im Falle der Funktion „configPin“ wird nur der Standard-PIN „1234“ konfiguriert. Zunächst wird die Funktion „configName“, die im folgenden Aktivitätsdiagramm beschrieben wird, genauer erläutert.

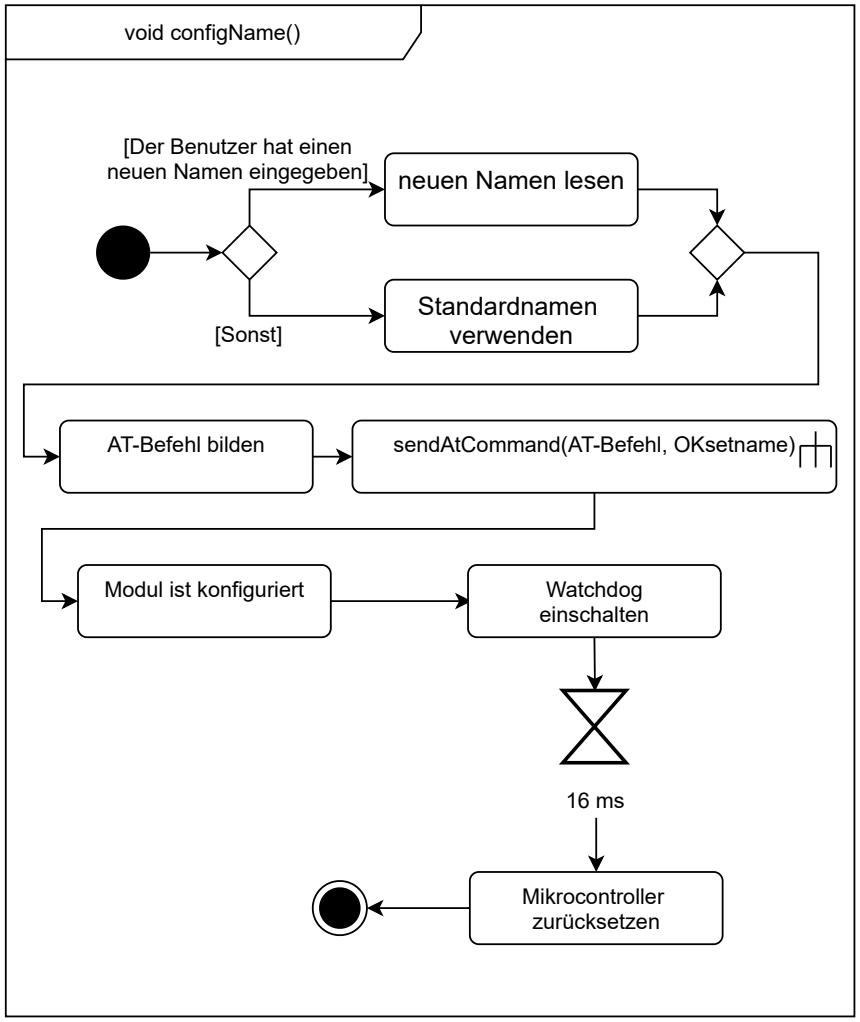


Abb. 5.9: Aktivitätsdiagramm der Funktion „configName“

Die Funktion beginnt mit dem Lesen der Information aus dem EEPROM, um herauszufinden, ob der Standardname verwendet werden soll oder nicht. Um den Namen des HC-06 zu ändern, ist der Befehl „AT+NAME“ gefolgt von dem gewünschten Namen zu senden. Im Falle des Standardnamens lautet der AT-Befehl „AT+NAMEJILZV-01“. Diese Zeichenfolge wird später per UART an das HC-06-Modul gesendet. Dieses antwortet mit „OKsetname“, um eine erfolgreiche Konfiguration zu signalisieren. Der Watchdog wird dann mit einer Zeit von 16 ms aktiviert, wodurch ein Reset des Mikrocontrollers ausgelöst wird.

Der Prozess der Konfiguration der PIN ist ähnlich wie bei der Konfiguration des Namens, mit dem Unterschied des zu sendenden Befehls. Im Falle der PIN wird der Befehl „AT+PIN“ gefolgt von der Standard-PIN, also „AT+PIN1234“, verwendet. Der Sendeprozess eines Befehls und das Warten auf eine Antwort durch die „sendATCommand“-Funktion wird in Abbildung 5.10 ausführlich beschrieben.

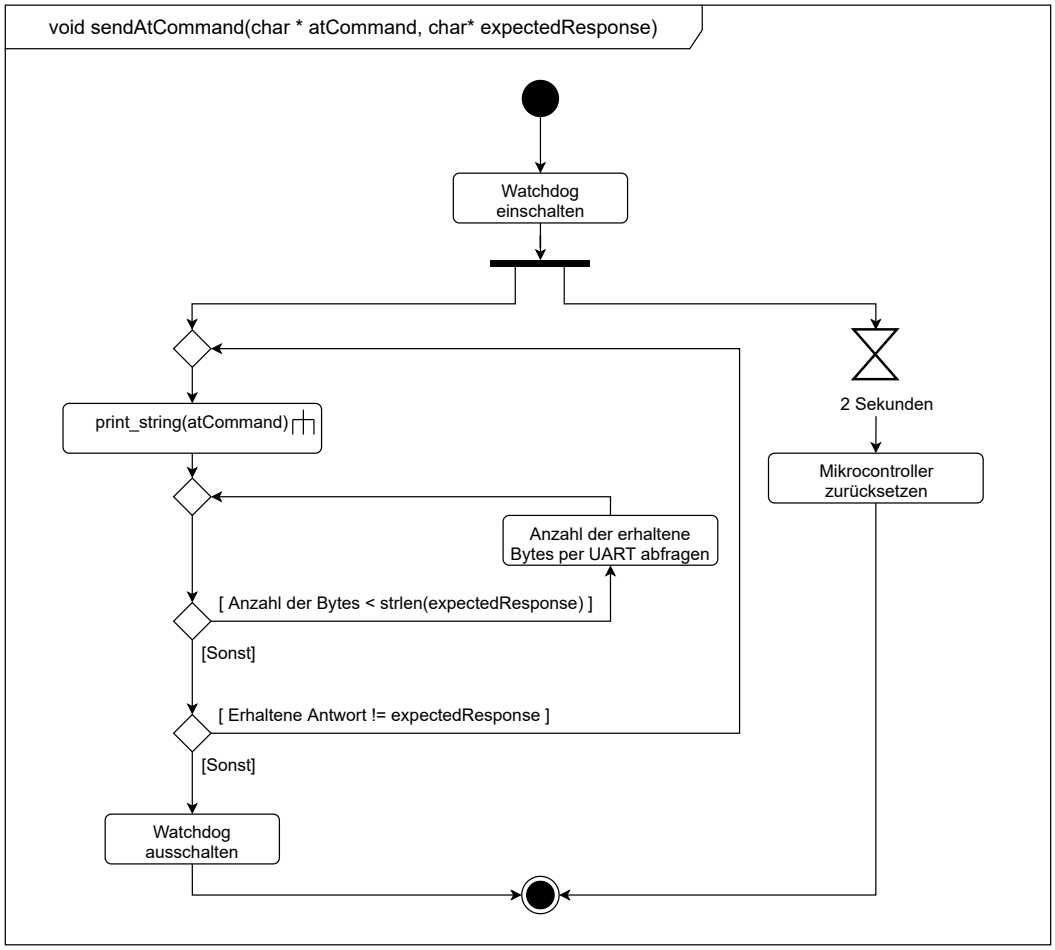


Abb. 5.10: Aktivitätsdiagramm der Funktion „sendAtCommand“

Die Funktion sendet den Befehl über UART und wartet auf den Empfang aller Bits. Dann wird geprüft, ob die empfangene Antwort die erwartete ist. Um eine Endlosschleife zu vermeiden, wird der Watchdog mit einer Zeit von 2 Sekunden eingeschaltet. Wenn diese 2 Sekunden überschritten werden, wird der Mikrocontroller zurückgesetzt.

Sowohl die Namens- als auch die PIN-Konfiguration erfordern eine Speicherung von Daten im EEPROM-Speicher des ATtiny85, um abfragen zu können, ob die Konfiguration bereits vorgenommen wurde. Die Hexadezimalzahl 0xaf wird verwendet, um zu symbolisieren, dass die Konfiguration durchgeführt wurde. Wenn die Konfiguration des Namens abgeschlossen ist, wird diese Nummer in der Adresse 0x010 des EEPROMs gespeichert, im Falle der PIN in der Adresse 0x030. Abbildung 5.11 zeigt den EEPROM vor und nach der Konfiguration. Die Adressen lassen sich wie folgt zuordnen: „0x“ gibt an, dass es sich um eine Hexadezimalzahl handelt. Die ersten beiden Ziffern geben die Reihe und die letzte Stelle gibt die Spalte an. Die relevanten Werte und Adressen wurden in der Grafik außerdem farblich markiert.

nicht Konfiguriert

Address	0	1	2	3	4	5	6	7
00000000	ff	ff	ff	ff	ff	ff	ff	ff
00000010	ff	ff	ff	ff	ff	ff	ff	ff
00000020	ff	ff	ff	ff	ff	ff	ff	ff
00000030	ff	ff	ff	ff	ff	ff	ff	ff

Konfiguriert

Address	0	1	2	3	4	5	6	7
00000000	ff	ff	ff	ff	ff	ff	ff	ff
00000010	af	ff	ff	ff	ff	ff	ff	ff
00000020	ff	ff	ff	ff	ff	ff	ff	ff
00000030	af	ff	ff	ff	ff	ff	ff	ff

Abb. 5.11: EEPROM vor und nach Ausführung der Funktion „ConfigBluetoothModul“

Falls das Bluetooth-Modul nicht auf die Konfigurationsbefehle reagiert, wird der Wert 0xfd auf der Adresse 0x0000 des EEPROM gespeichert. Dieser Wert wird nach 210 (hexadezimal = 0xd2) erfolglosen Versuchen gespeichert. Die Anzahl der Versuche wird dafür unter der Adresse 0x0001 des EEPROMs gespeichert. Abbildung 4 zeigt einen Vergleich der beiden Fälle, dass das Bluetooth-Modul antwortet oder nicht antwortet.

Bluetooth-Modul antwortet

Address	0	1	2	3	4	5	6	7
00000000	ff	ff	ff	ff	ff	ff	ff	ff

Bluetooth-Modul antwortet nicht

Address	0	1	2	3	4	5	6	7
00000000	fd	d2	ff	ff	ff	ff	ff	ff

Abb. 5.12: Belegung des EEPROMS bei Nicht-Antworten des Bluetooth-Moduls

Die Anzahl der Fehlversuche von 210 kann mithilfe von Sekunden gemessen werden, da das Senden von AT-Befehlen durch den Watchdog-Timer des Mikrocontrollers überwacht wird. Der Mikrocontroller wird zurückgesetzt, wenn er nicht innerhalb von 2 Sekunden eine Antwort vom Bluetooth-Modul erhält. Das Zurücksetzen des Mikrocontrollers wird als ein Fehlversuch gewertet. Daher entspricht die Zeit, die für 210 Versuche benötigt wird, 420 Sekunden.

Die zweite Funktion des Modulkonfigurators ist „ renameModul“. Diese wird verwendet, um den Namen des Bluetooth-Moduls zu ändern und den neuen Namen im EEPROM zu speichern. Um zu signalisieren, dass ein neuer Name zur Konfiguration vorliegt, wird der Wert 0xcf unter der Adresse 0x011 gespeichert. Der Mikrocontroller liest diese Adresse. Der Name wird ab der Adresse 0x012 bis 0x026 abgespeichert, wobei der erste Buchstabe in 0x012 steht. Am Ende des Namens im Speicher steht immer die Hexadezimalzahl 0x21. Zulässige Zeichen für den Namen befinden sich zwischen 0x31 und 0x127. Abbildung 5.13 zeigt das EEPROM nach Ausführen dieser Funktion mit „Hallo Welt“ als neuem Namen.

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ff	f6	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ff	ÿöÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000010	ff	cf	48	61	6c	6c	6f	20	57	65	6c	74	21	ff	ff	ff	ÿÏHallo Welt!ÿÿÿ

Abb. 5.13: Speicherung des Namens „Hallo Welt“ im EEPROM

In der Grafik ist zu erkennen, dass der erste Buchstabe des Namens „H“ auf Adresse 0x012 und die hexadezimale Zahl 0xcf auf Adresse 0x011 gespeichert wird. Der Name „Hallo Welt“ hat 10 Zeichen, was bedeutet, dass der letzte Buchstabe an Adresse 0x01b und die hexadezimale Zahl 0x21 an Adresse 0x01c gespeichert wird. Im Dump-Segment werden die hexadezimalen Zahlen im ASCII-Format angezeigt.

Nach der Ausführung dieser Funktion setzt der Mikrocontroller das Bluetooth-Modul über seinen RST-Pin zurück, der eine Spannung von $0\,V$ haben muss, um einen Reset auszuführen. Anschließend wird auch der Mikrocontroller zurückgesetzt.

5.1.3 Steuereinheit

Die Steuereinheit interpretiert die Kommandos der Smartphone-Anwendung. Das Modul basiert auf dem Zustandsautomaten, der in Abbildung 4.13 beschrieben ist. Um den Zustandsautomaten in Code umzusetzen, wird eine Switch-Case-Struktur implementiert, bei der jeder Zustand als ein „Case“ definiert wird. Die Steuereinheit interpretiert einerseits die Ein- und Ausschaltbefehle und andererseits die Namensänderung.

Um die Funktionsweise dieses Moduls visuell darzustellen, werden Sequenzdiagramme in Abbildung 5.14 und 5.15 verwendet. Zuerst wird der Prozess des Ladevorgang Stoppens abgebildet, der erfolgt, wenn die Smartphone-Anwendung den Befehl zum Ausschalten

sendet. Im Sequenzdiagramm 5.14 ist zu erkennen, dass der Ausschaltvorgang beginnt, wenn der HC-06 den Ausschaltbefehl von der Smartphone-Applikation erhält. Dann sendet das Bluetooth-Modul den Befehl weiter an den Mikrocontroller, der zunächst schläft, um Energie zu sparen, und geweckt wird, wenn der HC-06 beginnt, Daten zu senden. Danach sperrt der Mikrocontroller den MOSFET, um die Stromzufuhr zu unterbrechen. Anschließend wird der Buchstabe 'Y' an den HC-06 geschickt, was diesem die erfolgreiche Durchführung bestätigt. Andernfalls wird das Wort „NACK“ gesendet, um zu symbolisieren, dass die Software den Befehl nicht erkannt hat.

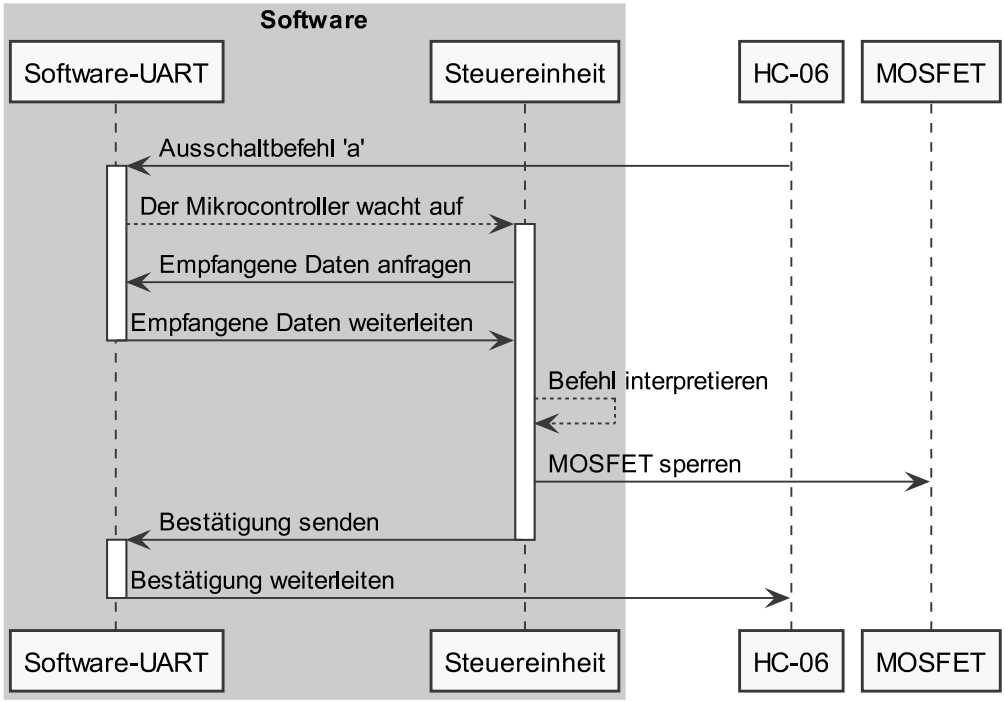


Abb. 5.14: Abschaltsequenz des MOSFETs

Das erneute Aktivieren des Ladevorgangs funktioniert ähnlich, es werden allein die Befehle geändert, wobei der Einschaltbefehl dem Buchstaben 'e' und dessen Bestätigung dem Buchstaben 'X' entspricht.

Auf diese Weise wird geregelt, ob der Ladevorgang aktiv ist oder inaktiv ist. Dabei gilt, dass der Vorgang i.d.R. aktiviert ist und nur beim Erhalt eines anderweitigen Befehls durch die Smartphone-Anwendung unterbrochen wird.

Das erneute Aktivieren kann durch den entsprechenden Befehl oder durch Abbruch der Bluetooth-Kommunikation ausgelöst werden. Zweiteres stellt sicher, dass das an das Vorschaltgerät angeschlossene Smartphone auf jeden Fall geladen wird. Hierfür benötigt die Steuereinheit einen Pin Change Interrupt, um zu erkennen, ob das Bluetooth-Modul noch mit dem Smartphone verbunden ist. Der Interrupt wird nur im Zustand „Load Off' aktiviert, damit die Steuereinheit bei einer Unterbrechung der Bluetooth-Verbindung in den Zustand „Load On“ zurückkehrt.

Um den Namen des HC-06, wie im vorigen Unterkapitel beschrieben, zu ändern, muss der Mikrocontroller von der Smartphone-Anwendung den Befehl „X“ gefolgt von dem gewünschten Namen erhalten, zum Beispiel „XHallo Welt“. Dieser Vorgang wird im Sequenzdiagramm in Abbildung 5.15 beschrieben.

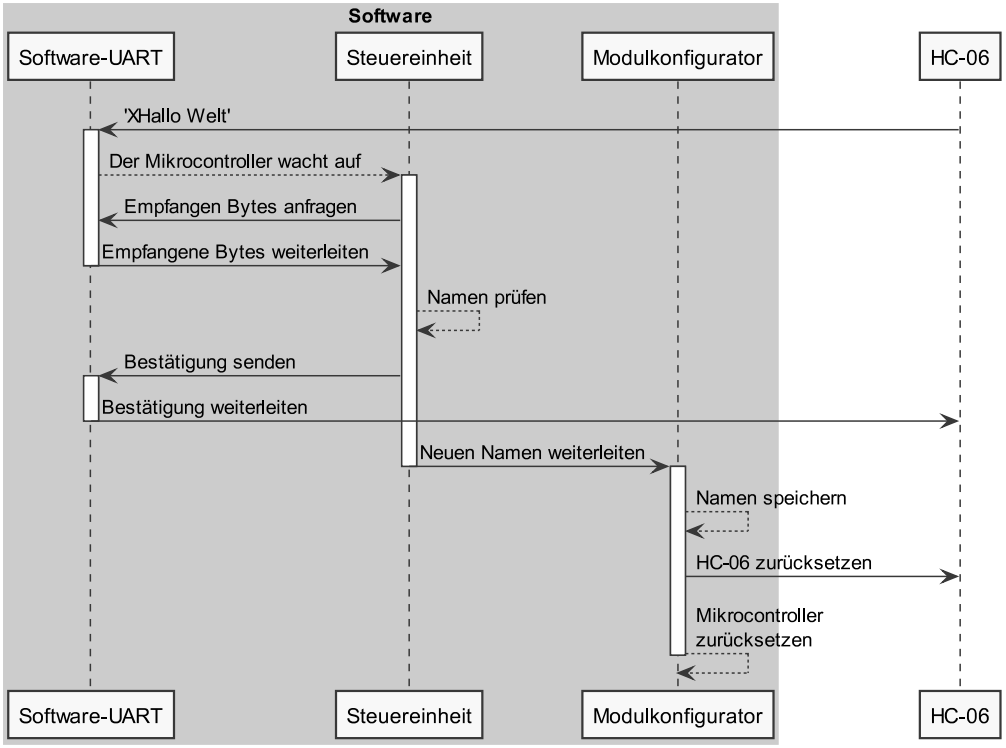


Abb. 5.15: Verfahren zur Änderung des Namens

5.2 Mikrocontroller-Pinout

In diesem Abschnitt wird erklärt, wie die Mikrocontroller-Pins mit der Hardware verbunden werden. Der ATtiny85-Mikrocontroller hat 6 Pins, die in Abbildung 5.16 gezeigt werden. Die möglichen Verwendungszwecke sind neben dem jeweiligen Pin aufgelistet. Pin Nummer 1 oder „PB5“ wird zum Zurücksetzen des Mikrocontrollers verwendet. Um diese Verwendung zu deaktivieren und diesen Pin stattdessen als GPIO zu verwenden, muss die „RSTDISBL“-Fuse programmiert werden. Dadurch wird gleichzeitig die Programmierung des Mikrocontrollers über ISP deaktiviert, so dass ein Hochspannungsprogrammiergerät zur Neuprogrammierung des Mikrocontrollers verwendet werden muss [20].

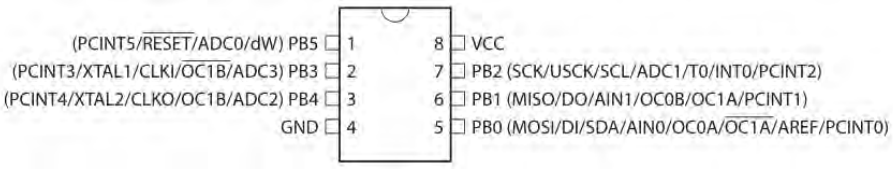


Abb. 5.16: Pinout des ATtiny85-Mikrocontrollers [20]

In der Software werden die 6 Pins des Mikrocontrollers verwendet. Tabelle 5.1 gibt an, wie jeder Pin konfiguriert wird und welche Funktion ausgeführt werden soll.

Tab. 5.1: Verwendung der ATtiny85-Pins

Pin	Aufgabe	Konfiguration
PB0	Steuerung der RGB-LED	Digital-Output
PB1	TX-Pin der UART-Kommunikation	Digital-Output
PB2	RX-Pin der UART-Kommunikation	Digital-Input als INT0
PB3	Überwachung der Bluetooth-Verbindung	Digital-Input als PCINT3
PB4	Steuerung des RST-Pins	Digital-Output
PB5	Steuerung des MOSFETs	Digital-Output

5.3 Hardware-Entwicklung

Die Hardware-Komponenten werden gemäß der in Abschnitt 4.2 beschriebenen Hardware-Architektur angeschlossen. Um die Hardware-Konstruktion durchführen zu können, ist es notwendig, Transistoren zur Steuerung der RGB-LED und des MOSFETs zu verwenden, da die ATtiny85-Pins nur maximal 10 mA pro Pin liefern können [20]. Dabei besteht die Gefahr, dass die Pins beim Schalten zerstört werden. Aufgrund des geringen Preises und der kleinen Größe, die zur Erfüllung von Req-12 und Req-8 beitragen, ist das verwendete Transistormodell der BC847. Für die Ansteuerung der RGB-LED werden zwei Transistoren benötigt, von denen einer vom HC-06 und der andere vom ATtiny85 angesteuert wird, wie man auf Abbildung 5.17 sehen kann.

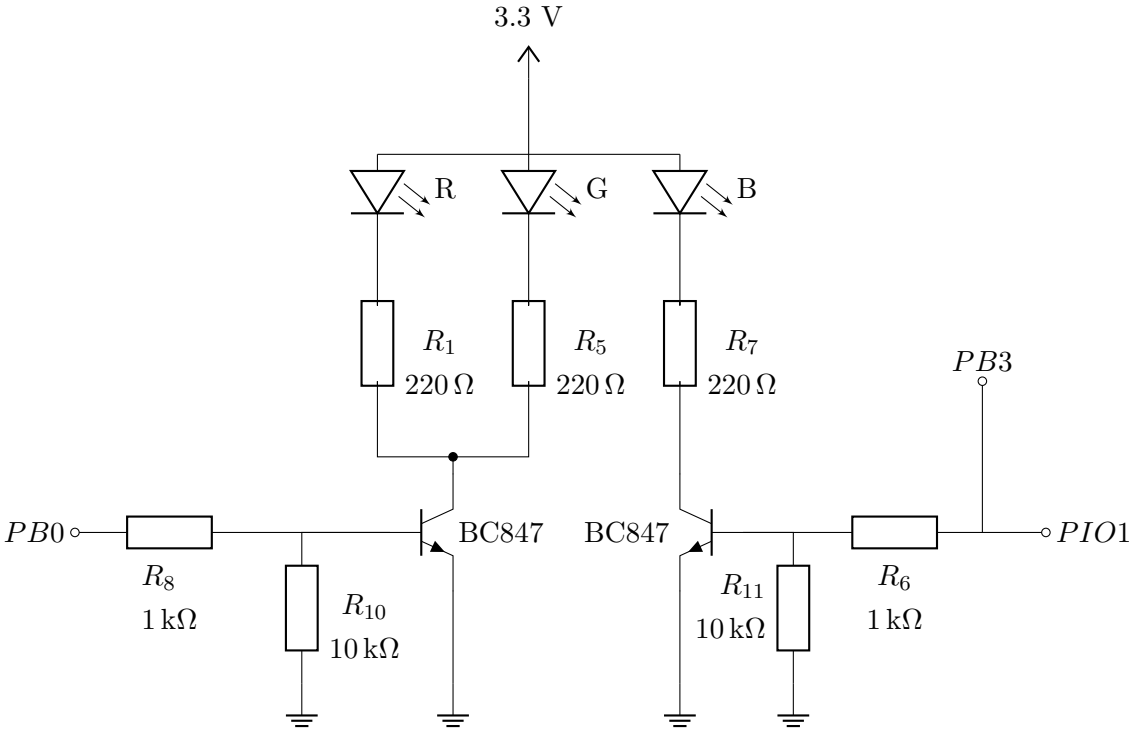


Abb. 5.17: Schaltung zur Steuerung der RGB-LED

Der ATtiny85 steuert über den Pin „PB0“ die rote und grüne Farbe der RGB-LED. Das HC-06 steuert die blaue Farbe über seinen Pin „PIO1“, sodass ein blaues Blinklicht ausgestrahlt wird, wenn das HC-06 über Bluetooth mit keinem Gerät verbunden ist. Wenn eine Bluetooth-Verbindung besteht, wird ein kontinuierliches blaues Licht erzeugt.

Die Signalgebung wurde wie beschrieben festgelegt, da diese Sequenz in vielen Geräten, die mit Bluetooth arbeiten, verwendet wird [13]. Somit sollte die Signalgebung intuitiv verständlich und damit Req-10 erfüllt sein. Zusätzlich wird in diesem Projekt weißes Licht verwendet, um dem Benutzer anzuzeigen, dass der Ladevorgang abgeschlossen ist. Dazu aktiviert der ATtiny85 den Stromdurchgang zur roten und grünen LED, in dem Moment, in dem er den Befehl erhält, den Ladevorgang zu beenden. Die Kombination mit der noch leuchtenden blauen LED führt zu weißem Licht.

Zum Erreichen einer möglichst kleinen Größe werden für die Widerstände der SMD-Typ mit einer Größe von 0805 ausgewählt. Jede LED wird mit einem Serienwiderstand von 220 Ω versehen. Außerdem gibt es jeweils zwei Widerstände für die Transistoren: einer von 1 $k\Omega$, der den Basisstrom zum Transistor liefert, und ein Pulldown-Widerstand von 10 $k\Omega$. Die Nummerierung der Widerstände entspricht dem in Abschnitt A.2 beigefügten Schaltplan.

Zur Ansteuerung des MOSFETs werden ein Transistor, ein Z-Diode und vier Widerstände verwendet, wie in Abbildung 5.18 dargestellt wird. Der Grund für die Verwendung eines Transistors liegt ebenfalls darin, dass ein Pin des ATtiny85 nur 10 mA liefern kann. Die beste Option besteht also darin, die VBUS-Spannung des Ladegeräts zur Versorgung des MOSFET-Gates zu verwenden. Der Grund für die Verwendung einer Z-Diode ist der Schutz des MOSFET-Gates, da die VBUS-Spannung je nach Hersteller des Ladegeräts variieren kann. Über die Widerstände R3 und R2 wird das MOSFET-Gate mit Ladung gefüllt. Sobald der NPN-Transistor aktiviert ist, fließt die gesammelte Ladung durch den Widerstand R2. Da der MOSFET nicht von einem hochfrequenten Signal angesteuert wird, können die Widerstände mit einem höheren Wert von 1 $k\Omega$ oder höher dimensioniert werden. Die Funktion der Wiederstande R2 und R3, besteht darin, den Strom zum MOSFET-Gate und zum NPN-Transistor zu begrenzen.

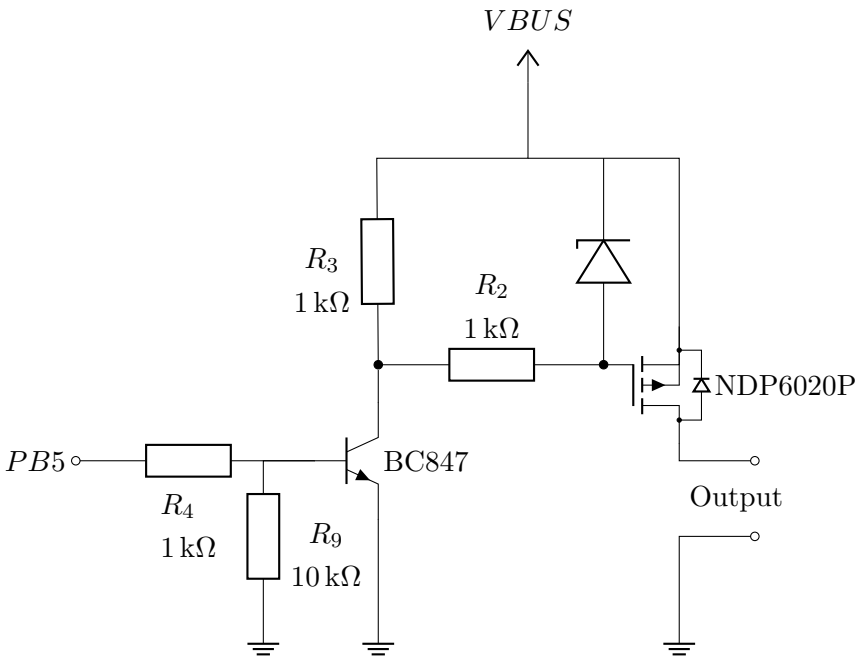


Abb. 5.18: Schaltung zur Steuerung des MOSFETs

Die Schaltung funktioniert wie folgt: Wenn der Pin „PB5“ des Mikrocontrollers auf "HIGH"gesetzt ist, beginnt der NPN-Transistor zu leiten, was eine negative Spannung zwischen den Gate- und Source-Anschlüssen des MOSFETs erzeugt. Da der MOSFET „NDP6020P“ von Typ P-MOS ist, wird den MOSFET niederohmig, was den Stromfluss zum Smartphone ermöglicht. Die Spannung U_{GS} des MOSFETs darf laut Datenblatt eine Spannung von mehr als $8\,V$ nicht überschreiten [24]. Daher muss die zu verwendende Z-Diode eine Durchbruchspannung zwischen $5\,V$ und $7\,V$ Volt aufweisen, damit der MOSFET ordnungsgemäß funktioniert. Als Zener-Diode wurde die „BZX85C5V1-TR“ der Firma Vishay mit einer Durchbruchspannung von $5,1\,V$ gewählt. Die USB-A-Buchse wird an den Schaltungsausgang angeschlossen, wobei der VBUS-Leitung mit dem Transistor-Drain verbunden wird. Die Datenleitungen D+ und D- der USB-A-Buchse werden mit den Datenleitungen D+ und D- der USB-B-Buchse verbunden.

Für die Namensänderung des HC-06-Modul ist auch die Verwendung eines Transistors erforderlich, um unerwünschte Rücksetzungen zu vermeiden. Die Schaltung in Abbildung 5.19 wird zur Steuerung des RST-Pins des HC-06-Moduls verwendet.

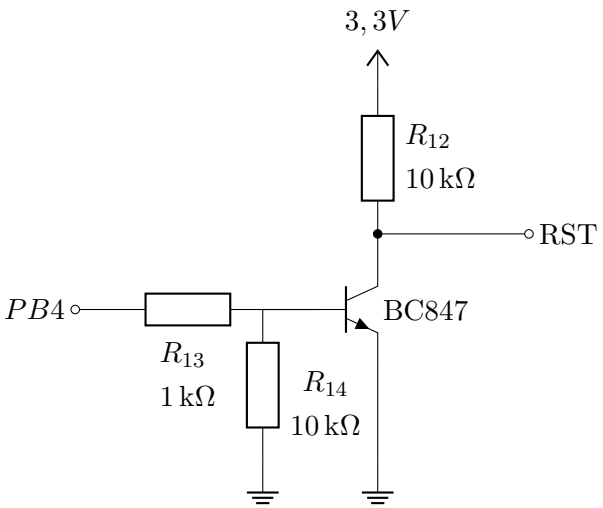


Abb. 5.19: Schaltung zur Steuerung des RST-Pins des HC-06-Moduls

Wenn das HC-06-Modul umbenannt wird, ändert der ATtiny85 den Zustand von Pin „PB4“ auf HIGH, wodurch der Transistor aktiviert wird. Das bedeutet, dass der „RST“-Pin des HC-06-Moduls kein 3,3 V-Potential hat, sondern 0V, was dazu führt, dass das Modul zurückgesetzt wird. Anschließend wird der Mikrocontroller zurückgesetzt, indem ein Watchdog-Reset ausgelöst wird. Da die Basis des Transistors einen Pulldown-Widerstand hat, wird der Transistor gesperrt, wenn der Pin „PB4“ einen undefinierten Zustand einnimmt. Dadurch wird ein mehrfaches Zurücksetzen vermieden.

Um die Baugröße möglichst gering zu halten, wird eine Custom-PCB angefertigt. Dies soll die Umsetzung von Req-8 erleichtern. Um einen visuellen Eindruck zu vermitteln, wird hier ein 3D-Modell der Platine mit all ihren Komponenten abgebildet. Abbildung 5.20 zeigt die Draufsicht und Abbildung 5.21 die Unterseite der PCB.

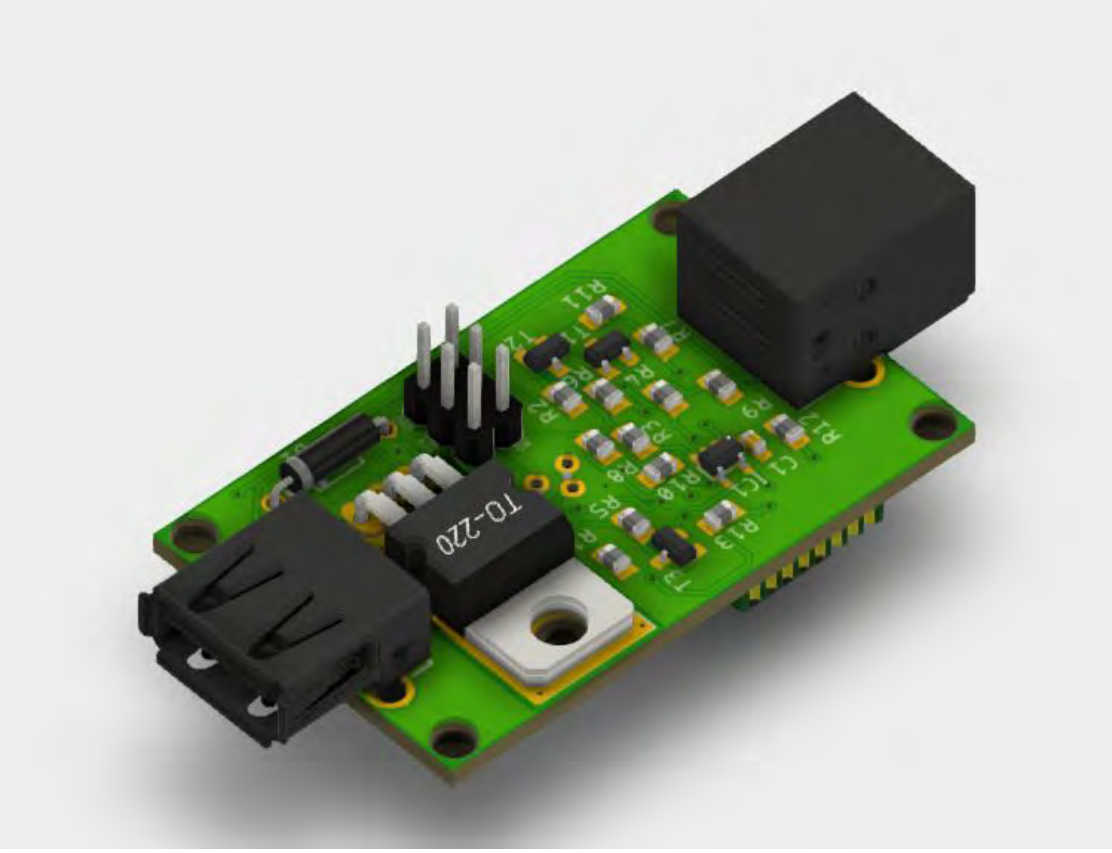


Abb. 5.20: 3D-Draufsicht der Platine

Auf der Oberseite der Platine befinden sich der MOSFET, der Spannungsregler sowie die meisten Widerstände und Transistoren. Man kann auch die sechs Stecker sehen, die zur Programmierung des Mikrocontrollers erforderlich sind.

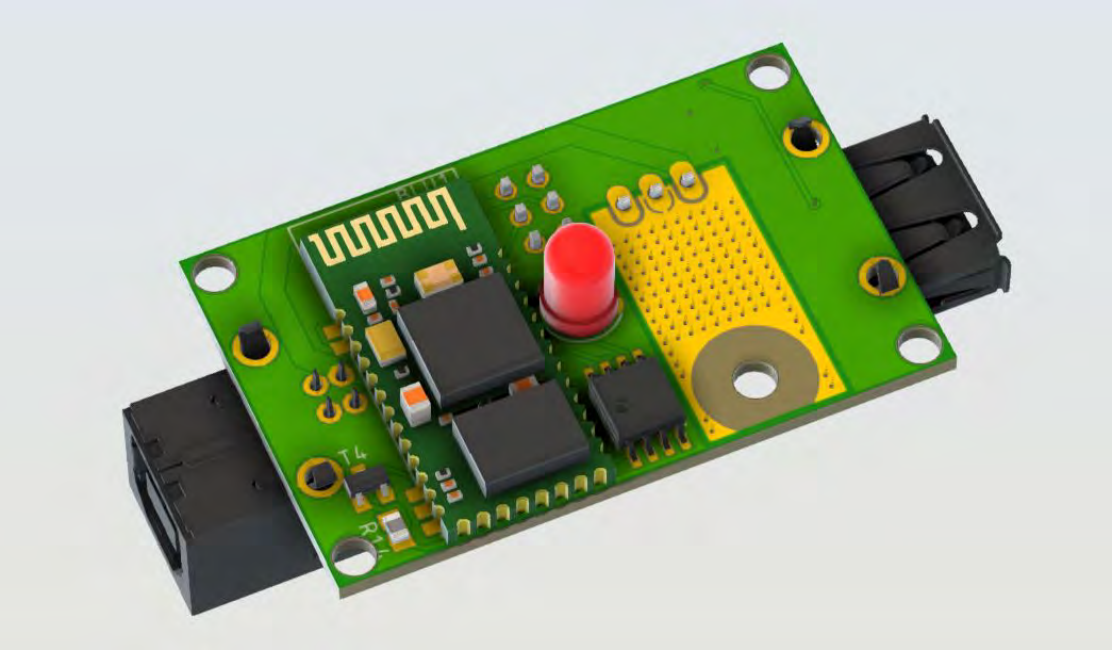


Abb. 5.21: 3D-Bodenansicht der Platine

Auf der Unterseite sind das HC-06-Modul, der ATtiny85 und die RGB-LED zu erkennen. Um eine Überhitzung des MOSFETs zu verhindern, wurde ein Kühlkörper direkt auf der Leiterplatte angebracht. Dieser befindet sich neben der RGB-LED und dem Mikrocontroller. Die Löcher darin sind Wege, die helfen, die Hitze abzuleiten. Diese Maßnahme wurde zum einen als Sicherheitsmaßnahme und zum anderen im Hinblick auf eine spätere Regelung des Ladestroms durchgeführt.

Um die Platine abzudecken, wird ein 3D-Modell eines Gehäuses entworfen. Für die Erstellung des Gehäuses wurde das Programm „FreeCad“ verwendet. Das Gehäuse verfügt über vier Bohrungen mit einem Durchmesser von 2,9 mm und einer Länge von 13 mm zur Befestigung der Platine im Gehäuse. Die 3D-Modelle des Gehäuses sowie das 3D-Modell der Platine befinden sich im Ordner „3D-Modelle“ der beigefügten CD.

6 Evaluation

Dieses Kapitel ist in zwei Abschnitte unterteilt: der Abschnitt Anforderungsprüfung, in dem deutlich gemacht wird, welche Teststrategie verfolgt wurde, um die Anforderungen zu prüfen, und der Abschnitt Hardwareprüfung, in dem die Schaltung getestet wird.

6.1 Hardwareprüfung

In diesem Abschnitt wird durch Simulationen untersucht, ob die Schaltung korrekt arbeitet sowie ob sich die Komponenten in dem vom Hersteller angegebenen Arbeitsbereich befinden. Die zu untersuchende Schaltung ist die von Abbildung 5.18, weil sie für die Erfüllung der Hauptfunktionalität des Vorschaltgeräts zuständig ist.

Um den Stromfluss in den Schaltungskomponenten zu analysieren, wird zunächst der Stromfluss des Transistors BC847 betrachtet, wenn Pin „PB5“ des ATtiny85 von LOW auf HIGH wechselt. Abbildung 6.1 zeigt die Simulation der Schaltung mit einer an VBUS anliegenden Spannung von $9\,V$, wobei der Strom im Transistor mit der blauen Linie und die Spannung an Pin „PB5“ mit der roten Linie dargestellt ist.

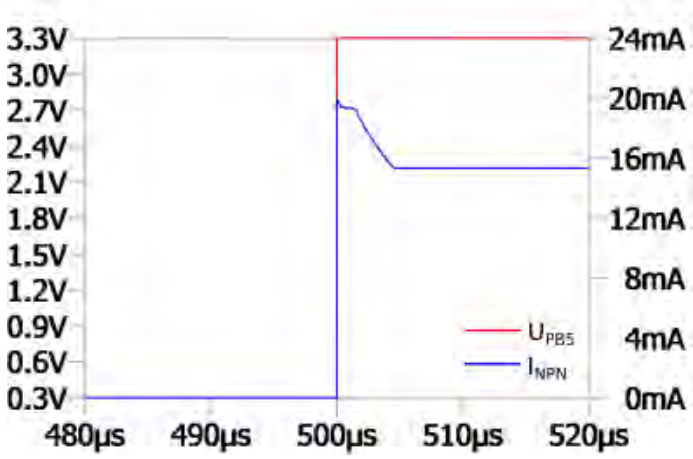


Abb. 6.1: Strom im NPN-Transistor nach der Einschaltung des Ladevorgangs

Es ist zu erkennen, dass der Strom (I_{NPN}) eine Spitze von ca. $20mA$ hat, was auf die akkumulierte Ladung des MOSFET-Gates zurückzuführen ist, die im Moment der Durchschaltung des Transistors zu dem niedrigsten Potenzial fließt. Nachdem die Ladung des MOSFET-Gates vollständig durchgelaufen ist, nimmt der Strom einen konstanten Wert von ca. $15\,mA$ an. Dies ist auf die Summe der Ströme der Widerstände R3 (I_{R3}), R2 (I_{R2}) sowie des Stroms von der Basis des NPN-Transistors (I_B) zurückzuführen. Diese berechnen sich wie folgend:

$$\begin{aligned} I_{R3} &= \frac{U_{VBUS}}{R_3} \\ I_{R3} &= \frac{9\,V}{1\,k\Omega} \\ I_{R3} &= 9\,mA \end{aligned} \tag{6.1}$$

$$\begin{aligned} I_{R2} &= \frac{U_{VBUS} - U_{Zener}}{R_2} \\ I_{R2} &= \frac{9\,V - 5,1\,V}{1\,k\Omega} \\ I_{R2} &= \frac{3,9\,V}{1\,k\Omega} \\ I_{R2} &= 3,9\,mA \end{aligned} \tag{6.2}$$

$$
\begin{aligned}
I_B &= \frac{U_{Pin} - U_{BE}}{R_4} \\
I_B &\approx \frac{3,3\,V - 0,7\,V}{1\,k\Omega} \\
I_B &= 2,6\,mA
\end{aligned}
\tag{6.3}
$$

$$
\begin{aligned}
I_{NPN} &= I_B + I_{R2} + I_{R3} \\
I_{NPN} &= 2,6\,mA + 3,9\,mA + 9\,mA \\
I_{NPN} &= 15,5\,mA
\end{aligned}
\tag{6.4}
$$

Es ist ersichtlich, dass der Stromwert weit unter dem vom Hersteller angegebenen Maximalwert von 100 mA liegt [5].

Der vom Hersteller angegebene Maximalwert der Kollektor-Emitter-Spannung (U_{CEsat}) wird zur Berechnung der Leistung an dem Transistor (P_{NPN}) verwendet:

$$
\begin{aligned}
P_{NPN} &= I_{NPN} \cdot U_{CEsat} \\
P_{NPN} &= 15,5\,mA \cdot 600\,mV \\
P_{NPN} &= 9,3\,mW
\end{aligned}
\tag{6.5}
$$

Diese Leistung liegt in einem deutlich niedrigeren Bereich als die vom Hersteller angegebene maximale Leistung von 250 mW [5].

Die Leistung an der Z-Diode (P_{Zener}) wird mit dem Strom aus Gleichung (6.2) berechnet:

$$
\begin{aligned}
P_{Zener} &= I_{R2} \cdot U_{Zener} \\
P_{Zener} &= 3,9\,mA \cdot 5,1\,V \\
P_{Zener} &= 19,89\,mW
\end{aligned}
\tag{6.6}
$$

Auch hier wird deutlich, dass der Wert unter der vom Hersteller angegebenen $1,3\,W$ Maximalleistung liegt [40].

Um zu bestimmen, ob der MOSFET leitet oder nicht, muss die Gate-Spannung untersucht werden. Abbildung 6.2 zeigt die Gate-Spannung des MOSFETs, nachdem der Pin „PB5“ des ATtiny85 von LOW auf HIGH wechselt.

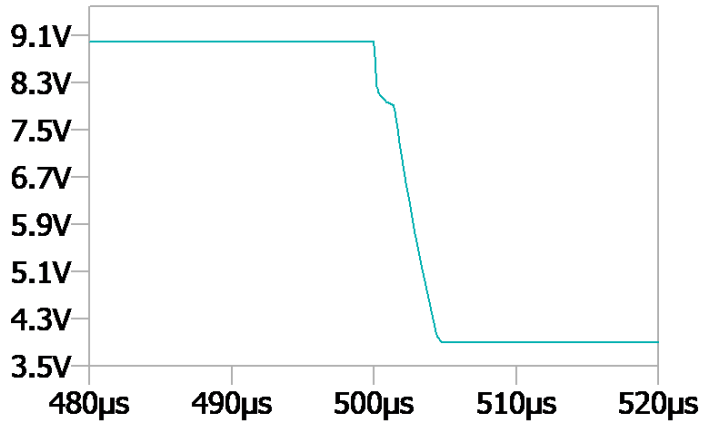


Abb. 6.2: Gate-Spannung nach der Einschaltung des Ladevorgangs

Die Simulation zeigt, wie die Spannung auf $3,9\,V$ abfällt. Die nichtlinearen Effekte sind auf die parasitären Kapazitäten des MOSFETs zurückzuführen. Dieser Effekt spielte in dieser Arbeit keine große Rolle, da der MOSFET nicht mit einem hochfrequenten Signal angesteuert wird. [41]

Die Spannung zwischen Gate und Source (U_{GS}) wird wie folg berechnet:

$$\begin{aligned} U_{GS} &= U_G - U_{VBUS} \\ U_{GS} &= -5,1\,V \end{aligned} \tag{6.7}$$

Diese Spannung bewirkt, dass der MOSFFET niederohmig wird und Strom zum Smartphone fließen kann.

Die Simulation zeigt eine Spannung zwischen Drain und Source (U_{DS}) von $80\,mV$ bei einem Strom von $2,5\,A$ (I_{ds}), was zu folgender Leistung (P_{MOS}) führt:

$$\begin{aligned} P_{MOS} &= I_{ds} \cdot U_{DS} \\ P_{MOS} &= 0,2\,W \end{aligned} \tag{6.8}$$

Diese Leistung liegt weit unter der maximalen Leistung von $60\,W$ des MOSFETs, so dass der MOSFET keine Schwierigkeiten hätte, noch höhere Ströme durchzuschalten [24].

Um den Ladevorgang zu stoppen, wechselt der Pin „PB5“ von einem HIGH- in einen LOW-Zustand, wodurch der Transistor BC847 gesperrt wird. Die Simulation in Abbildung 6.3 zeigt in Grün den Verlauf des Gate-Stroms und in Rot den Verlauf der Gate-Spannung sowie die Spannung von Pin „PB5“ in Blau.

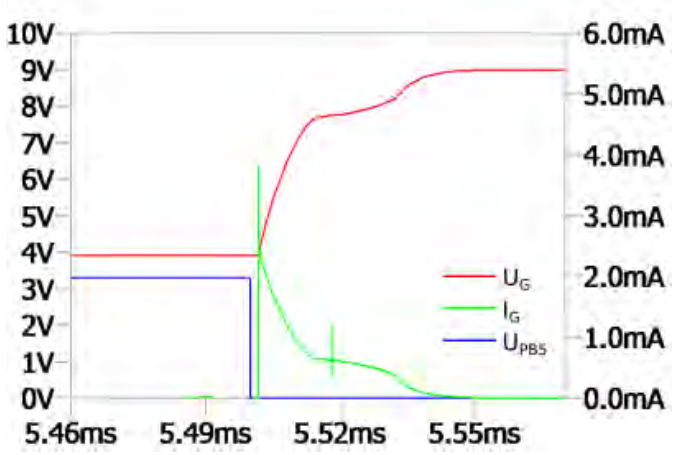


Abb. 6.3: MOSFET-Gate zum Zeitpunkt der Ausschaltung des Ladevorgangs

Der Strom beginnt durch die Widerstände R3, R2 und die Z-Diode zu fließen, bis das Gate geladen ist. Schließlich steigt die Spannung auf den VBUS-Wert an, der in diesem Fall 9 V beträgt. Dadurch wird die Spannung U_{GS} gleich Null und der MOSFET sperrt. [41]

6.2 Anforderungsprüfung

In diesem Abschnitt wird überprüft, ob die in Tabelle 3.9 aufgelisteten Anforderungen an das Vorschaltgerät erfüllt wurden. Dazu werden zuerst die funktionalen, dann die Qualitätsanforderungen und schließlich die Namensänderung überprüft. Die Strategie, die zur Überprüfung der funktionalen Anforderungen verfolgt wird, basiert auf den Use Cases von Abschnitt 3.2. Um zu testen, ob ein Use Case korrekt funktioniert, wird wie folgt vorgegangen:

1. Prüfung der Vorbedingung
2. Use Case anwenden

3. Nachbedingung überprüfen

Welcher Use Case verwendet wird, um eine bestimmte Anforderung zu prüfen, ist in Tabelle 3.10 in der Spalte „Use Case“ angegeben. Diese Spalte gibt die Beziehungen einer Anforderung zu einem Use Case an. Die Vor- und Nachbedingungen werden aus der in Unterabschnitt 3.2.2 angegebenen Spezifikation entnommen. Stimmen die Nachbedingungen nach der Anwendung des Use Case Szenarios mit den in 3.2 angegebenen Bedingungen überein, gilt die Anforderung als erfüllt.

Tabelle 4 zeigt die Vor- und Nachbedingungen jedes Use Case sowie die jeweiligen Anforderungen, die mit diesem Anwendungsfall getestet werden. Die Tabelle gibt außerdem an, ob die Anforderungen erfüllt wurden.

Tab. 6.1: Überprüfung der funktionale Anforderungen

ID	Vorbedingung	Use Case	Nachbedingung	Erfüllt
Req-1 Req-5	Das Vorschaltgerät ist mit Spannung versorgt	U3	Das Vorschaltgerät ist im Zustand „Ladevorgang aktiv“	**Ja**
Req-2	Das Vorschaltgerät ist im Zustand „Ladevorgang inaktiv“	U5	Das Vorschaltgerät geht in den Zustand „Ladevorgang aktiv“ zurück	**Ja**
Req-3	Das Vorschaltgerät ist im Zustand „Ladevorgang aktiv“	U4	Das Vorschaltgerät geht in den Zustand „Ladevorgang inaktiv“ über	**Ja**
Req-4	Das Vorschaltgerät ist mit Spannung versorgt	U0	Die RGB-LED zeigt die Zustände des Vorschaltgeräts an	**Ja**

Die Tabelle zeigt, dass die funktionalen Anforderungen erfüllt wurden und dementsprechend das Gerät funktioniert. Des Weiteren werden zur Überprüfung der Qualitätsanforderungen folgende Maßnahme durchgeführt:

- Kostenanalyse
  Eine Kostenanalyse wird benötigt, um zu überprüfen, ob Req-12 erfüllt wurde.

- Messung des Vorschaltgeräts
  Mit einer Messung der Außenmaße wird festgestellt, ob das Vorschaltgerät innerhalb der in Req-8 angegebenen Maße liegt.

- Benutzer-Feedback
  Durch Beobachtung von Benutzereaktionen auf die Signalgebung der RGB-LED kann gemessen werden, ob Req-10 erfüllt wurde.

- Prüfung der Gehäusestabilität
  Dieser Test zielt darauf ab, die Fähigkeit des Gehäuses gegen Stürze zu prüfen, um festzustellen, ob Req-9 erfüllt wurde.

Um die Kosten zu analysieren, werden die Preise aller Hardwarekomponenten aus der Stückliste von Abschnitt A.1 addiert. Das ergibt eine Summe von 11,80 Euro als Gesamtpreis. Dieser Preis liegt leicht über dem in Req-12 festgelegten Grenzwert von 10 Euro. Der Grund dafür ist der hohe Preis des HC-06 von 4,50 Euro. Deshalb sollte für eine spätere Version des Vorschaltgeräts nach einem anderen Bluetooth-Modul gesucht werden.

Mit 64mm x 41,44mm x 26,95mm liegen die Maße des Vorschaltgeräts innerhalb des in Req-8 festgelegten Bereichs. In einer zukünftigen Version sollte dennoch darauf geachtet werden, die Größe weiter zu reduzieren, um das ästhetische Erscheinungsbild des Vorschaltgeräts zu verbessern.

Die Rückmeldungen 5 verschiedener Testnutzer zur Farbe und Leuchten bzw. Blinken der LED waren positiv, da die blaue Farbe schnell mit einer erfolgreich hergestellten Bluetooth-Verbindung in Beziehung gesetzt wurde. Hiermit kann Req-10 als erfüllt gelten.

Nach mehrmaligem Fallenlassen des Gehäuses aus 60 cm Höhe wurden keine relevanten Schäden am Gehäuse festgestellt. Das liegt vor allem daran, dass vier Schrauben vorhanden sind, die das Gehäuse zusammenhalten.

Die Namensänderung des Vorschaltgeräts konnte in Kombination mit der Smartphone-Anwendung nicht getestet werden, da die Anwendung beim Ändern des Namens keine Befehle sendet. Deshalb wurde ohne Anwendung getestet, indem ein Serial Bluetooth Terminal verwendet wurde, welches ermöglicht, mit einem Smartphone verschiedene Befehle zu senden. Die Ergebnisse waren positiv, sodass man davon ausgehen kann, dass eine Änderung des Gerätenamens auch mit der App problemlos möglich ist. Damit die Funktionalität implementiert werden kann, muss ein Upgrade der Smartphone-Anwendung durchgeführt werden.

Tab. 6.2: Überprüfung der Namensänderung

Test Case	**Nachbedingung**	**Erfüllt**
Befehl „XHallo Welt“ senden	Name geändert	**Ja**
Befehl „X123456789123456789AB“ senden	Name geändert	**Ja**
Befehl „X123456789123456789ABC“ senden	Keine Änderung	**Ja**
Befehl „X“ senden	Keine Änderung	**Ja**

7 Schlussbetrachtung

7.1 Zusammenfassung

Ziel dieser Arbeit war es, die Lebensdauer eines Smartphone-Akkus zu verbessern. Dies wurde durch Einschränkung des Ladezustands erreicht. Dazu war es notwendig, eine Schaltung zu erstellen, die in der Lage ist, die Anweisungen der in der Bachelorarbeit von Emrhra Demir entwickelte Smartphone-Anwendung zu interpretieren und auszuführen. Die Lösung basiert auf einer Hardwareschaltung mit einem Bluetooth-Modul, das in der Lage ist, Daten über Bluetooth zu senden und zu empfangen sowie einem Mikrocontroller, der die über Bluetooth empfangenen Nachrichten interpretiert und die entsprechenden Aktionen auszuführen kann. Wobei in diesem Fall die Hauptfunktion darin besteht, den Ladevorgang des Smartphones mit Hilfe eines MOSFETs, der als Schalter verwendet wird, zu stoppen. Die Hardwarekomponenten sind auf einer Platine platziert, die durch ein Gehäuse geschützt ist. Die Platine hat einen USB-Eingang vom Typ B, den der Benutzer über ein Kabel mit seinem jeweiligen Handyladegerät verbinden muss, und einen weiteren USB-Eingang, an den der Benutzer sein Handy über ein entsprechendes Kabel anschließt.

7.2 Fazit

Durch die Auswahl geeigneter Hardware-Komponenten und die Erstellung einer funktionsfähigen Software für den Mikrocontroller ist ein Gerät entstanden, das die erwartete Funktionalität bietet. Im Rahmen dieser Arbeit wurden somit alle funktionalen Anforderungen an das Vorschaltgerät erfolgreich umgesetzt.

Als positiv lässt sich vor allem die Software bewerten, da diese auf einer gut strukturierten Architektur basiert und mit einer Größe von weniger als 3 Kilobytes als sehr klein

bezeichnet werden kann. Zusätzlich wurde die Software sehr ausführlich und übersichtlich dokumentiert. Hierfür wurde das Dokumentationstool Doxygen verwendet.

Bei der Hardware lässt sich Verbesserungspotential erkennen. Im Laufe des Projekts verlangsamte die Auswahl und Beschaffung geeigneter Hardware-Komponenten die Entwicklung. Dies hing zum einen mit der großen Anzahl von Optionen, zum anderen mit der COVID-19-Pandemie zusammen. Da der zeitliche Rahmen dieser Arbeit begrenzt ist, mussten hier die größten Einschränkungen hingenommen werden. Daher sollte eine genauere Analyse der Möglichkeiten ein primäres Ziel für zukünftige Realisierungen sein. Dabei sollten vor allem Preis und Größe einzelner Bauteile berücksichtigt werden, um das Gerät für den Anwender interessant zu machen.

Das betrifft vor allem das Bluetooth-Modul. Durch die Implementierung kleiner Hardware-Komponenten wurde das Ziel, ein möglichst kleines Gerät zu entwickeln, zwar erreicht, aber die Herausforderung, die Größe zu reduzieren, bleibt bestehen. Außerdem konnte die Anforderung, einen Preis von 10€ nicht zu überschreiten, nicht erfüllt werden. Dies ist vor allem auf das teure HC-06 zurückzuführen. Dieses ist ebenfalls bei der Größe der limitierende Faktor gewesen. Dementsprechend sollte für eine nutzerfreundlichere Hardware ein geeignetes, kleineres und kostengünstigeres Bluetooth-Modul gesucht werden.

Des Weiteren sollten mehr Messungen mit dem fertigen Gerät durchgeführt werden. Dies war aus Zeitgründen leider nicht in ausreichendem Umfang möglich. Zustellungsprobleme hatten den Prozess gegen Ende der Arbeit unerwartet verlangsamt. Zum Ausgleich konnten einige Tests mithilfe eines Breadboards durchgeführt werden. Für zuverlässige und umfangreichere Ergebnisse sollten die Tests jedoch mit dem fertigen Gerät wiederholt und erweitert werden.

Was das Design des Gehäuses betrifft, besteht die Herausforderung darin, ein Gehäuse zu bauen, das für das Auge des Benutzers attraktiv ist. Auch wenn dies in dieser Arbeit ebenfalls als erstrebenswert angesehen wurde, hatte der Aspekt keine hohe Priorität. Es könnte also noch mehr Aufwand in das Design des Gehäuses fließen. Idealerweise würde dabei Benutzerfeedback in den Prozess mit aufgenommen werden.

Auch die eingesetzte Beleuchtung spielt beim Design eine Rolle. Hierfür müsste untersucht werden, welche Lichtintensität für den Benutzer am angenehmsten ist, und diese sollte anschließend für die Zustandssignale verwendet werden.

7.3 Ausblick

Das Projekt, App und Vorschaltgerät zu entwickeln, fand im Rahmen zweier Abschlussarbeiten statt. Trotz der Aufteilung in zwei Teilprojekte besteht noch weiteres Entwicklungspotenzial. Dementsprechend lässt sich die gefundene Lösung verbessern und erweitern. Zum Beispiel indem die App für aktuelle Androidversionen weiterentwickelt wird und eine Version für iOS erstellt wird. Daneben müsste auch ein Weg gefunden werden, das Vorschaltgerät möglichst kostengünstig und dennoch so nachhaltig wie möglich zu produzieren, damit es tatsächlich genutzt werden würde.

Was die Verbesserung der Funktionalität des Geräts betrifft, muss es in der Zukunft möglich sein, dass der Benutzer die PIN des Bluetooth-Moduls ändern kann. Diese Funktionalität ist bereits in der Software implementiert. Der zu sendende Befehl ist „P“, gefolgt von der gewünschten Pin, zum Beispiel „P2020“. Die Software prüft auch, ob es sich um eine gültige PIN handelt. Allerdings muss bei der Implementierung dieser Funktionalität berücksichtigt werden, dass der Benutzer seine PIN vergessen kann. Daher muss eine Variante implementiert werden, welche die Rückkehr zum Werkszustand ermöglicht. Zusätzlich wurde dem Benutzer bisher nicht die Option gegeben, die LED zu deaktivieren. Weil das Licht Nutzer in einigen Situationen stören könnte, sollte diese Funktion für zukünftige Versionen des Vorschaltgeräts und der Smartphone-Applikation berücksichtigt werden.

Um die Lebensdauer des Smartphone-Akkus noch weiter zu verlängern, könnte der Ladestrom reduziert werden. Eine Möglichkeit wäre es, dass der Benutzer die Möglichkeit hat, den Strom zu regulieren, sodass z.B. nachts das Handy mit einem geringeren Strom geladen werden kann. Zu diesem Zweck könnte eine Stromregelung implementiert werden.

Um die Hardwareseite möglichst benutzerfreundlich zu gestalten, wäre ein nächster Schritt, das Vorschalt- und das Ladegerät in einem Gerät zu vereinen. Zum einen wäre die Nutzung komfortabler, zum anderen könnten dadurch weitere Nutzer dazugewonnen werden, indem das Vorschaltgerät nicht eine zusätzliche Anschaffung, sondern eine Alternative beim Kauf eines Ladegeräts darstellt. Hierbei müssten ebenfalls Kosten und Größe des Geräts gering gehalten werden.

Ein weiterer, sehr wichtiger Aspekt ist in diesem Falle jedoch ein nichttechnischer: Vielen Nutzern ist die Problematik des falschen Ladens überhaupt nicht bewusst, sodass bei

den Nutzern von Smartphones und darüber hinaus von Geräten mit Lithium-Akkus im Allgemeinen zuerst ein Bewusstsein für die Problematik geschaffen werden muss, um eine Veränderung zu erwirken. Dazu gehören zwei Aspekte: Zum einen das Bewusstsein über die ökologischen und sozialen Konsequenzen der Handyproduktion und Entsorgung und zum anderen das Bewusstsein über den richtigen Umgang mit vorhandenen Geräten, um eine möglichst hohe Lebensdauer zu erwirken. Dennoch spielt die Entwicklung technischer Hilfsmittel eine wichtige Rolle bei der Umsetzung. Dieses Projekt ist ein erster Schritt in diese Richtung.

Literaturverzeichnis

[1] Brandt, Mathias: Akku und Leistung haben Priorität. – URL https://de.statista.com/infografik/17153/funktionen-die-sich-verbraucher-fuer-ihr-naechstes-smartphone-wuenschen/. – Zugriffsdatum: 26.02.2019

[2] Chandrasekaran, Siddharth: Implementing Circular Buffer in C. May 2014. – URL https://embedjournal.com/implementing-circular-buffer-embedded-c/. – Zugriffsdatum: 28.11.2020

[3] Choi, Soo S. ; Lim, Hong S.: Factors that affect cycle-life and possible degradation mechanisms of a Li-ion cell based on LiCoO2. In: Journal of Power Sources 111 (2002), Nr. 1, S. 130–136

[4] Demir, Emrah: Applikation zur Verbesserung der Laderegelung in Smartphones, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 14.01.2020

[5] Diotec Semiconductor AG: Surface Mount General Purpose Si-Epi-Planar Transistors. April 2004

[6] Ebert, Christof: Systematisches Requirements-Engineering und Management Anforderungen ermitteln, spezifizieren, analysieren und verwalten. dpunkt-Verl., 2008. – ISBN 3898645460

[7] Esteban, Guillermo O.: CARACTERIZACIÓN DE BATERÍAS DE LITIO PARA EL ESTUDIO DEL ENVEJECIMIENTO, Universidad Politécnica de Madrid, Dissertation, 2019

[8] Fischl, Thomas: USBasp - USB programmer for Atmel AVR controllers. – URL https://www.fischl.de/usbasp/. – Zugriffsdatum: 26.11.2020

[9] Guangzhou HC Information Technology Co., Ltd: Product Data Sheet HC-06. Nov. 2006. – Rev. 2.0

[10] Hoffmann, Dirk W.: Software-Qualität. Springer-Verlag, 2013. – ISBN 978-3-642-35700-8

[11] IAR Systems: Code size limitation persist. – URL https://www.iar.com/support/tech-notes/general/code-size-limitation-persist/?_t_id=jvwKNsjoEpfQVRrG6M-1fw%3d%3d&_t_q=PC-locked+license&_t_tags=andquerymatch%2clanguage%3aen%2csiteid%3a0c16470a-f6d4-4850-8178-d6bc97e6f4ea%2ctab%3atechNotes&_t_hit.id=IAR_Find_Models_Indexing_SearchablePage/7cbf21fb-5e5d-40a0-92cd-302826830f14-en&_t_hit.pos=6. – Zugriffsdatum: 26.11.2020

[12] IAR Systems AB: C-SPY® Debugging Guide for Microchip Technology's AVR Microcontroller Family. April 2017. – Sixth edition

[13] JBL: Kurzanleitung $TUNE160^{BT}$. – URL https://de.jbl.com/on/demandware.static/-/Sites-masterCatalog_Harman/default/dw352a9812/pdfs/JBL_TUNE160BT_QSG_Multilingual.pdf. – Zugriffsdatum: 17.12.2020

[14] Jeckle, Mario ; Rupp, Chris ; Hahn, Jürgen ; Zengler, Barbara ; Queins, Stefan: UML 2 glasklar. In: Auflage. München. Carl Hanser (2012). ISBN 3446430571

[15] John, Alexander ; Meran, Renata ; Roenpage, Olin ; Staudter, Christian: Pugh-Matrix. S. 318–320. In: Six $Sigma^{+Lean}$ Toolset: Mindset zur erfolgreichen Umsetzung von Verbesserungsprojekten, Springer, 2014. – ISBN 9783662446133

[16] Jossen, Andreas ; Weydanz, Wolfgang: Moderne Akkumulatoren richtig einsetzen. Cuvillier Verlag: Göttingen, Germany, 2019. – ISBN 3939359114

[17] Kurzweil, Peter ; Dietlmeier, Otto K.: Elektrochemische Speicher. Springer, 2015. – ISBN 978-3-658-10900-4

[18] Leuthner, Stephan: Übersicht zu lithium-ionen-batterien. In: Handbuch Lithium-Ionen-Batterien. Springer, 2014, S. 13–19. – ISBN 978-3-642-30653-2

[19] Microchip Technology Inc.: AVR304: Half Duplex Interrupt Driven Software UART. – URL https:ww1.microchip.com/downloads/en/Appnotes/doc0941.pdf. – Zugriffsdatum: 9.11.2020

[20] Microchip Technology Inc.: Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash. Aug. 2013. – Rev. 2586Q

[21] Microchip Technology Inc.: 8-bit Atmel with 8KBytes In-System Programmable Flashh. Feb. 2013. – Rev. 2486AA

[22] Microchip Technology Inc.: 28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers with LCD Driver. Juni 2013. – Rev. C

[23] Nordmann, Julia ; Welfens, Maria J. ; Fischer, Daniel ; Nemnich, Claudia ; Bookhagen, Britta ; Bienge, Katrin ; Niebert, Kai: Perspektiven für eine nachhaltige Handynutzung. In: Die Rohstoff-Expedition. Springer, 2015, S. 63–72. – ISBN 978-3-662-44083-4

[24] ON Semiconductor: NDP6020P / NDB6020P P-Channel Logic Level Enhancement Mode Field Effect Transistor. Sep. 2017. – Rev. 3

[25] Panasonic: Connection Methods for PhotoMOS Relays. Juni 2015. – Application Note 036

[26] Pohl, Klaus ; Rupp, Chris: 6.3.2 Use Case Specifications. S. 67–69. In: Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam, Rocky Nook, 2015. – ISBN 193753877X

[27] Pohl, Klaus ; Rupp, Chris: Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam. Rocky Nook, 2015. – ISBN 193753877X

[28] Rahimzei, E ; Sann, K ; Vogel, M: Kompendium: Li-Ionen-Batterien–Grundlagen, Bewertungskriterien, Gesetze und Normen. In: BMWi Förderprogramm IKT für Elektromobilität II: Smart Car–Smart Grid–Smart Traffic (2015)

[29] Saad, Mustafa ; Farij, Abdalhalim ; Salah, Ahamed ; Abdaljalil, Abdalroof: Automatic street light control system using microcontroller. In: 14th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering, 2013

[30] Schmitt, Günter ; Riedenauer, Andreas: Mikrocontrollertechnik mit AVR: Programmierung in Assembler und C–Schaltungen und Anwendungen. Walter de Gruyter GmbH & Co KG, 2019. – ISBN 978-3-11-040388-6

[31] Slade, Mel ; Jones, Mark H. ; Scott, Jonathan B.: Choosing the right microcontroller: A comparison of 8-bit Atmel, Microchip and Freescale MCUs / Faculty of Engineering, The University of Waikato. 2011. – Forschungsbericht

[32] Starke, Gernot ; Hruschka, Peter: Software-Architektur kompakt:-angemessen und zielorientiert. Springer-Verlag, 2011. – ISBN 978-3-8274-2835-6

[33] Statista: Welche Arten von Schäden hatten Sie bisher schon an Ihren Smartphones?. – URL https://de.statista.com/statistik/daten/studie/663298/umfrage/umfrage-zu-schaeden-an-smartphones-in-deutschland/. – Zugriffsdatum: 12.09.2020

[34] Susnea, Ioan ; Mitescu, Marian: Microcontrollers in practice. Bd. 18. Springer Science & Business Media, 2005. – ISBN 978-3-540-28308-9

[35] Texas Instruments : LDO Basics. 2020. – SLYY151A

[36] Texas Instruments: LP2985 150-mA Low-noise Low-dropout Regulator With Shutdown. July 2013. – SLVS522O

[37] Texas Instruments: ULTRALOW-POWER 100-mA LOW-DROPOUT LINEAR REGULATORS. Juni 1999. – SLVS203E

[38] Urbanski, Klaus ; Woitowitz, Roland: Digitaltechnik: ein Lehr-und Übungsbuch. Springer-Verlag, 2012. – ISBN 978-3-642-20872-0

[39] van Heesch, Dimitri: Doxygen. – URL https://www.doxygen.nl/files/doxygen_manual-1.8.20.pdf.zip. – Zugriffsdatum: 8.12.2020

[40] Vishay Intertechnology, Inc.: BZX85-Series. Nov. 2017. – 85607 Rev. 2.1

[41] VISHAY SILICONIX: Power MOSFET Basics: Understanding the Turn-On Process. Juni 2015. – Application Note AN850

[42] Wunsch, Joerg: AVRDUDE. – URL https://www.nongnu.org/avrdude/user-manual/avrdude.html. – Zugriffsdatum: 26.11.2020

[43] Xu, Bolun ; Oudalov, Alexandre ; Ulbig, Andreas ; Andersson, Göran ; Kirschen, Daniel S.: Modeling of lithium-ion battery degradation for cell life assessment. In: IEEE Transactions on Smart Grid 9 (2016), Nr. 2, S. 1131–1140

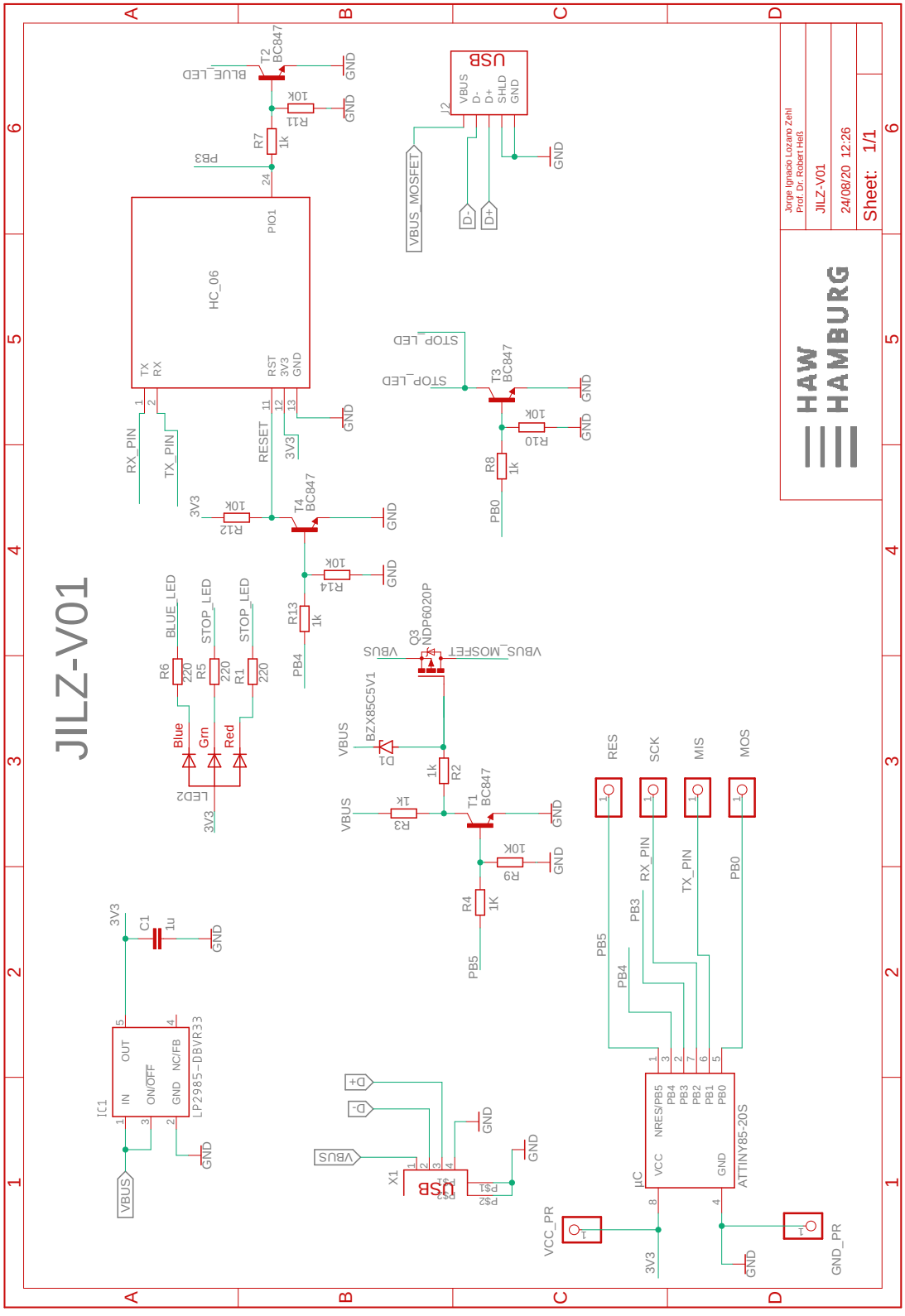
A Anhang

A.1 Stückliste

Projektname: **Vorschaltgerät**
Antragsteller: **Jorge Ignacio Lozano Zehl**

#Item	Menge	Hersteller	Package	Name auf der PCB	Description	Händler	Artikel-Nummer
1	1	Stewart Connector / Bel		X1	USB-Stecker USB 2.0 RA Receptacle Type B	Mouser Electronics	530-SS-52300-001
2	3	Tru Components	805	R6, R5,R1	Dickschicht-Widerstand 220 Ω SMD 0.125 W	Conrad Electronic	1584486 - 62
3	5	Tru Components	805	R14, R9 , R10, R11, R12	Dickschicht-Widerstand 10 kΩ SMD 0.125 W	Conrad Electronic	1585577 - 62
4	6	Tru Components	805	R3, R2, R4, R8, R7, R13	Dickschicht-Widerstand 1 kΩ SMD 0.125 W	Conrad Electronic	1585317 - 62
5	1	KEMET Corporation	603	C1	Keramik-Kondensator SMD 1 µF 25 V	Conrad Electronic	457913 - 62
6	1	Microchip	SOIC8	µC	ATTINY85-20SU Embedded-Mikrocontroller	Conrad Electronic	1266115 - 62
7	4	Diotec Semiconductor	SOT23	T1, T2, T3, T4	Bipolartransistor, NPN, 45V, 0,1A, 0,25W	Conrad Electronic	140540 - 62
8	1	Guangzhou HC Information Technology		BLU1	HC-06 Drahtloser Bluetooth-Transceiver Modul	Eckstein Komponente	CP06004
9	1	VISHAY	TO220	Q3	IRF9540PBF MOSFET P-Kanal 150 W	Conrad Electronic	597146 - 62
10	1	Texas Instruments	SOT-23-5	IC1	Spannungsregler 150-mA niedriges Rauschen 1,5% Toleranz	Mouser Electronics	595-LP2985-33DBVR
11	1	Kingbright			Standard-LEDs - Durchsteckmontage 5MM RGB LED	Mouser Electronics	604-WP154A43VBDZGWCA
12	1	RND connect			USB-Einbaubuchse, Typ A, gerade, Printmontage	Reichelt Elektronik	RND 205-00855
13	1	MPE-Garry			Stiftleisten 2,00 mm, 2X03, gerade	Reichelt Elektronik	MPE 150-3-006
14	1	VISHAY	DO-41	D1	Zener-Dioden 5.1 Volt 1.3 Watt 5%	Mouser Electronics	625-BZX85C5V1

A.2 Schaltplan



A.3 Doxygen Dokumentation

A.3.1 Latex Dokumentation

Doxygen generiert eine Dokumentation im Latex-Format, die sich auf der CD im Ordner Software/Dokumentation/Doxygen/Latex befindet.

A.3.2 Html Dokumentation

Die von Doxygen generierte Html-Dokumentation befindet sich auf der CD im Ordner Software/Dokumentation/Doxygen/Html, wo die Datei „index.html“ geöffnet werden muss, um die Seite zu öffnen, auf der sich die Code-Dokumentation befindet.

A.4 Eagle Dateien

Die von Eagle erzeugten Dateien befinden sich im Ordner „Hardware/Eagle“ der CD.

A.5 Programmcode

Der geschriebene Code befindet sich im Ordner „Software/Code“ der CD mit den folgenden Dateien:

Tab. A.1: Dateien-Liste

Name
ControlUnit.c
ModuleConfig.c
SoftwareUart.c
main.c
ControlUnit.h
ModuleConfig.h
SoftwareUart.h

A.6 HEX Datei

Im Ordner „Software/Hex“ auf der CD befindet sich die .hex-Datei, die das Programm enthält, das auf dem Mikrocontroller läuft.

A.7 STL Dateien

Im Ordner „3D-Modelle“ auf der CD sind die 3D-Modelle des Gehäuses sowie ein 3D-Modell der PCB gespeichert.

A.8 Datenblätter

Die Datenblätter befinden sich im Ordner „Hardware/Datenblätter“, in dem die Dokumentationen der folgenden Komponenten zu finden sind:

Tab. A.2: Liste der Datenblätter

Name	Beschreibung	Hersteller
BC847	Bipolartransistor	Diotec Semiconductor
NDP6020P	MOSFET P-Kanal	ON Semiconductor
ATTiny85	Embedded-Mikrocontroller	Microchip
HC-06	Bluetooth-Transceiver Modul	Guangzhou HC Information Technology
LP2985-DBVR33	Spannungsregler	Texas Instruments
BZX85C5V1-TR	Z-Diode	Vishay

Abbildungsverzeichnis

Tabellenverzeichnis

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original