
Bachelorarbeit

Energieautarkes NFC-basiertes Authentifizierungssystem

im Studiengang Technische Informatik
der Fakultät Informationstechnik
Sommersemester 2013

Matthias Singer
Matrikelnummer: 735098
E-Mail: matthias.singer@gmx.net

Zeitraum: 01.04.2013 - 26.08.2013
1. Prüfer: Prof. Dr.-Ing. Andreas Rößler
2. Prüfer: Prof. Dr.-Ing. Reinhard Schmidt

Firma: Fraunhofer-Institut für Arbeitswirtschaft und Organisation
Abteilung: KEIM
Betreuer: M.Sc. B.Eng. Vikas Agrawall

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Esslingen, den 25. August 2013

Matthias Singer

Inhaltsverzeichnis

Abkürzungsverzeichnis	i
Symbolverzeichnis	ii
Abbildungsverzeichnis	iii
1 Einleitung	1
1.1 Motivation	1
1.2 Aufgabenbeschreibung	2
2 Grundlagen und Stand der Technik	3
2.1 Near Field Communication	4
2.1.1 Technologie der Near Field Communication	4
2.1.2 Passiver Kommunikationsmodus	4
2.1.3 Aktiver Kommunikationsmodus	5
2.1.4 Das NDEF-Datenaustauschformat	6
2.1.5 NFC-Antenne	7
2.1.6 NFC Energieübertragung auf dem Prinzip des Transformators	7
2.1.7 Elektrischer Schwingkreis	8
2.1.8 Datenübertragung von NFC mit Modulationsverfahren Amplitude Shift Keying (ASK)	9
2.2 Physikalische Grundlagen induktiv gekoppelter Systeme	10
2.2.1 Magnetisches Feld	10
2.2.2 Magnetische Spannung	10
2.2.3 Magnetische Feldstärke	10
2.2.4 Kopplung	10
2.2.5 Induktion	10
2.3 Mikrocontroller	12
2.3.1 Harvard-Architektur	12

Inhaltsverzeichnis

2.3.2 I ² C Bus	12
2.4 Beschreibung des M24LR-Discovery Board Kit	14
2.4.1 Funktionsweise des M24LR Board	15
2.4.2 Beschreibung des Mikrocontrollers <i>STM8L152C6</i>	16
2.4.3 Beschreibung des NFC EEPROM	16
2.5 Gleichrichterschaltungen	17
2.5.1 Zweiweg Graetz-Gleichrichter mit Glättung	17
2.5.2 Spannungsverdopplung mit der Delonschaltung	18
2.5.3 Spannungsvervielfachung durch Kaskadierung der Villard-schaltung	19
3 Inbetriebnahme der Hardware	20
3.1 Das M24LR-Discovery Board	20
3.2 RF Transceiver Demonstration Board	21
3.3 Android App NFC-V Reader	22
4 Entwicklung der eingebetteten Authentifizierungssoftware	23
4.1 Analyse	24
4.2 Entwurf	25
4.3 NDEF Message Daten im NFC EEPROM	26
4.4 Implementierung des byteweisen Auslesens der Nutzdaten	26
4.5 Im Zustand geschlossen einen Pin erhalten und vergleichen	31
4.6 Implementierung der Zustände 'S_GEÖFFNET' und 'S_GESCHLOSSEN'	32
4.7 Die Funktion Text in das Display schreiben	33
5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem	35
5.1 Energy Harvesting	36
5.2 Bewertung geeigneter physikalischer Effekte für das Energie Harvesting für den Betrieb eines Schlosses	36
5.3 Versuch: Graetz Zweiweggleichrichter	36
5.4 Versuch: Spannungsverdopplung durch Delonschaltung	38
5.5 Spannungsvervielfachung mit Schwingkreis und Delonschaltung	39
5.5.1 Dimensionierung eines Schwingkreises	39

Inhaltsverzeichnis

5.6	Versuch: Schwingkreis mit Delonschaltung	43
5.6.1	Anwendung zweier Spulen	44
5.7	Weitere Möglichkeiten des Energy Harvesting in Abhängigkeit von der Umgebung	45
5.7.1	Funktionsprinzip eines Piezokristalls	45
5.7.2	Einsatz von Piezokristallen als Energielieferant	46
5.8	Einsatz von weiteren Energiequellen	47
6	Zusammenfassung und Ausblick	48
6.1	Zusammenfassung	48
6.2	Ausblick	49
	Literaturverzeichnis	50
	Listingverzeichnis	52
	Anhang	53

Abkürzungsverzeichnis

ACK	Acknowledgement
ASK	Amplitude Shift Keying
CF	Chunk Flag
DC	Direct Current
DIFS	Distributed Coordination Function Interframe Spacing
ECMA	European Computer Manufacturers Association
EEPROM	Electrically Erasable Programmable Read-only Memory
GND	Ground
I²C	Inter-Integrated Circuit
IC	Integrated Circuits
IEC	International Electrotechnical Commission
IL	ID Length Present
ISO	International Organization for Standardization
JTAG	Joint Test Action Group
MB	Message Begin
ME	Message End
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
Pin	Personal Identifikation Number
RFID	Radio Frequency Identification
SCL	Serial Clock
SDA	Serial Data Line
SPI	Serial Peripheral Interface
SR	Short Record
SRAM	Static Random-Access Memory
TNF	Type Name Format
USART	Universal Synchronous and Asynchronous Serial Receiver and Transmitter
VCC	Positive Supply Voltage

Symbolverzeichnis

Ω	elektrischer Widerstand, <i>Ohm</i> $\Omega = \frac{kg \cdot m^2}{A^2 \cdot s^3}$
μ_0	magnetische Feldkonstante, $\mu_0 = 4 \cdot \pi \cdot 10^{-7} \frac{m \cdot kg}{s^2 \cdot A^2}$
\vec{H}	vektorielle Größe der magnetischen Feldstärke, $\vec{H} = \frac{A}{m}$
k	Kopplungsfaktor, dimensionslos
C	elektrische Kapazität, <i>Farad</i> $F = \frac{A^2 \cdot s^4}{kg \cdot m^2}$
L	Induktivität, <i>Henry</i> $H = \frac{kg \cdot m^2}{A^2 \cdot s^2}$
f	Frequenz, <i>Herz</i> $Hz = \frac{1}{s}$
P	elektrische Leistung, <i>Watt</i> $W = \frac{kg \cdot m^2}{s^3}$
U	elektrische Spannung, <i>Volt</i> $V = \frac{kg \cdot m^2}{A^2 \cdot s^3}$
I	elektrische Stromstärke, <i>Ampere</i> A Si-Basiseinheit
N	Wicklungszahl, dimensionslos
G	elektrischer Leitwert, <i>Siemens</i> $S = \frac{A^2 \cdot s^3}{kg \cdot m^2}$

Abbildungsverzeichnis

2.1	Aufbau eines <i>NDEF</i> -Record	6
2.2	Aufbau einer <i>NDEF</i> -Nachricht	6
2.3	<i>NFC Smartphone Antenne</i>	7
2.4	Prinzip eines Transformators	7
2.5	Amplitude Shift Keying	9
2.6	<i>M24LR-Discovery Kit</i>	14
2.7	<i>M24LR-Discovery Board</i>	15
2.8	<i>M24LR04E-R EEPROM</i>	16
2.9	Aufbau eines Brückengleichrichters mit Glättung	17
2.10	Spannungsverlauf bei Graetz-Gleichrichter	17
2.11	Aufbau einer Delonschaltung	18
2.12	Spannungsverlauf einer Delonschaltung	18
2.13	Kaskade einer Villardschaltung	19
2.14	Spannungsverlauf einer Villardschaltung	19
3.1	<i>M24LR-Discovery</i> und <i>ST-LINK/V2</i>	21
3.2	<i>NFC SD-Karte</i>	22
4.1	Zustandsdarstellung der Authentifizierung	25
4.2	Programmablaufdiagramm byteweises lesen	28
4.3	Programmablaufplan Funktion <i>Copy_Data_NFC_EE_to_Int_EE()</i> . .	30
4.4	Programmablaufplan der Zustand GEOEFFNET/GESCHLOSSEN .	33
5.1	Zweiweggleichrichter P_{Max}	37
5.2	Villardschaltung P_{Max}	39
5.3	Testschaltung zur Ermittlung des Schwingkreises	41
5.4	Schwingkreis mit Delonschaltung	42
5.5	Leistungsverlauf bei Schwingkreis mit Delonschaltung P_{Max} . . .	43
5.6	Aufbau eines Piezostapeltranslators	45
5.7	Vibrationsdiagramm Schienenfahrzeug	47
5.8	Energie Harvesting Windkraft	47
6.1	Anhang: <i>Amplitude Shift Keying</i>	62
6.2	Anhang: <i>M24LR-Discovery</i> mit externer Spule	62
6.3	Anhang: Gelötete Greatz-Gleichrichterschaltung	63
6.4	Anhang: Gelötete Delonschaltung	63

1 Einleitung

1.1 Motivation

Im Jahre 2002 wurde von *Sony* und *NXP Semiconductors* eine auf *RFID* basierende Funktechnik entwickelt. Vorteile dieser Technik sind die geringe Reichweite von maximal 10 Zentimetern, welche Sicherheit gegenüber Dritten bietet und die Integration zweier bisher strikt voneinander getrennten Funktionen, dem Lesegerät und dem Transponder. So kann ein mobiles Gerät entweder als Lesegerät oder als kontaktlose Chipkarte emuliert werden. Da es auf der *RFID*-Technik basiert, ist *Near Field Communication (NFC)* auch immer abwärtskompatibel und kann auch *RFID*-Chips lesen und beschreiben. *Near Field Communication* ermöglicht es, sich mit mobilen Geräten (Smartphones) auszuweisen, bargeldlos zu bezahlen, Tickets zu kaufen oder auch den Austausch kleinerer Datenmengen zwischen zwei Smartphones oder einem *Tag* zu tätigen. Dabei soll die *Near Field Communication* immer mehr Karten ersetzen und so als eine universelle Chipkarte genutzt werden können. Sogar die Personalausweise, welche die Bundesrepublik Deutschland seit dem 1. Nov. 2010 ausstellt, basieren auf der *RFID*-Technik. Immer mehr Smartphones werden mit *Near Field Communication* ausgestattet. *NFC*-Systeme sind induktiv gekoppelte Systeme. Dies stellt die Voraussetzung für ein energieautarkes System dar. Im Zusammenhang mit dem Energy Harvesting ist dies eine interessante Technik, welche diese Geräte unabhängig von Infrastruktur und Wartung macht und somit kostengünstig in Betrieb und Installation sind.

1.2 Aufgabenbeschreibung

Ziel dieser Arbeit ist es, ein Entwicklungsboard der Firma *STMicroelectronics* in Betrieb zu nehmen und darauf eine Authentifizierungssoftware zu implementieren. Dabei soll ein über *Near Field Communication* übertragener Pin (Personal Identification Number) zur Identifikation benutzt werden. Der Pin soll frei wählbar sein und das System mit diesem geschlossen werden. Dem Smartphone, mit entsprechender Identifikationsnummer, wird der Zugang gewehrt. Stimmt die Identifikationsnummer jedoch nicht überein, so soll der Zugang verwehrt bleiben. Des Weiteren soll untersucht werden, wie viel Leistung aus der Induktion der *Near Field Communication* bereitgestellt werden kann. Dazu wird mit verschiedenen Gleichrichterschaltungen experimentiert. Letzten Endes sollen in dieser Arbeit die Grundlagen erarbeitet werden, um ein Schließfach energieautark betreiben zu können. Dies bedeutet, es soll ohne Anschließen an eine vorhandene Strominfrastruktur funktionieren.

2 Grundlagen und Stand der Technik

In diesem Kapitel werden die Grundlagen behandelt, die zum Verständnis der *Near Field Communication*, der Übertragungstechnik, das Übermitteln von Energie und der verwendeten Hardware, beitragen.

2.1 Near Field Communication

Near Field Communication (NFC) Technologie bedeutet übersetzt Nahfeldkommunikation. Mit ihr lassen sich kontaktlos Nachrichten über kurze Distanzen austauschen. Die *Near Field Communication* basiert auf einer schon mehrere Jahrzehnte alten Technologie, bekannt als *RFID*¹. Der Vorläufer der *RFID*, *Radio Frequency Identifikation*, wurde im Zweiten Weltkrieg zur Erkennung der eigenen und befreundeten Truppen eingesetzt. In den folgenden Nachkriegsjahren wurden immer mehr Produktionsprozesse mit der *RFID*-Technik optimiert. In den 1970er Jahren wurde die *RFID*-Technik das erste Mal für elektronische Warenaufbereitungssysteme eingesetzt. Diese kennt man heute noch von den Eingangsbereichen einiger Kaufhäuser. Bis heute werden viele Systeme durch die *RFID* effizienter, bequemer oder erst möglich gemacht. Einige davon sind z.B. Mautsysteme, Tieridentifikation, elektronische Zutrittskontrollsysteeme, Wegfahrsperren im KFZ und Bezahlsysteme. [1, Seite 4], [2]

2.1.1 Technologie der Near Field Communication

Die *Near Field Communication* arbeitet im Frequenzbereich von 13.56 MHz. Bei dieser Technologie ist die klare Trennung in Transponder und Lesegerät aufgehoben. Während bei *RFID*-Systemen die aktiven und passiven Module von Anfang an festgelegt sind, können bei der *NFC*-Technologie die Geräte abwechselnd das passive und aktive Modul sein. *NFC*-Systeme werden ausschließlich durch die induktive Kopplung betrieben. Daher sind *NFC*-Geräte in der Lage kontaktlose Chipkarten und *Tags* mit Energie zu versorgen oder eine kontaktlose Chipkarte zu emulieren. Die *NFC*-Kommunikation basiert in beiden Fällen auf den bestehenden Standardprotokollen der ISO/IEC 18092 bzw. ECMA-340. Diese geht aus der Norm ISO/IEC 14443 hervor. Für die Verbindung der physikalischen und digitalen Verarbeitung ist ein *NFC-IC* notwendig. Zusätzlich ist ein *Smartcard-Mikrochip* im *NFC*-Gerät zur Emulation einer kontaktlosen Chipkarte notwendig. [1, Seite 6]

2.1.2 Passiver Kommunikationsmodus

Im passiven Kommunikationsmodus gibt es einen Initiator und ein *Target*-Gerät. Die *NFC*-Komponente vom *Target* wird durch die hochfrequente Induktion des Initiators mit Energie versorgt. Im passiven Kommunikationsmodus hat der Initiator

¹radio-frequency identification

einen erhöhten Energieverbrauch, das *Target* dagegen muss evtl. nur Energie für die weitere Verarbeitung des *Peer-to-Peer*-Protokollstapels aufbringen, da die Übertragungsgenergie vom Initiator aufgewendet werden muss. Die Übertragung im passiven Kommunikationsmodus wird vom Initiator mittels *ASK*² (Kapitel 2.1.8) über das Trägersignal moduliert. Zum Aufbau einer Verbindung wird das *NFC*-Gerät, das standardmäßig immer im *Target*-Modus konfiguriert ist, in den Initiator-Modus umgeschaltet. Zunächst überprüft das *NFC*-Gerät, ob ein externes Trägersignal vorhanden ist. Sobald das Medium für die Dauer eines *DIFS*³ (*Distributed Coordination Function Interframe Spacing*) frei ist, wird die Übertragung vom Initiator gestartet. Dabei wird zunächst die Übertragungsgeschwindigkeit, durch die Initialisierungs- und Antikolissionsprozedur *Sense Request* oder *Polling Request*, festgelegt. Durch die *Single Device Detection (SDD)* wird ein passives *NFC*-Gerät ausgesucht, mit dem dann eine *Peer-to-Peer*-Kommunikation aufgebaut wird. Der Datenaustausch wird vom Initiator durch den Aktivierungsbefehl (*Attribute Request*) eingeleitet. [1, Seite 92 - 93] [3] [4] [5]

2.1.3 Aktiver Kommunikationsmodus

Der aktive Kommunikationsmodus zeichnet sich dadurch aus, dass sich die beteiligten *NFC*-Geräte selbständig mit Energie versorgen. Dadurch ist die Energieaufwendung für den Initiator im aktiven Kommunikationsmodus deutlich geringer als im passiven Kommunikationsmodus. Die Energie für das Sendesignal wird von beiden Teilnehmern selbst aufgebracht. Im aktiven Kommunikationsmodus wird dasselbe *ASK*-Modulationsverfahren für beide Richtungen, wie beim passiven Modus, angewendet. Zunächst wird vom Initiator *Collision Avoidance* durchgeführt. War dies erfolgreich, wird vom Initiator ein *Attribute Request* eingeleitet und dieser schaltet anschließend das Trägersignal ab. Mit dem *Attribute Request* werden Informationen über die Übertragungsgeschwindigkeit und das verwendete Bitübertragungsprotokoll an die *Target*-Geräte übermittelt. Nun führen alle *Target*-Geräte das *Collision Avoidance* durch, um daraufhin den *Attribute Response* senden zu können. Die Wartezeit der *Targets*, um senden zu können, wird durch eine Zufallszahl vom *Target* bestimmt. Um Kollisionen zu vermeiden erzeugen die *Targets* Zufallszahlen, welche die Wartezeit bestimmen. Das Gerät mit der kleinsten Zufallszahl antwortet als erstes. Die Geräte mit höherer Zufallszahl erkennen das Sendesignal des Gerätes mit der kleineren Zufallszahl und bleiben daraufhin stumm. Kommt es zufällig dazu, dass zwei *Target*-Geräte die gleiche Zufallszahl haben, führt dies zu einer Kollision. Dies erkennt der Initiator und beginnt erneut mit der Aktivierungprozedur. Ist die Aktivierungsprozedur erfolgreich abgeschlossen, wird vom Initiator ein *Target* zur *Peer-to-Peer*-Kommunikation ausgewählt. [1, Seite 93 - 94]

²Amplitude Shift Keying

³Unter DIFS versteht man die Mindestzeit, die vor dem Senden eines Frames vergangen sein muss.

2.1.4 Das NDEF-Datenaustauschformat

Die Androidanwendung (App) von *STMicroelectronics* *NFC-V reader* versendet den Pin als *NDEF*-Nachricht. Das in dieser Arbeit verwendete *M24LR-Discovery Board* hat direkt an der Antenne ein

NFC fähiges *EEPROM*. Dieses verhält sich wie ein passives Speicherelement, auch *Tag* genannt. Damit diese *Tags* mit Geräten verschiedener Hersteller, ohne proprietäre Anwendungen, kommunizieren können, wurde vom *NFC-Forum*⁴ das *NDEF* Format eingeführt. *NDEF* bedeutet *NFC Data Exchange Format* und definiert den Aufbau der Datenstruktur, der zum Austausch zwischen *NFC*-Geräten und *Tags* spezifiziert wurde.

Eine *NDEF*-Nachricht kann aus mehreren *NDEF-Records* bestehen. Ein *NDEF-Record* besteht aus einem Datenteil und einem *Header*. Der *Header* eines *NDEF-Records* hat fünf *Flags*. So wird im *Flag* MB und ME der erste und letzte *Record* in einer *NDEF-Message* bekanntgegeben. Neben weiteren *Flags* wird in 1 Byte großen Feldern unter anderem auch die Nutzdatenlänge (*Payload Length*) des Feldes der Nutzdaten (*Payload*) angegeben. Das Format der Nutzdaten entspricht der Formatangabe im *Type-Feld*. [1, Seite 120 - 122]

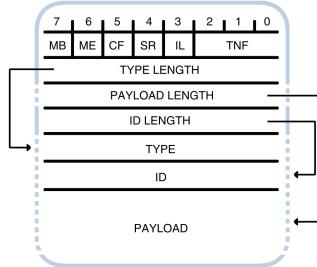


Abbildung 2.1: Aufbau eines *NDEF-Records*.

Flags: MB (*Message Beginn*), ME (*Message End*), CF (*Chunk Flag*), SR (*Short Record*), IL (*ID Length Present*), TNF (*Type Name Format*)

[1, Seite 121]

NDEF Message

R₁ MB = 1 ... R_X ... R_n ME = 1

Abbildung 2.2: Aufbau einer *NDEF*-Nachricht aus mehreren *NDEF-Records*. Zu sehen ist hier, dass das *Message Beginn Flag* nur bei dem ersten *Record* gesetzt ist, das *Message End Flag* ist nur bei dem letzten *Record* gesetzt.

[1, Seite 122]

⁴*NFC-Forum*: Konsortium, das aus Mitgliedern von ca. 100 Firmen besteht und sich um die Architektur und Spezifikationen der *NFC*-Technik kümmert

2.1.5 NFC-Antenne

Die *NFC*-Antenne muss nach dem Prinzip einer Spule aufgebaut sein. Am häufigsten werden in Smartphones *NFC*-Antennen mit einer Wicklungszahl von $N=4$ Wicklungen eingesetzt. (siehe Abbildung 2.3)

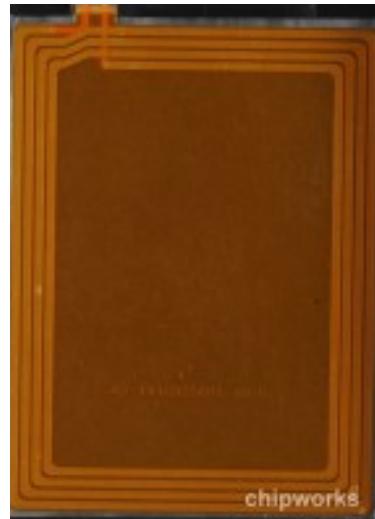


Abbildung 2.3: induktive *NFC*-Antenne des Smartphones Samsung Galaxy Nexus [6]

2.1.6 NFC Energieübertragung auf dem Prinzip des Transformators

Bei der *NFC*-Technologie gibt es die Möglichkeit, wie in Kapitel 2.1.2 beschrieben, ein *Target* mit Energie zu versorgen. Dazu wird das Prinzip eines Transformators angewendet. Wird in einer Spule, in diesem Falle in einer Smartphone *NFC*-Antenne, eine

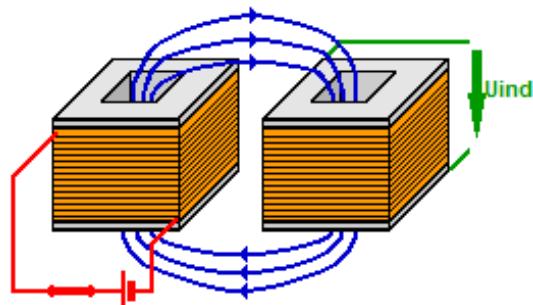


Abbildung 2.4: Prinzip eines Transformators ohne Eisenkern [7]

schnelle Stromänderung bewirkt, so wird auch eine Flussänderung des magnetischen Feldes mit gleicher Frequenz bewirkt. Ist nun eine Sekundärspule idealerweise direkt darüber positioniert, so wird in dieser Spannung induziert. Der Sekundärspule⁵ wird eine Kapazität C parallel geschaltet, dadurch erhält man einen Parallelschwingkreis. Da die Kapazität auf niedrige Frequenzen kaum Auswirkungen hat, bleibt die Ausgangsspannung gleich der Eingangsspannung. Geht die Sendefrequenz an die Resonanzfrequenz, so wird die induzierte Spannung überhöht. Bei Frequenzen sehr viel größer als die der Resonanzfrequenz wirkt der Kondensator C dämpfend und die Ausgangsspannung wird dadurch niedriger. Letzten Endes ist dieser Schwingkreis notwendig um den Transponder mit genügend Spannung zu versorgen, da der Kopplungsfaktor durch die Lage und den Abstand bei NFC-Systemen meist $k \ll 1$ ist. In Einzelfällen kann die induzierte Spannung auch einige hundert Volt erreichen. Die induzierte Spannung wird nun über einen Gleichrichter in Gleichspannung umgewandelt. Die heute üblichen Arbeitsspannungen von ICs (integrierte Schaltungen) liegen zwischen 0,8 V und 5 V. Das wiederum macht eine Spannungsbegrenzung notwendig. [1, Seite 19]

2.1.7 Elektrischer Schwingkreis

Ein elektrischer Schwingkreis besteht aus einer Spule und einem Kondensator. Es gibt zwei Arten von Schwingkreisen, Reihenschwingkreis und Parallelschwingkreis. Im Schwingkreis wird die elektrische Ladung periodisch zwischen dem Kondensator und der Spule ausgetauscht, dabei findet ein Wechsel zwischen elektrischer und magnetischer Energie statt. [8, Seite 433]

Formel zur Berechnung der Resonanzfrequenz eines elektrischen Schwingkreises:

$$f_r = \frac{1}{(2 \cdot \pi \cdot \sqrt{L_2 \cdot C_2})} \quad (2.1)$$

Resonanzfrequenz der Near Field Communication

$$f_r = 13,56 \text{ MHz} \pm 2 \text{ kHz}$$

[1, Seite 20]

⁵ Als Sekundärspule wird die Spule bezeichnet, welche aus dem magnetischen Fluss induzierte Spannung bekommt. Diejenige, die den magnetischen Fluss verursacht wird als Primärspule bezeichnet.

2.1.8 Datenübertragung von NFC mit Modulationsverfahren Amplitude Shift Keying (ASK)

Im vorherigen Kapitel 2.1.6 wurde beschrieben, wie die Energieübertragung funktioniert. Es wird eine sinusförmige Spannung mit 13,56 MHz induziert. Damit digitale Werte wie eine ‘1’ und ‘0’ übertragen werden können, wird bei NFC das zweistufige Modulationsverfahren *Amplitude Shift Keying* angewendet. So wird eine digitale ‘0’ mit einer geringeren Amplitude, welche 10 % kleiner ist als die eigentliche Höhe der Amplitude, dargestellt. Eine digitale ‘1’ bedeutet 100 % Höhe der Amplitude.

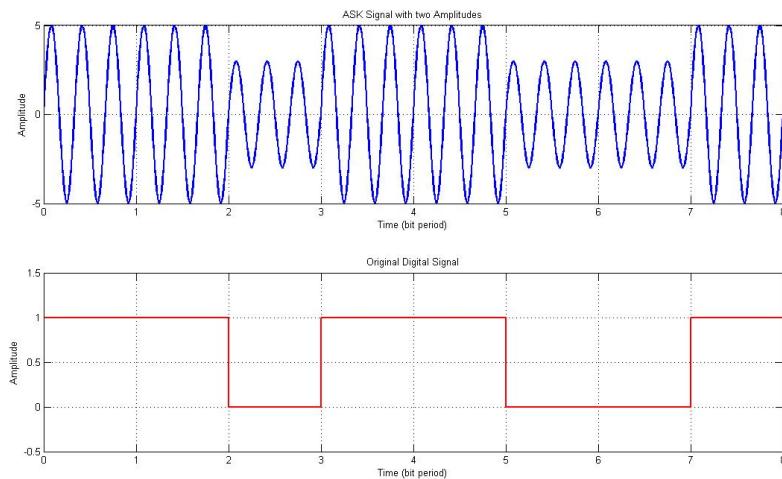


Abbildung 2.5: Im oberen Diagramm ist das *Amplitude Shift Keying* mit dem Modulationsindex von 10% zu sehen. Im unteren Diagramm ist der binäre Wert, der im oberen Diagramm mit ASK dargestellt wird, zu sehen.

[9]

2.2 Physikalische Grundlagen induktiv gekoppelter Systeme

2.2.1 Magnetisches Feld

Wird ein elektrischer Leiter mit Strom (elektrische Ladung) durchflossen, so bewirkt dies ein magnetisches Feld. Das magnetische Feld eines stromdurchflossenen Leiters kann durch ein Vektorfeld im Raum anschaulich gemacht werden. Dabei gibt die Tangente an der Feldlinie immer die Richtung der Kraft an. Die magnetische Kraft unterscheidet sich grundlegend von der Coulomb-Kraft bezüglich ihrer Stärke, Richtung und der Entstehung. [8, Seite 351 - 354]

2.2.2 Magnetische Spannung

Die magnetische Spannung U_m in einem geschlossenen Leiter mit dem Weg s ist die Summe aller Ströme I_n . Mit der Summe aller Ströme ist hier die Anzahl der stromdurchflossenen Leiter gemeint. So ist bei einer Spule mit drei Wicklungen und einem Strom von 5 mA die Summe der Ströme 15 mA. [1, Seite 14]

$$U_m = \sum_n I_n$$

2.2.3 Magnetische Feldstärke

Die magnetische Feldstärke ergibt sich aus dem magnetischen Feld und der Anordnung des stromdurchflossenen Leiters. Die Stärke der magnetischen Feldlinie wird mit \vec{H} bezeichnet und hat die Einheit A/m. Die Stärke ist abhängig von dem Strom I im Leiter und der Entfernung zu diesem (Radius r). [8, Seite 351 - 354]

2.2.4 Kopplung

Der Kopplungsfaktor gibt Aufschluss darüber, wie sich zwei benachbarte Spulen magnetisch beeinflussen. Dabei gibt der Kopplungsfaktor an, inwiefern sie vom selben magnetischen Fluss durchsetzt sind. Für $k = 0$ gibt es keine Kopplung und für $k = 1$ sind beide Spulen mit demselben magnetischen Fluss durchsetzt. [1, Seite 18]

$$k = \frac{M}{\sqrt{L_1 \cdot L_2}} \quad (2.2)$$

2.2.5 Induktion

Ist eine Spule in einem, sich über die Zeit verändernden magnetischen Feld, so entsteht am Spulenausgang eine induzierte Spannung U_{ind} . [8, Seite 383 - 384] [1, Seite 18]

Spezielle Werte zur Berechnung der Induktivität einer rechteckigen Luftspule:

$K_1 = 2,34; K_2 = 2,75 \rightarrow$ Für eine rechteckige Luftspule [10, Seite 12]

d_{out} = äußerster Leiter

d_{in} = innerster Leiter

$$d = \frac{d_{out} + d_{in}}{2} \quad \text{in mm} \quad (2.3)$$

$$p = \frac{d_{out} - d_{in}}{d_{out} + d_{in}} \quad \text{in mm} \quad (2.4)$$

Magnetische Feldkonstante:

$$\mu_0 = 4\pi \cdot 10^{-7} \frac{N}{A^2} \rightarrow \text{magnetische Feldkonstante}$$

$L_{antenne} \rightarrow$ Induktivität einer Rechteckspule \rightarrow Maßeinheit Henry [H]

$$L_{antenne} = K_1 \cdot \mu_0 \cdot N^2 \cdot \frac{d}{1 + K_2 \cdot p} \quad (2.5)$$

2.3 Mikrocontroller

2.3.1 Harvard-Architektur

Der verwendete Mikrocontroller *STM8L152C6* basiert auf der Harvard Architektur. Die Harvard Architektur ist bekannt als ein Schaltungskonzept um sehr schnelle Prozessoren realisieren zu können. Bei der Harvard Architektur ist der Programm- codespeicher logisch und physisch von dem Datenspeicher getrennt und über zwei verschiedene Busse verbunden. Der Vorteil ist, dass Daten und Programmspeicher zur gleichen Zeit geladen werden können. Nachteilig ist, dass der Speicher stark fragmentiert wird. [11]

2.3.2 I²C Bus

Für diese Bachelorarbeit ist es wichtig sich mit den Grundlagen des *I²C-Bus* zu beschäftigen, da dieser für die Kommunikation zwischen dem Mikrocontroller *STM8L152C6* und dem *EEPROM M24LR04E* verantwortlich ist.

Der *I²C-Bus* wurde von *NXP Semiconductors* in den 80er Jahren entwickelt. *I²C* steht für *IIC* was ausgeschrieben *Inter-Integrated Circuit* bedeutet. Der *I²C-Bus* wird häufig benutzt um verschiedene Peripherien, die ein Mikrocontroller enthält, kommunikativ zu verbinden. Aber auch, um zwischen dem auf einer Platine angebrachten Mikrocontroller, den Speicherbausteinen und den Sensoren kommunizieren zu können. *I²C* ist ein *Master-Slave* Bussystem, es sind aber auch mehrere *Master* möglich (Multimastersystem). Ein *Slave* kann nur senden, wenn er von einem *Master* dazu berechtigt wird. In diesem Falle ist der *STM8L152C6* der *Master* und das *EEPROM M24LR04E* der *Slave*. Ein *I²C* Bussystem besteht aus mindestens einem *Master* und einem *Slave*, die über zwei Signalleitungen verbunden sind. Auf der einen Leitung wird der Takt (*SCL*) übertragen, auf der anderen die Daten (*SDA*). Beide Leitungen werden über eine Versorgungsspannung (*V_{DD}*) versorgt. Für den *High*-Pegel muss die Spannung mindesten $0,7 \times V_{DD}$ Volt betragen, für den *Low*-Pegel darf die Spannung $0,3 \times V_{DD}$ Volt nicht überschreiten. Der *I²C-Bus* arbeitet mit positiver Logik, d.h. wenn der Pegel auf der Datenleitung *High* ist, so stellt dies eine logische ‘1’ dar. Ist der Pegel auf *Low*, wird eine logische ‘0’ übertragen. Der Bus ist in Ruhestellung auf *High* Pegel. Der *I²C-Bus* kann in vier Bus-Takt-Modi betrieben werden. Diese lassen dann eine maximale Bus Taktrate von 100 kHz im *Standard Mode* und 3,4 MHz im *High Speed Mode* zu. Der *I²C-Bus* auf dem *M24LR-Discovery Board* läuft im *Fast Mode*, d.h. mit einer Taktfrequenz von 400 kHz. [12] [13]

Bitübertragung: Auf der Datenleitung *SDA* darf der Pegel nur toggeln, solange

auf der Takteleitung *SCL* ein *Low*-Pegel anliegt. Ist *SCL* im Zustand *High*, so darf sich bei *SDA* nichts ändern. [13]

Startbedingung: Der *Master* sendet zum Bekanntgeben einer Datenübertragung einen *High/Low*-Pegelwechsel auf der Datenleitung *SDA*, ohne eine Taktung auf der *SCL*-Leitung. Nach Senden der Startbedingung beginnt der *Master* mit der Taktung. Möchte der *Master* eine andere Adresse oder den Bus weiterhin belegen, so sendet er erneut eine Startbedingung ohne eine Stoppbedingung zu senden.

Stoppbedingung: Die Stoppbedingung wird vom *Master* durch eine *Low-High*-Pegelwechsel auf *SDA* bei beendeter Taktung auf der *SCL*-Leitung definiert.

Acknowledgment: Der *Slave* bestätigt den Empfang eines Bytes mit einem *ACK*-Bit. Das *ACK*-Bit des *Slaves* wird nur gesetzt, wenn der *Slave* Daten aufnehmen bzw. empfangen kann. Zum Bestätigen einer Nachricht wird die Datenleitung während des neunten *Clock*-Pulses der *SCL*-Leitung auf *Low* gezogen. [13] [12]

2.4 Beschreibung des M24LR-Discovery Board Kit

Das *M24LR-Discovery Board* Kit besteht aus dem *M24LR Board* siehe Bild 2.7 und dem *RF Transciever Board* siehe Abbildung 2.6. Das *RF Transceiver Board* wird als aktiver Initiator über die PC Software gesteuert und verschickt Nachrichten im *NDEF*-Format. Das *M24LR Board* ist nur für den Empfang von Nachrichten ausgelegt und kann nicht aktiv senden. Auf dem *M24LR Board* ist ein LowPower Mikrocontroller *STM8L152C6*, ein *NFC EEPROM*, ein *SWIM Connector* und eine LED verbaut. Die Kommunikation zwischen dem *NFC EEPROM*, in das die über *NFC* übermittelten Nachrichten direkt gespeichert werden und dem *STM8L152C6*, findet über I^2C statt. [14]

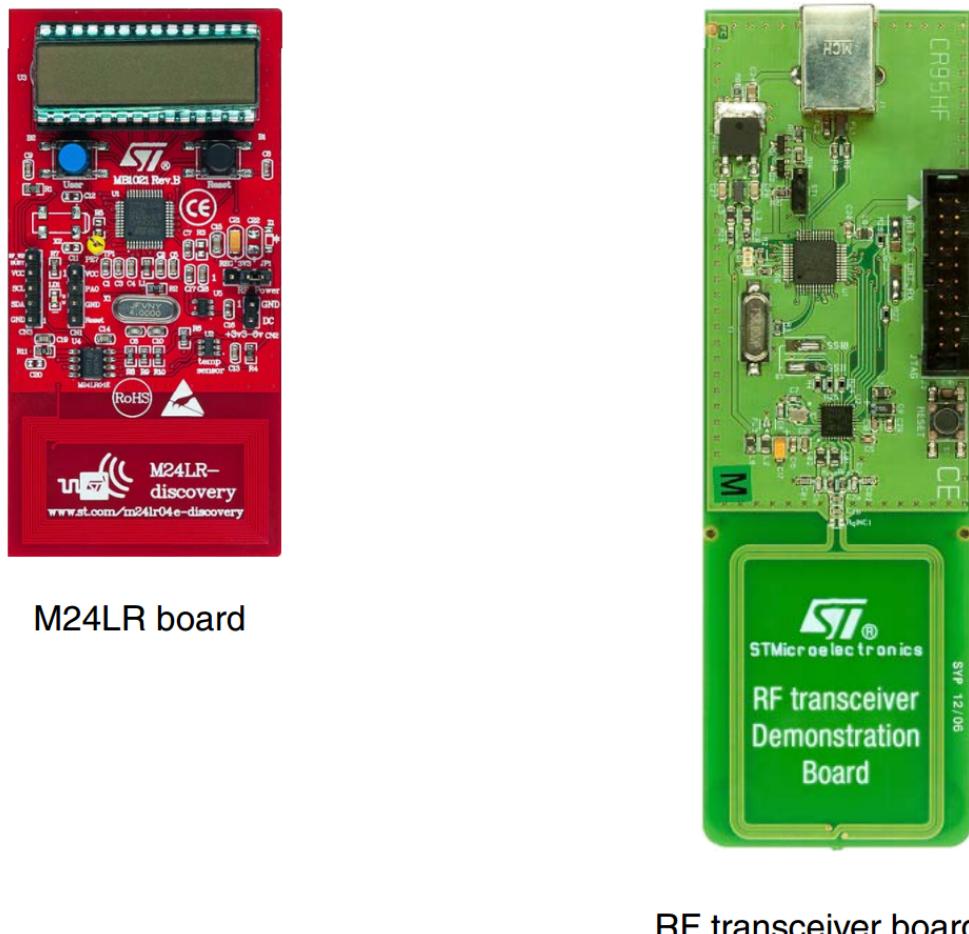


Abbildung 2.6: *M24LR-Discovery Kit*
[15]

2.4.1 Funktionsweise des M24LR Board

Wie auf der Abbildung 2.7 zu sehen ist, verhält sich das *M24LR-Discovery Board* wie ein *Tag*, es wird aus der NFC-Antenne und dem NFC-fähigen *EEPROM* gebildet. Dieses *Tag* kann beschrieben und ausgelesen werden. Für den Betrieb erhält es seine Energie über den Initiator (Smartphone/*RF Transciever*). Der Mikrocontroller wird über einen speziellen Ausgangspin des *EERPOMs* mit einer Spannung von 3,3 V versorgt. Der Mikrocontroller hat zudem über ein *I²C Interface* Lese- und Schreibzugriff auf den Speicherbereich des *NFC-EEPROMs*. Er hat jedoch keine Möglichkeit das Senden einer *NFC-Nachricht* über den *EEPROM* zu initiieren, auch wenn Energie zur Verfügung stehen würde. Er hat also bis in den Speicher des *EERPOMs* Einfluß, aber nicht darüber hinaus. Des Weiteren hat der Mikrocontroller Zugriff auf das Display und eine LED. [14]

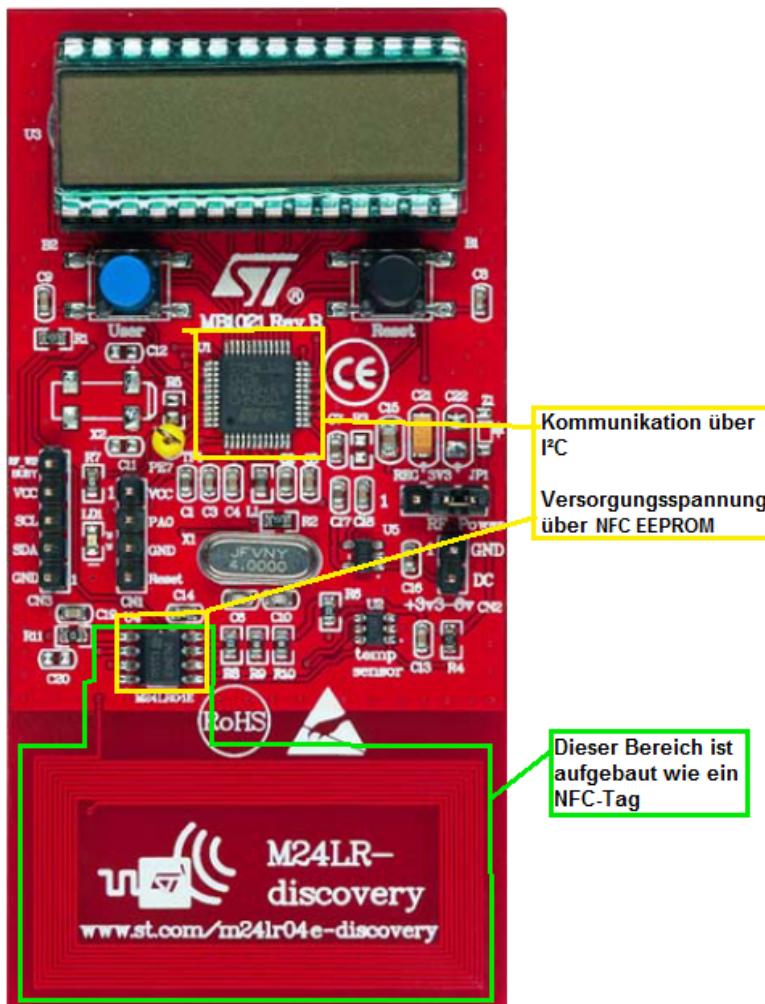


Abbildung 2.7: *M24LR-Discovery Board*
[15]

2.4.2 Beschreibung des Mikrocontrollers STM8L152C6

Der Mikrocontroller *STM8L152C6* ist auf der Harvard⁶ Architektur aufgebaut und kann mit einer maximalen Frequenz von 16 MHz getaktet werden. Zur Kommunikation stehen verschiedene Bussysteme zur Verfügung, wie *SPI*, *USART* und *I²C*. Die Besonderheiten des *STM8L152C6* sind, dass er mit einer Spannung von 1,8 V bis 3,6 V betrieben werden kann, dass der Mikrocontroller im *Low Power*-Modus nur 5,1 μ A benötigt und ein internes *EEPROM* steht zum Speichern von Variablen zur Verfügung. [16, Seite 1]

2.4.3 Beschreibung des NFC EEPROM

Das *M24LR04E* von *STMicroelectronics*, welches auf diesem Board verbaut ist, hat eine *I²C*-Schnittstelle und kann in einem Spannungsbereich von 1,8 V bis 5,5 V betrieben werden. Der Speicher hat eine Größe von 4-Kbit. Die maximale Schreigeschwindigkeit mit *I²C* liegt bei 5 ms, über *NFC* bei 5,75 ms. Das *M24LR04E* unterstützt das 10% und 100% *ASK*-Modulationsverfahren. Es hat zusätzlich noch einen analogen Ausgang *V_{out}* für das Energy Harvesting. [17, Seite 1]

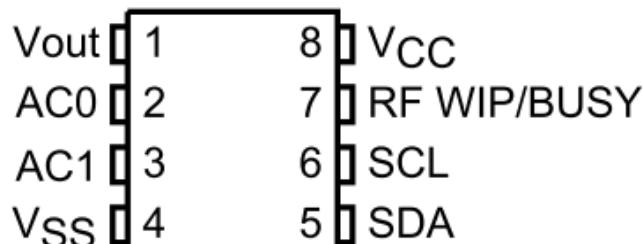


Abbildung 2.8: Ausgänge des *M24LR04E-R* (*NFC*-fähiges *EEPROM*):

V_{out} = Energy Harvesting analoger Ausgang,
AC0 und AC1 = *NFC*-Antennen Ein- und Ausgang,
 V_{SS} = Ground,
 V_{CC} = Supply Voltage,
SCL und *SDA* = *I²C*-Bus

[17, Seite 14]

⁶siehe Kapitel 2.3.1

2.5 Gleichrichterschaltungen

Gleichrichterschaltungen dienen dazu aus Wechselspannung, wie die induzierte Spannung der NFC-Antenne, eine Gleichspannung zu gewinnen. Bei Gleichrichterschaltungen wird mit Dioden gearbeitet, sie lassen bei richtiger Schaltung die positive Halbwelle der sinusförmigen Wechselspannung durch und sperren die negative Halbwelle. Aufgrund der nichtlinearen Kennlinie der Dioden treten, je nach Verhältnis von Eingangsspannung und Diodendurchlassspannung, Abweichungen der Sinusform auf. Mit der in Kapitel 2.5.1 beschriebenen Schaltung wird zusätzlich aus der negativen Halbwelle eine positive gemacht. [18, Seite 73]

2.5.1 Zweiweg Graetz-Gleichrichter mit Glättung

Die Brückenschaltung ist ein Zweiweggleichrichter. Er wird für kleinere bis mittlere Leistungen als Bauteil (z.B. B40C1500) angeboten. Die Brückenschaltung

macht zusätzlich aus der negativen Halbwelle eine positive Halbwelle. Durch

Hinzuschalten eines Kondensators, parallel zum Lastwiderstand (Verbraucher) wird die gleichgerichtete Spannung geglättet. Diese Schaltung ist aber, aufgrund der zwei stromdurchflossenen Dioden, nicht geeignet für kleinere Spannungen, da an den Dioden immer eine Durchlassspannung von $U_S \approx 0.7$ V abfällt. [18, Seite 82]

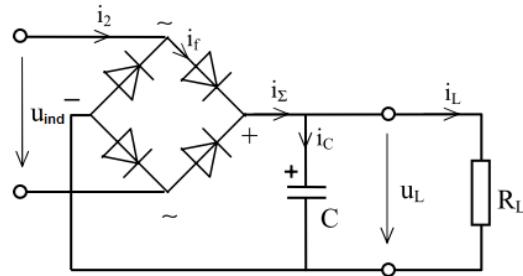


Abbildung 2.9: Aufbau eines Brückengleichrichters mit Glättung

[18]

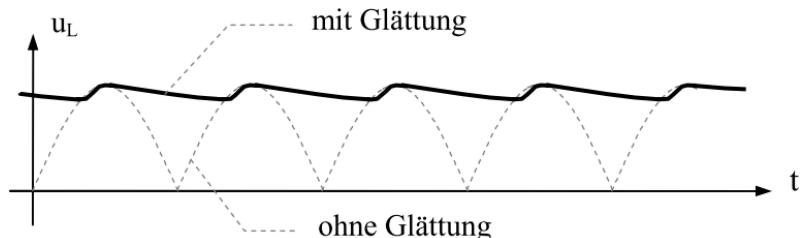


Abbildung 2.10: Spannungsverlauf des belasteten Graetz-Gleichrichters mit und ohne Glättung

[18]

2.5.2 Spannungsverdopplung mit der Delonschaltung

Beim Gleichrichten von Wechselspannung lässt sich die Amplitudenspannung verdoppeln. Dazu ist die Delonschaltung geeignet. Mit Hilfe von zwei Dioden und zwei Kondensatoren wird nach einer positiven und einer darauf folgenden negativen Halbwelle, wie in Abbildung

2.12 zu sehen, die Spannung verdoppelt. Ein Vorteil dieser Vorgehensweise ist das schnelle Erreichen der doppelten Spannung. Durch Anschließen einer Last wird der Kondensator entladen und die Ausgangsspannung wellig, dadurch kann nie der Wert $2 \cdot \hat{u}_2$ erreicht werden. [18, Seite 100 - 101]

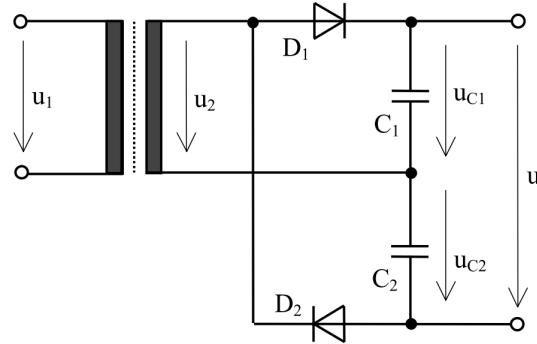


Abbildung 2.11: Aufbau einer Delonschaltung
[18]

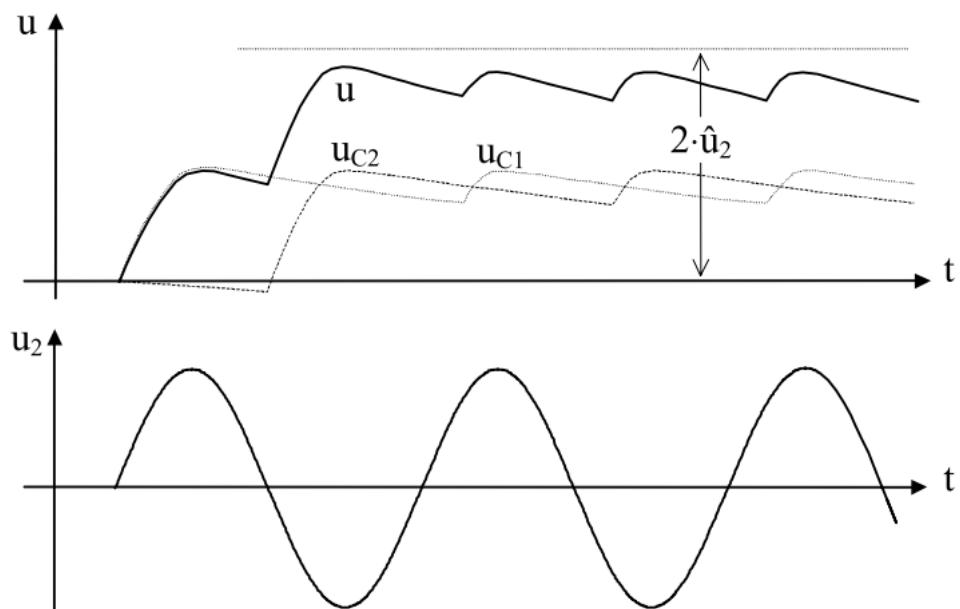


Abbildung 2.12: Spannungsverlauf einer belasteten Delonschaltung
[18]

2.5.3 Spannungsvervielfachung durch Kaskadierung der Villardschaltung

Durch die Kaskadierung der Villardschaltung (siehe Abbildung 2.13) können, je nach Kaskadierung hohe Spannungen erreicht werden. Nachteilig ist aber, dass der Spannungsanstieg um so länger dauert je höher die Ausbaustufe ist. Zudem ist der Ausgang nur gering belastbar. [18, Seite 103]

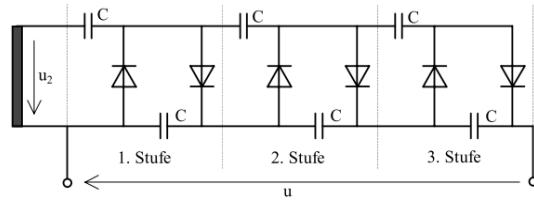


Abbildung 2.13: Kaskade einer Villardschaltung
[18]

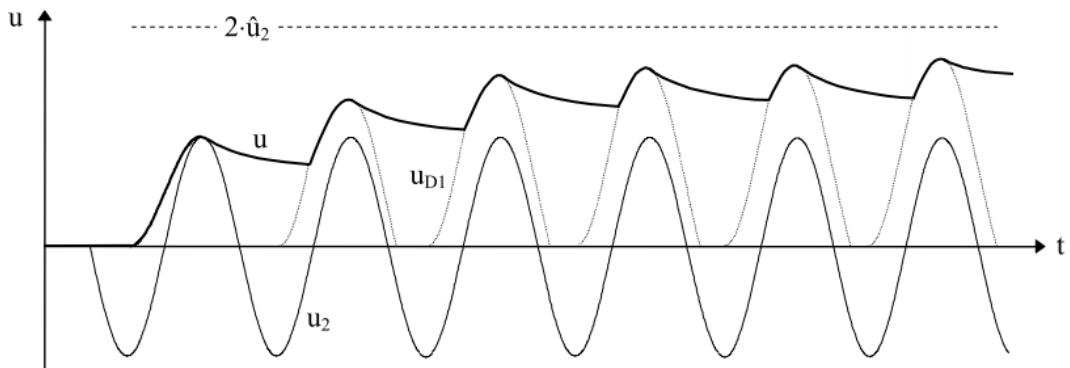


Abbildung 2.14: Spannungsverlauf einer belasteten Villardschaltung mit einer Ausbaustufe.
[18]

3 Inbetriebnahme der Hardware

Für dieses Projekt wurde das *M24LR-Discovery Board Kit* von der Firma *STMicroelectronics* gewählt. Die Besonderheit, des sich darauf befindenen Mikrocontrollers *STM8L152C6*, ist die niedrige Spannungsversorgung, die für den Betrieb benötigt wird. Diese liegt zwischen 1.8 V bis 3.6 V. [16]

3.1 Das M24LR-Discovery Board

Das *M24LR-Discovery Board* ist mit vier *Pins* (*VCC*, *PAO*, *GND*, *Reset*) für den *SWIM connector*, zwei *Pins* (*GND*, *DC*) für den Anschluss einer Gleichspannungsquelle und drei *Pins* (*REG*, *3V3*, *JPI*) mit zugehörigem Jumper zum Auswählen der Spannungsquelle, versehen. Über den *SWIM connector* (4 Pins) wird das *Debug* und *Flash Tool ST-Link/V2* angeschlossen (Hinweis: *VCC* ist der rechte *Pin* am *ST-LINK/V2*). Die externe Gleichspannungsquelle, die zum *Flashen* und *Debuggen* benötigt wird, muss eine Spannung von mindestens 3,3V bis maximal 6V bereitstellen. Für den Betrieb des *ST-LINK/V2* unter Windows 7 werden die Treiber *STSW-LINK003*¹ benötigt. Für die *STM8L152C6* Mikrocontrollerreihe stellt *STMicroelectronics* die Entwicklungsumgebung *ST Visual Develop* zur Verfügung. Diese enthält das Programm *ST Visual Develop* zum Entwickeln und Erstellen eines .s19 Files und den *ST Visual Programmer* um das .s19 File auf den *STM8L152C6* zu *flashen*. Als integrierten Compiler für das *ST Visual Develop* wird der *Cosmic CxSTM8 32K*² benötigt. Der Quellcode, der auf dem *M24LR-Discovery Board* läuft, steht auf www.st.com unter dem Namen *STSW-M24LR011* zum Herunterladen zur Verfügung. [14]

¹http://www.st.com/web/en/catalog/tools/FM146/CL1984/SC720/SS1450/PF251168?s_searchtype=partnumber

²<http://www.cosmic-software.com/>



Abbildung 3.1: *M24LR-Discovery* und *ST-LINK/V2*

3.2 RF Transceiver Demonstration Board

Das *RF Transceiver Demonstration Board* ist mit einem *JTAG*-Stecker und einem *USB 2.0 Typ B*-Stecker, versehen. Auf diesem Board ist der Mikrocontroller *STM32F103C8T6* verbaut. Zum *Flashen* und *Debuggen* wird das *ST-LINK/V2* über den *JTAG*-Stecker verbunden. Die Spannungsversorgung wird über den *USB*-Stecker gewährleistet. Um den Mikrocontroller zu *flashen* wird das *STM32 ST-LINK Utility* benötigt. Als Entwicklungsumgebung wird das *ST Visual Develop*-Programm verwendet. Es besteht die Möglichkeit, den *EEPROM* des *M24LR-Discovery Board* mit den *RF Transceiver Demonstration Board* zu beschreiben. Dazu wird das *M24LRxx Application Tool* benötigt. [14]

3.3 Android App NFC-V Reader

Es besteht ebenfalls die Möglichkeit, den *EEPROM* des *M24LR-Discovery Board* mit einem Smartphone zu beschreiben. Dazu wird ein *NFC*-fähiges Smartphone (z.B. *LG Nexus 4*, *Galaxy Nexus* oder *Samsung S4*) benötigt oder ein Smartphone mit einem Speicherkartenplatz. Dieser kann mit einer *NFC*-Karte besteckt werden. Das dazugehörige App (*NFC-V Reader*), welches Nachrichten im *NDEF*-Format verschickt, kann über den *Google Play Store* auf dem Gerät installiert werden.

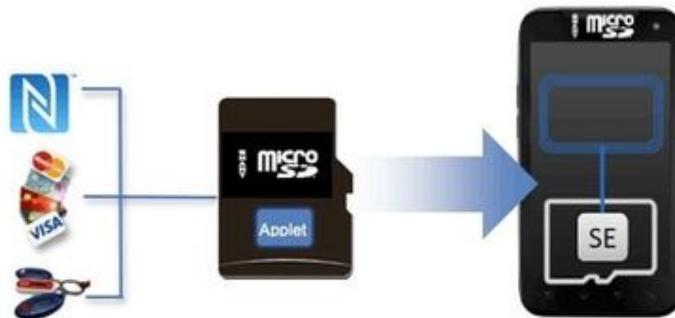


Abbildung 3.2: *NFC*-fähige *SD*-Karte
[19]

4 Entwicklung der eingebetteten Authentifizierungssoftware

In diesem Kapitel wird beschrieben, wie die Authentifizierung implementiert wird. Dabei wurde, sofern es möglich war, nach dem V-Modell vorgegangen. Zuerst wurden die Anforderungen festgelegt, dann eine Analyse des vorhandenen Systems vorgenommen, danach der Entwurf in Form eines Zustandsdiagramms und wie die wichtigsten C-Funktionen implementiert wurden erklärt. Dazu wurde jeweils ein Programmablaufplan zum besseren Verständnis hinzugefügt.

4.1 Analyse

Vorgaben:

Als Vorgaben der Authentifizierung wurden folgende Ziele gesetzt:

- Durch erstmaliges Empfangen einer Pinnummer soll diese gespeichert werden und das System in den Zustand “geschlossen” gehen. Dabei soll eine LED über einen Pin eingeschaltet werden.
- Durch erneute Eingabe einer Pinnummer, soll diese mit der vorherigen Nummer verglichen werden und bei Gleichheit das System in den Zustand “geöffnet” übergehen. Dabei ist die zuvor eingeschaltete LED wieder auszuschalten.

Beachtung von technischen und physikalischen Besonderheiten:

Dieses auf Energy Harvesting basierende System, hat zufolge, dass es nicht wie ein normaler Mikrocontroller über eine stetig verfügbare Energiequelle, wie eine Batterie, Lichtmaschine oder einen Stromanschluss am Netz, verfügt. Daher müssen ein paar Gegebenheiten für das Programmieren auf dem *STM8L152C6* beachtet werden:

- Durch das beschränkte Angebot an Energie muss beachtet werden, dass der Mikrocontroller nach Entfernen der induktiven Spannung keine Versorgungsspannung mehr hat. Daher muss der Zustand, bei Widererwachen gleich bleiben.
- Aus dem selben Grund wie eben genannt, muss die eingegebene Pinnummer im geschlossenen Zustand in einem nichtflüchtigen Speicher hinterlegt sein, um sie nach erneuter Aktivierung des Mikrocontrollers vergleichen zu können.
- Die durch die *NFC*-Antenne geschriebenen Daten in das *NFC-EEPROM* müssen, wenn das System in den Zustand ”geschlossen” oder ”geöffnet” übergeht, wieder überschrieben werden. So kann vermieden werden, dass Dritte die Pinnummer auslesen und sich so Zugang verschaffen.

4.2 Entwurf

Bevor der erste Programmcode geschrieben wird, ist es ratsam sich eine Grundstruktur zu überlegen. Diese kann am Besten in einem Zustandsdiagramm dargestellt werden. Das Zustandsdiagramm ist in diesem Falle eine Grundstruktur, welche die Übersicht für das Programmieren vereinfacht. Im Zustandsdiagramm werden flüchtige Variablen und die Variablen, deren Daten auch ohne Strom erhalten werden sollen, festgelegt. Das hier dargestellte Zustandsdiagramm zeigt den Status '*S_GEÖFFNET*' und '*S_GESCHLOSSEN*' an und zudem noch, den Status *S_ON* und *S_OFF*, welche aber nicht implementiert werden, sondern nur die Energiezufuhr darstellen sollen. Im Rahmen der Energiezufuhr sollen dann die Unterzustände '*S_GEÖFFNET*' und '*S_GESCHLOSSEN*' eingebunden sein. Während sich das System im geöffneten Zustand befindet, wird im *NFC EEPROM* auf das Flag '\$' das von der App in den Speicher geschrieben wird, überprüft. Ist dieses vorhanden, so werden die vier nachfolgenden Speicherplätze byteweise ausgelesen und in den internen *EEPROM*-Speicher geschrieben.

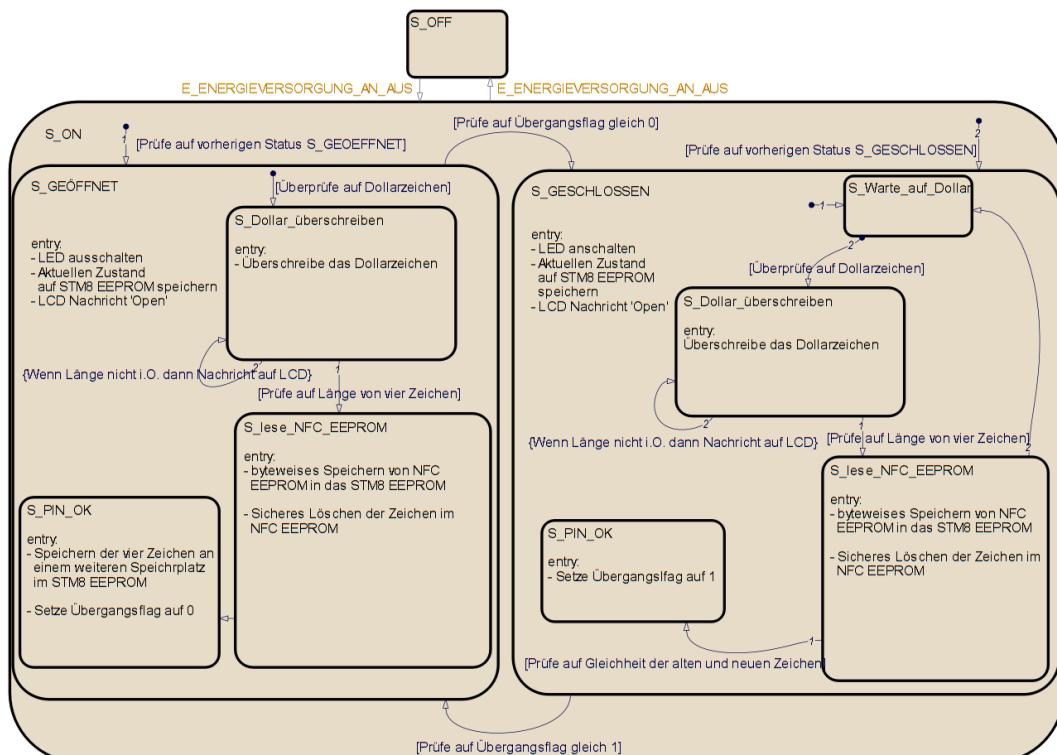


Abbildung 4.1: Vorentwicklung einer Zustandsdarstellung für das Authentifizierungssystem. Die Zustände *S_OFF* und *S_ON* werden von der Energieübertragung gesteuert

Schaubild: Matthias Singer, Erstellt mit MATLAB® StateFlow

4.3 NDEF Message Daten im NFC EEPROM

Wie im Grundlagenteil schon behandelt, werden die durch *NFC* übermittelten Daten direkt in das *EEPROM* hineingeschrieben. Dabei werden die Rohdaten mit *Header* hingingeschrieben. Um diese Daten nun über I^2C heraus lesen zu können muss die Adresse, in der die Daten liegen, bekannt sein. Die I^2C Benutzeradresse, des frei verfügbaren Speichers, ist von *STMicroelectronics* festgelegt und hat den Hexadezimalwert ‘A6’. Um nun die Nutzdaten zu bekommen muss einem der Aufbau eines *NDEF*-Records bekannt sein. Wie im Grundlagenkapitel 2.1.4 beschrieben, sind die ersten 6 Bytes, eines *NDEF*-Records, der *Header*. Da die ersten 8 Bytes eines frei verfügbaren Speichers beim *Tagtyp 4* den *Capability Container* enthalten müssen, werden diese dazugezählt [1, Seite 118-119]. Nun ist die genaue Anfangsadresse, in der die *NDEF*-Nutzdaten liegen, bekannt. Das ist hier die I^2C Adresse ‘A6’ mit einem Offset von 14. Weil die Adresse von Null an hochgezählt wird ist der Offset (13) in HEX 0x000d. An dieser Stelle liegen nun die Nutzdaten, wie sie über das App (*NFC-V Reader*) von *STMicroelectronics* übermittelt wurden.

4.4 Implementierung des byteweisen Auslesens der Nutzdaten

Copy_Data_NFC_EE_to_Int_EE()

Ziele:

- Daten über I^2C aus *M24LR04E EEPROM* auslesen und in den *STM8L152C6* internen *EEPROM* speichern
- *Flag* setzen für den Übergang vom offenen in den geschlossenen Zustand

Interne Funktionen:

```
OverwriteAll_CODE_EEPROM();  
M24LR04E_Init();  
M24LR04E_ReadOneByte();  
I2C_Cmd();  
CLK_PeripheralClockConfig()  
GPIO_HIGH();
```

Allgemeine Beschreibung:

In der C-Funktion *Copy_Data_NFC_EE_to_Int_EE()* werden die Daten, die im *NDEF*-Format in *M24LR04E EEPROM* durch die *NFC-V Reader* App geschrieben wurden, gelesen und im internen *EEPROM* des *STM8L152C6* abgespeichert. Ein *Flag* wird gesetzt und somit die Übergangsbestimmung, in den Zustand geschlossen, festgelegt.

Byteweises Auslesen:

M24LR04E_Init()

Für das byteweise Auslesen des *EEPROM* über I^2C muss zunächst mit der Funktion *M24LR04E_Init()* das *EERPOM* und der I^2C -Bus initialisiert werden. Des Weiteren wird in dieser Funktion das Clock Signal und dessen Frequenz für die *SCL*-Leitung aktiviert.

M24LR04E_ReadOneByte ()

Mit der Funktion *M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER, ReadAddr_Byt1, OneByteMemory1)* wird ein Byte von der I^2C -Adresse des *EERPOMs* mit dem Offset von 0x000d ausgelesen und mit dem Zeiger *OneByteMemory1* an den vorher deklarierten internen *EEPROM* Speicher *EE_Buffer_1* übergeben.

I^2C deaktivieren:

I2C_Cmd(M24LR04E_I2C, DISABLE)

Der *I2C_Cmd(M24LR04E_I2C, DISABLE)* wird mit den Überabeparametern *M24LR04E_I2C* und *DISABLE* der I^2C -Bus wieder deaktiviert.

GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT,M24LR04E_I2C_SCL_PIN)

Das *CLK*-Signal der *SCL* und die *SDA*-Datenleitung vom I^2C müssen auf *High* gesetzt werden.

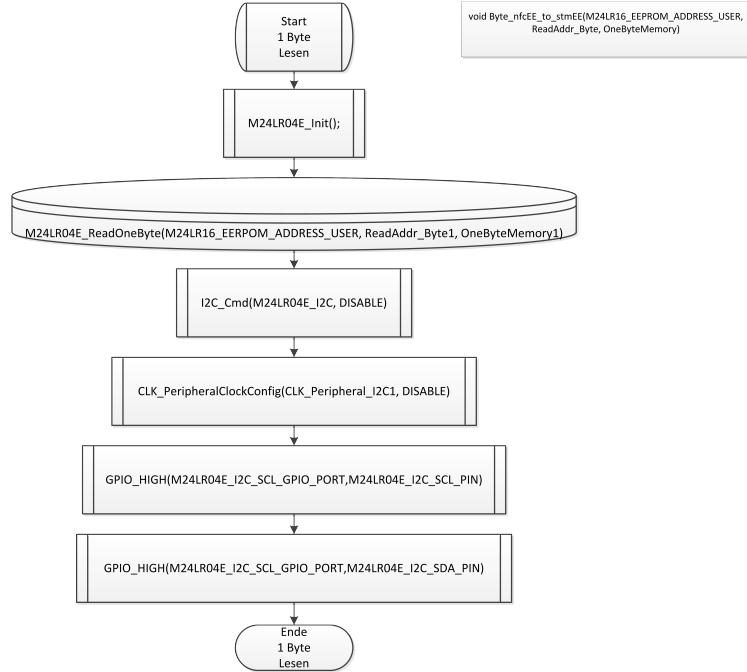


Abbildung 4.2: Programmablaufdiagramm zum Auslesen von einem Byte aus dem *EEPROM* und Kopieren in den internen *EEPROM*

Beschreibung des gesamten Programmablaufs nach Abbildung 4.3:

Das erste Zeichen, das in den Nutzdaten einer *NDEF*-Nachricht vorhanden ist, muss für diese Anwendung ein Dollarzeichen sein. Bis jetzt muss der Benutzer, in das *NFC-V Reader* App, selbst ein '\$' Zeichen einfügen. Später soll dies in einem geeigneten App von vornherein implementiert sein. Das erste ausgelesene Byte enthält ein '\$' Zeichen, ist dieses Zeichen vorhanden, so handelt es sich um eine neu eingegebene *NDEF*-Nachricht. Durch eine logische Abfrage wird dem System nun erlaubt auch die weiteren vier Zeichen im *EERPOM* auszulesen und auf dem internen *EEPROM* zu speichern. Das Vorgehen ist hier, wie in Abbildung 4.3 dargestellt, beschrieben. Kurz davor sollte noch im *EEPROM* mit der Funktion *Overwrite_dollar_EEPROM()* das Dollarzeichen im *NFC EEPROM* überschrieben werden, somit wird eine *NDEF*-Nachricht einmal vollständig gelesen und danach, solange keine neue Nachricht mit einem Dollarzeichen in das *NFC EEPROM* geschrieben wird, nicht mehr.

4 Entwicklung der eingebetteten Authentifizierungssoftware

Listing 4.1: Auszug aus `Copy_Data_NFC_EE_to_Int_EE()` Abfrage; wenn ein Dollarzeichen vorhanden ist, wird dies sofort im *NFC EEPROM* überschrieben.

```
1 if ( EE_Buffer_1[0] == '$' )
2 {
3     Overwrite_dollar_EEPROM();
4     ...
5 }
```

Nach dem Kopieren der vier Zeichen wird ein *Flag* im internen *EEPROM* gesetzt. Dieses wird benötigt um den Zustand, in dem sich das System befindet, zu speichern nachdem die Energie zwischenzeitlich unterbrochen wurde.

Zuletzt müssen nun noch die *NDEF-Message*-Nutzdaten aus dem *M24LR04E EEPROM* entfernt werden. Dies geschieht durch einfaches Überschreiben mit ASCII-Leerzeichen in der Funktion `OverwriteAll_CODE_EEPROM()`. Das gesamte Programmablaufdiagramm der Funktion `Copy_Data_NFC_EE_to_Int_EE()` ist dann wie in Abbildung 4.3 dargestellt.

4 Entwicklung der eingebetteten Authentifizierungssoftware

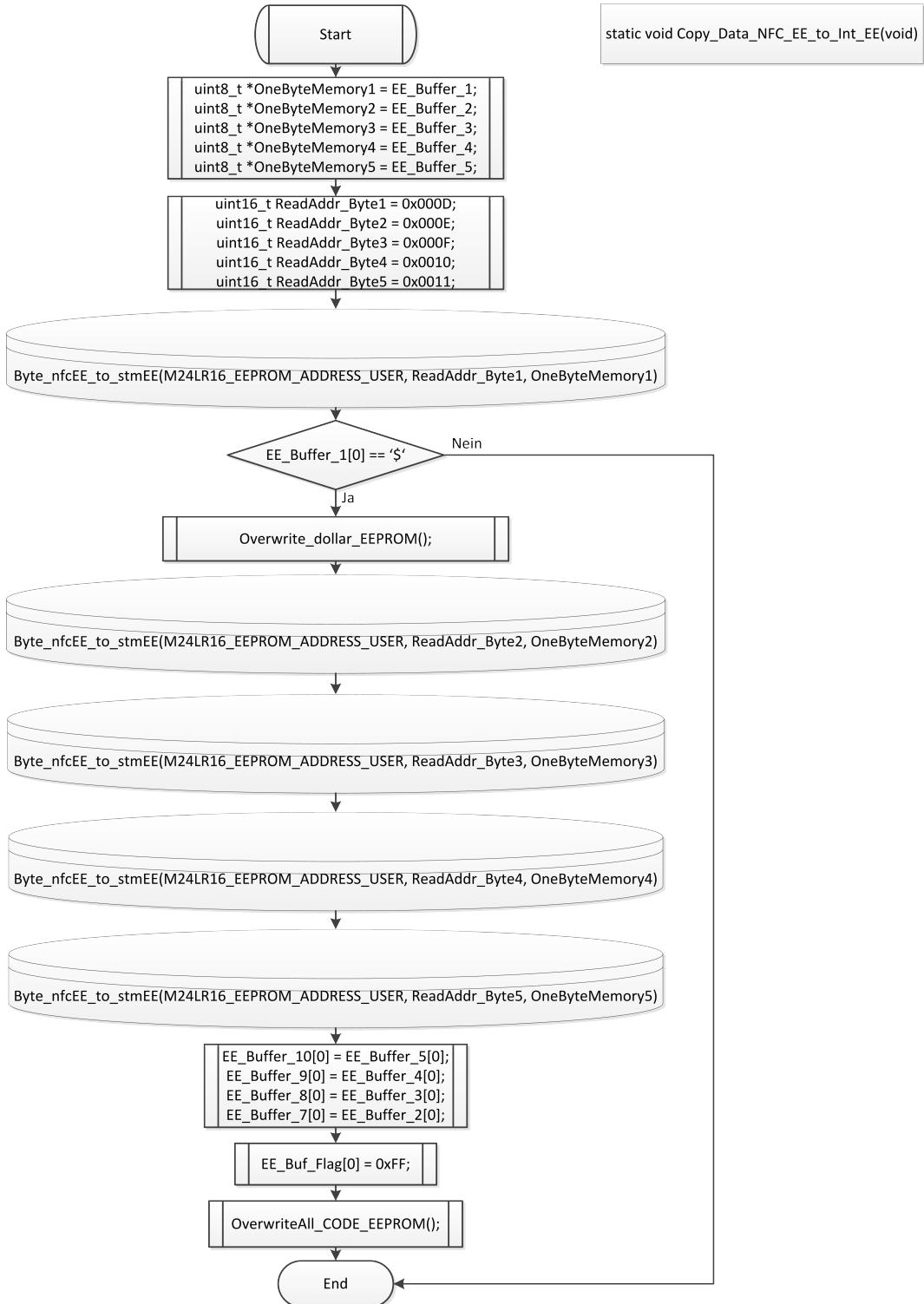


Abbildung 4.3: Programmablaufdiagramm der Funktion

`Copy_Data_NFC_EE_to_Int_EE()` mit Aufruf der Funktion
`Byte_nfcEE_to_stmEE(M24LR16_EEPROM_ADDRESS_USER,
ReadAddr_Byt,OneByteMemory)`

4.5 Im Zustand geschlossen einen Pin erhalten und vergleichen

Copy_Data_NFC_EE_to_Int_EE_and_comp()

Ziele:

- Daten über I^2C aus *M24LR04E EEPROM* auslesen und in den *STM8L152C6* internen *EEPROM* speichern
- vergleichen mit den vorherigen gespeicherten vier Zeichen
- Transaktionsbedingung von geschlossenen in geöffneten Zustand setzen

Interne Funktionen:

*OverwriteAll_CODE_EEPROM();
Overwrite_dollar_EEPROM(); Check_PID_Buffer_Equal()
M24LR04E_Init(); M24LR04E_ReadOneByte();
I2C_Cmd(); CLK_PeripheralClockConfig();
GPIO_HIGH();*

Allgemeine Beschreibung:

Die Funktion *Copy_Data_NFC_EE_to_Int_EE_and_comp()* wird nur dann aufgerufen, wenn sich das System in dem Zustand geschlossen befindet. Sie liest den ersten Speicherplatz der Nutzdaten aus und prüft ihn auf das Dollarzeichen. Zusätzlich schreibt sie die vier Zeichen in den internen *EEPROM* wie bei der Funktion *Copy_Data_NFC_EE_to_Int_EE()*, beschrieben in Kapitel 4.4. Danach werden die vier Zeichen mit den vorherigen verglichen und bei Gleichheit ein *Flag* gesetzt, das als Zustandsübergangsbedingung abgefragt wird. Auch hier wird am Ende der *NFC EEPROM* überschrieben, da die persönliche Nummer sonst über ein *NFC-fähiges Gerät* auslesbar ist. So könnten Dritte, die Gewohnheit oder Nachlässigkeit von Nutzern, welche immer den selben Pincode verwenden, ausnutzen.

static uint8_t Check_PID_Buffer_Equal()

Diese Funktion vergleicht die vier neu eingelesenen Zeichen, mit den Zeichen, die zu dem Zustand geschlossen geführt haben. Sind alle vier Zeichen identisch und von der Reihenfolge her richtig, so gibt die Funktion einen Boolean Wert ‘Success’ zurück, ansonsten einen ‘Error’.

Diese Funktion ist in eine *if*-Schleife eingebaut und der Inhalt wird ausgeführt, sobald die Bedingung wahr ist. Bei positiver Rückmeldung ‘Success’, wird das interne *Flag* auf 0x00 gesetzt, damit ist die Transaktionsbedingung für den Übergang in den Zustand ‘geöffnet’ freigegeben.

4.6 Implementierung der Zustände ‘S_GEÖFFNET’ und ‘S_GESCHLOSSEN’

Wie im Kapitel 4.4 und 4.5 beschrieben, wird jeweils ein *Flag* gesetzt, das die Bedingung für die Transaktion von einem in den anderen Zustand ist.

Die Abfrage auf das nicht flüchtig gespeicherte *Flag* wird in einer *if*-Schleife realisiert. Das heißt wenn das System aus dem S_OFF Zustand in Betrieb geht, wird zuerst das *Flag* abgefragt und das System wird dann sofort in den gleichen Zustand, wie das System vor dem Verlust der Energie gewesen ist, geführt.

Das System geht in den Zustand ‘S_GEÖFFNET’, wenn das *Flag* *EE_Buf_Flag[0]* den Wert 0x00 hat. Hat das *Flag* den Wert 0xFF so geht es in den Zustand ‘S_GESCHLOSSEN’.

Im Zustand ‘S_GESCHLOSSEN’ wird mit der Funktion *GPIO_SetBits(LED_GPIO_PORT,GPIO_Pin_6)* die LED eingeschaltet und mit der Funktion *State_DisplayMessage ("CLOSED", 6)* der String CLOSED in den RAM der LCD-Anzeige geladen. Durch Aufruf der Funktion *Copy_Data_NFC_EE_to_Int_EE_and_comp()* wird die Übergangsbedingung geprüft.

Im Zustand ‘S_GEÖFFNET’ wird mit der Funktion *GPIO_ResetBits(LED_GPIO_PORT,GPIO_Pin_6)*, die LED ausgeschaltet und mit der Funktion *State_DisplayMessage ("OPEN", 4)* der String ‘OPEN’ in den RAM der LCD-Anzeige zum Anzeigen geladen. Zum Schluss wird durch Aufrufen der Funktion *Copy_Data_NFC_EE_to_Int_EE()* die Übergangsbedingung geprüft, und das Übergangsflag gesetzt (siehe Kapitel 4.5).

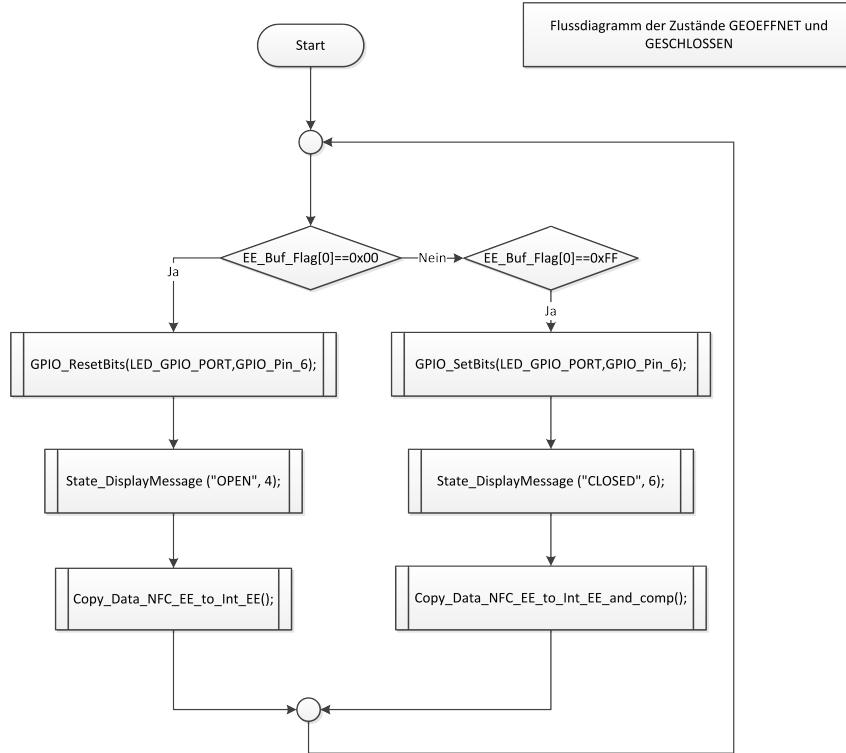


Abbildung 4.4: Zustand GEOEFFNET und GESCHLOSSEN als Flussdiagramm dargestellt

4.7 Die Funktion Text in das Display schreiben

```
static void State_DisplayMessage(uint8_t message[], uint8_t PayloadLength)
```

Ziele:

- einen String im Display des *M24LR-Discovery Board* anzeigen

Interne Funktionen:

```
CLK_SYSCLKDivConfig();
CLK_SYSCLKSourceConfig();
CLK_SYSCLKSourceSwitchCmd();
CLK_HSICmd();
LCD_GLASS_DisplayString_I();
```

Allgemeine Beschreibung:

Die Funktion *State_DisplayMessage(uint8_t message[],uint8_t PayloadLength)* bekommt als Übergabeparameter einen String und die Länge des Strings. Interne Funktionen, die von *STMicroelectronics* schon implementiert waren, steuern das Taktsignal und laden den String in den RAM des LCD- Displays.

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

In diesem Kapitel werden die verschiedenen Möglichkeiten bewertet, durch Energy Harvesting, Energie für ein elektronisches Schloss zu gewinnen. Dabei wird die Möglichkeit, die Energie aus der induzierten Spannung anhand von verschiedenen Schaltungen, näher untersucht. Außerdem werden weitere Möglichkeiten vorgestellt, mit denen Energie aus der Umgebung "geerntet" werden kann.

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

5.1 Energy Harvesting

Das Gewinnen kleiner Mengen elektrischer Energie aus der Umgebung wird Energy Harvesting genannt. Die Energie kann aus verschiedenen Energiequellen gewonnen werden. So kann aus Vibration und Stoss durch elektrostatischen Effekt, Piezoeffekt oder Induktion, die Energie mittels Kondensator, Kristall oder Mikrogenerator umgewandelt werden. In der Praxis könnte dies dann bei Schaltern, Motoren und Brücken, die durch Wind oder Verkehrsbelastung vibrieren, eingesetzt werden. Auch aus Licht, kann mit Photodioden Energie erzeugt werden. Induktive Systeme können mit dem Effekt des magnetischen Flusses auch Energie bereitstellen. [20]

5.2 Bewertung geeigneter physikalischer Effekte für das Energy Harvesting für den Betrieb eines Schlosses

Für einen energieautarken Mikrocontroller und der Bereitstellung von Energie für das elektrische Schloss eines Schließfaches, wird möglicherweise eine Kombination aus verschiedenen Energien notwendig sein. Diese muss dann an den jeweiligen Standort angepasst sein. So wird, für das Öffnen eines Schließfaches, in einem überdachten bzw. unterirdischen Bahnhof eine Photozelle nur sehr wenig Energie aus dem schmalen Spektrum, von den dort meist eingesetzten Leuchtstoffröhren gewinnen können. Ist das Schließfach allerdings nahe an den Schienen angebracht, so kann die durch ein- und ausfahrenden Züge verursachte Vibration zur Energieversorgung genutzt werden. Die Versorgung des Mikrocontrollers, der für die Authentifizierung zuständig ist, findet über die induzierte Spannung der NFC-Antenne statt.

5.3 Versuch: Graetz Zweiweggleichrichter

Zur Bereitstellung von Energie, für z.B. einen Piezomotor, wurde ein Zweiweggleichrichter zusammengelötet und eine selbstgemachte Spule in einem festgelegten Abstand von 0,5 cm Entfernung an den *RF Transceiver Board* angebracht. Die Zweiweggleichrichterschaltung ist aufgebaut wie im Grundlagenteil Kapitel 2.5.1 beschrieben. Eingesetzt wurde ein 10Ω Lastwiederstand und ein Kondensator mit der Kapazität von $C = 220 \mu\text{F}$. In dieser Konstellation fallen am Lastwiederstand

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

27 mV ab und daraus lässt sich ein Strom von 2,7 mA berechnen.

$$I_L = \frac{U}{R_L} = \frac{27 \text{ mV}}{10 \Omega} = 2,7 \text{ mA} \quad (5.1)$$

Die Messwerte für einen offenen Kreis, ohne Lastwiderstand R_L , liegen bei 14,5 V. Dabei fließt kein Strom. Mit der Punktsteigungsformel wird die idealisierte Kennlinie des Stromverlaufes bestimmt.

$$m = \frac{I_2 - I_1}{U_2 - U_1} = \frac{0 \text{ A} - 2,7 \text{ E-3 A}}{14,5 \text{ V} - 27 \text{ E-3 V}} = -187 \text{ E-3 } \frac{\text{A}}{\text{V}} \quad (5.2)$$

$$I(U) = m \cdot (U - U_1) + I_1 \quad (5.3)$$

$$I(U) = -187 \text{ E-3 } \frac{\text{A}}{\text{V}} \cdot U + 2,705 \text{ E-3 A} \quad (5.4)$$

Aus diesen Ergebnissen lässt sich nun eine maximale Leistung von 9,8 mW ableiten (siehe Abbildung 5.1). Die maximale Leistung wird aus der Zeichnung ermittelt. Der größte Wert für die Leistung ist dort, wo das Viereck unter der Geraden die größte Fläche aufspannt.

$$P_{max} = U \cdot I \quad (5.5)$$

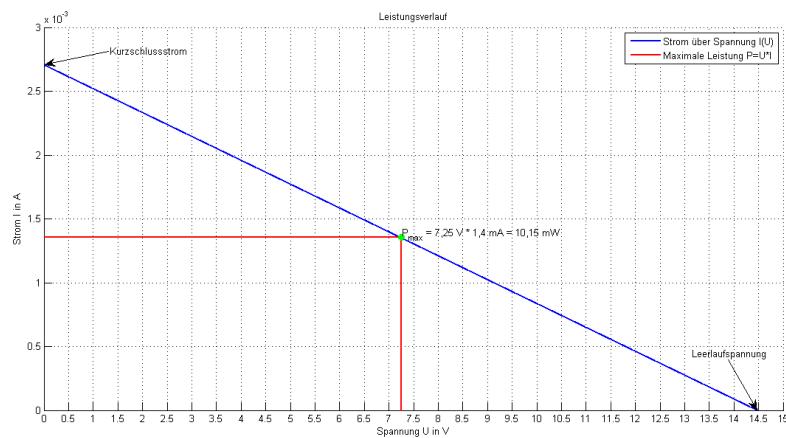


Abbildung 5.1: Verlauf der Spannung für einen Verbraucher, wie z.B. eines Piezokristallmotors mit Darstellung der maximal verfügbaren Leistung mit einer Zweiweggleichrichterschaltung
Erstellt mit MATLAB®

5.4 Versuch: Spannungsverdopplung durch Delonschaltung

Im Rahmen des Energy Harvesting und dieser Bachelorarbeit wurde die Delonschaltung mit den gleichen Bauelementen dimensioniert, wie der in Kapitel 5.3 beschriebene Zweiweggleichrichter. Einem 10Ω Lastwiderstand und zwei Kondensatoren von $22 \mu\text{F}$. Gemessen wurde unter den gleichen Bedingungen, wie in Kapitel 5.3, einmal mit Lastwiderstand und einmal in einen offenen Kreis. Dabei war die maximale Spannung (Open Loop) 27 V und die Spannung unter Last bei 20 mV.

$$I_L = \frac{U}{R_L} = \frac{20 \text{ mV}}{10 \Omega} = 2 \text{ mA} \quad (5.6)$$

$$m = \frac{I_2 - I_1}{U_2 - U_1} = \frac{0 \text{ A} - 2\text{E-}3 \text{ A}}{26 \text{ V} - 20\text{E-}3 \text{ V}} = -77\text{E-}6 \frac{\text{A}}{\text{V}} \quad (5.7)$$

$$I(U) = m \cdot (U - U_1) + I_1 \rightarrow I(U) = -77\text{E-}6 \frac{\text{A}}{\text{V}} \cdot U + 2,002\text{E-}3 \text{ A} \quad (5.8)$$

Aus diesen Ergebnissen lässt sich nun eine maximale Leistung $P_{max} = 13 \text{ mW}$ ableiten (siehe Abbildung 5.2).

$$P_{max} = U \cdot I \quad (5.9)$$

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

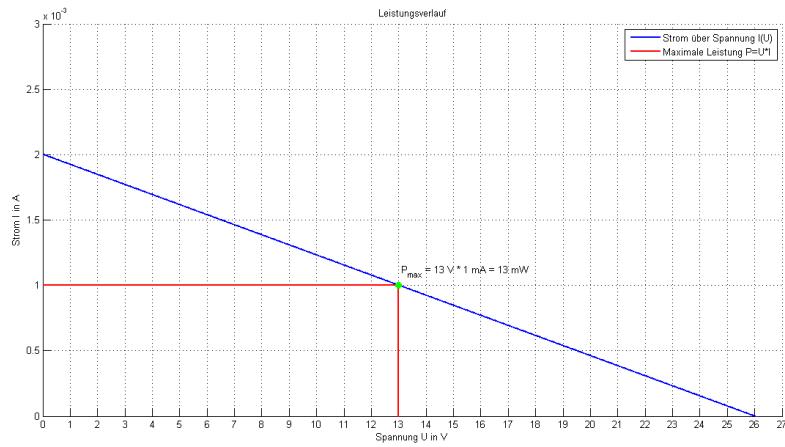


Abbildung 5.2: Verlauf der Spannung für einen Verbraucher, wie z.B. eines Piezokristallmotors mit Darstellung der maximal verfügbaren Leistung der Delonschaltung
Erstellt mit MATLAB®

5.5 Spannungsvervielfachung mit Schwingkreis und Delonschaltung

Durch einen Schwingkreis kann die Spannung, bei der Resonanzfrequenz, um ein Vielfaches überhöht werden. Wie sich das auf die Leistungsausbeute auswirkt, wird im Kapitel 5.6 erklärt.

5.5.1 Dimensionierung eines Schwingkreises

Verwendet wird eine selbstgewickelte rechteckige Luftspule. Gegeben sind folgende Daten:

- Wicklungszahl: $N = 5$
- magnetische Feldkonstante $\mu_0 = 4\pi \cdot 10^{-7} \frac{N}{A^2}$
- Außenmaße: $d_{out} = 50,2 \text{ mm}$
- Innenmaße: $d_{in} = 50 \text{ mm}$
- Faktoren für eine Rechteckspule: $K_1 = 2,34$ und $K_2 = 2,75$
- Resonanzfrequenz: $f_r = 13,56 \text{ MHz}$

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

Die Größe des Kondensators für den Schwingkreis muss bestimmt werden. Diese ist abhängig von der Frequenz f_r und der Induktivität L der Spule.

$$d = \frac{d_{out} + d_{in}}{2} = \frac{50,2 \text{ mm} + 50 \text{ mm}}{2} = 50,1 \text{ mm} \quad (5.10)$$

$$p = \frac{d_{out} - d_{in}}{d_{out} + d_{in}} = \frac{50,2 \text{ mm} - 50 \text{ mm}}{50,2 \text{ mm} + 50 \text{ mm}} = 1,996\text{E-}3 \quad (5.11)$$

$$L_{ant} = L_2 = K_1 \cdot \mu_0 \cdot N^2 \cdot \frac{d}{1 + K_2 \cdot p} = 2,34 \cdot 4 \cdot \pi \cdot 10^{-7} \frac{N}{A^2} \cdot 5^2 \cdot \frac{0,501 \text{ m}}{1 + 2,75 \cdot 1,996\text{E-}3} = 3,663 \text{ mH} \quad (5.12)$$

Mit:

$$f_r = \frac{1}{(2 \cdot \pi \cdot \sqrt{L_2 \cdot C_2})} \quad (5.13)$$

nach C_2 aufgelöst,

$$C_2 = \frac{1}{(4 \cdot f_r^2 \cdot \pi^2 \cdot L_2)} = 3,76\text{E-}12 \text{ F} = 3,76 \text{ pF} \quad (5.14)$$

ist nun die Kapazität des Kondensators bestimmt.

Bestimmung der genauen Induktivität durch Messen:

Aufgrund erster Versuche, die nicht die gewünschten Ergebnisse brachten, liegt die Vermutung nahe, dass die Formel zur Berechnung der Induktivität nur für Antennen gedacht ist, die auf eine Platine geätzt wurden. Dabei liegen die Leiterbahnen stets nebeneinander. Bei der selbst gewickelten Spule liegen diese sehr eng zueinander und teilweise auch übereinander. Zur genauen Bestimmung der Kapazität wird die Induktivität anhand eines Schwingkreises, mit bekanntem C und einer bekannten Anregungsfrequenz, durch einen Frequenzgenerator, bestimmt.

Zum Testen wird folgende Schaltung (Abbildung 5.3) verwendet:

Der Schwingkreis (Abbildung 5.3), hat hier einen Kondensator C_5 mit einer Kapazität von 100 pF und ist parallel zur Spule, deren Induktivität unbekannt ist angeschlossen. Der Schwingkreis wird mit einem Frequenzgenerator angeregt. Die

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

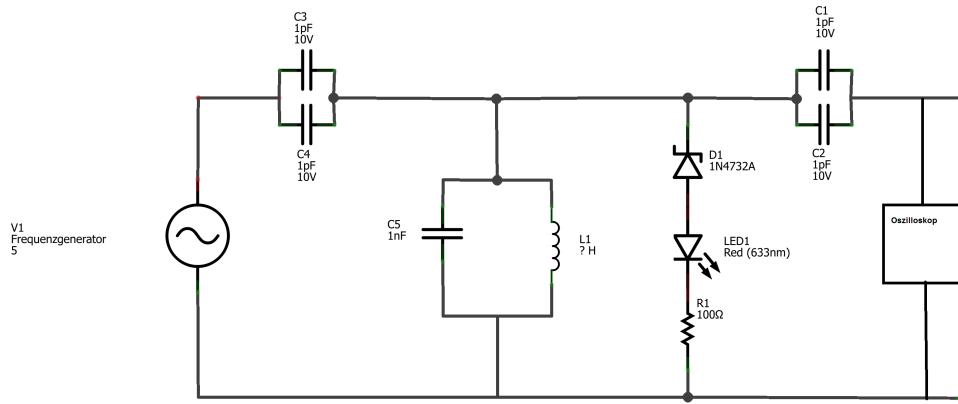


Abbildung 5.3: Testschaltung zur Ermittlung des Schwingkreises

zwei zusätzlichen Kondensatoren, mit $C_{1-4} = 1 \text{ pF}$ ¹, dienen zur galvanischen Entkoppelung. Der Oszillator wird wie in Bild 5.3 an den Schwingkreis angeschlossen und zusätzlich wird die Ausgangsfrequenz des Frequenzgenerators beobachtet. Nun wird die Frequenz erhöht. Sobald die Frequenz des Schwingkreises seine höchste Amplitude hat, ist die Resonanzfrequenz erreicht. Die Messwerte ergaben in dieser Konstellation eine Resonanzfrequenz von $f_{r1} = 5,1 \text{ MHz}$.

Die LED leuchtet ab einer Spannung von 10 Volt, wenn die Zener-Diode Spannung durchlässt. Dies wurde so gewählt, da es bei Schwingkreisen durchaus zu hohen Spannungen kommen kann. Die LED gilt als Warnhinweis, um vor hohen Spannungen zu warnen und um damit einen elektrischen Schlag vermeiden zu können.

Nun kann die Induktivität der Spule ausgerechnet werden.

Gegeben:

- $f_{r1} = 5,1 \text{ MHz}$
- $C_{Schw} = 100 \text{ pF}$

$$f_{r1} = \frac{1}{(2 \cdot \pi \cdot \sqrt{L_1 \cdot C_5})} \quad (5.15)$$

L_1 bestimmt die Induktivität.

$$L_1 = \frac{1}{(4 \cdot f_r^2 \cdot \pi^2 \cdot C_5)} = 9,74 \text{E}-6 \text{ H} = 9,74 \mu\text{H} \quad (5.16)$$

¹Da die Kondensatoren in 2 pF nicht darstellbar sind, wurden hier jeweils zwei parallel geschalten. Dadurch verdoppelt sich die Kapazität

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

Nun steht die Induktivität der Spule fest. Jetzt kann der Kondensator für den Schwingkreis auf die NFC-Frequenz von 13,56 MHz angepasst werden.

Gegeben:

- $f_{r,NFC} = 13,56 \text{ MHz}$
- $L_1 = 9,74 \mu\text{H}$

$$C_2 = \frac{1}{(4 \cdot f_{r,NFC}^2 \cdot \pi^2 \cdot L_2)} \quad (5.17)$$

$$C_2 = \frac{1}{(4 \cdot 13,56^2 \text{ MHz} \cdot \pi^2 \cdot 9,74 \mu\text{H})} = 14,1 \text{ pF} \quad (5.18)$$

Nach dem die Kapazität für C für den Schwingkreis in Abhängigkeit der Frequenz $f_r = 13,56 \text{ MHz}$ und der vorher bestimmten Induktivität festgelegt ist, kann der Schaltkreis aufgebaut werden (Siehe Abbildung 5.4). Aufgrund eines nicht vorhandenen Bauteils mit der Kapazität von 14,1 pF wurde eine Reihenschaltung von zwei Kondensatoren mit $C_1 = 15 \text{ pF}$ und $C_2 = 220 \text{ pF}$ gewählt.

$$C_{1+2} = C_{ges} = \frac{C_1 \cdot C_2}{C_1 + C_2} = \frac{15 \text{ pF} \cdot 220 \text{ pF}}{15 \text{ pF} + 220 \text{ pF}} = 14,1 \text{ pF} \quad (5.19)$$

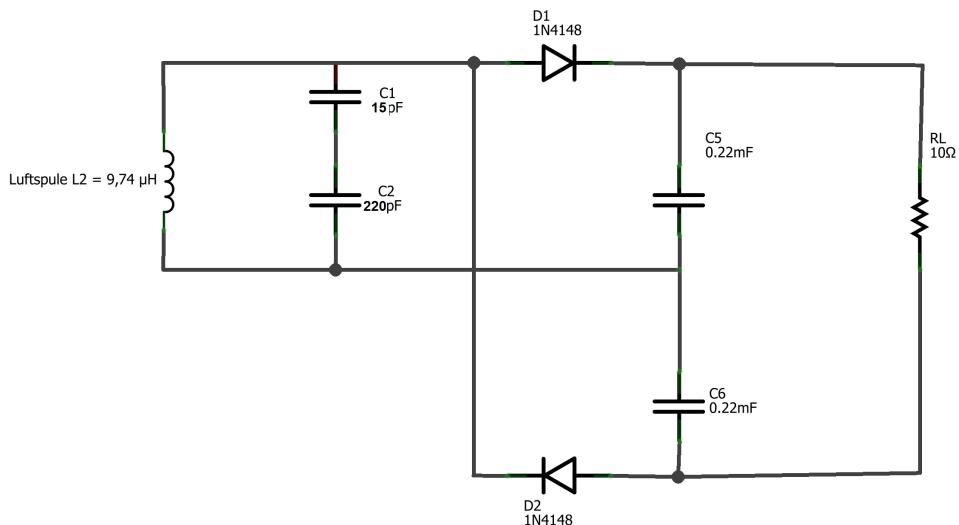


Abbildung 5.4: Zu sehen ist hier ein paralleler Schwingkreis mit anschließender Gleichrichtung durch eine Delonschaltung

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

5.6 Versuch: Schwingkreis mit Delonschaltung

Messwerte:

- 14,4 mV bei $R = 10 \Omega$
- 33,28 V bei $R \rightarrow \infty$

Mit dem Ohm'schen Gesetz wird der Strom berechnet, der am Widerstand abfällt.

$$I_{R_L} = \frac{14,4 \text{ mV}}{10 \Omega} = 1,44 \text{ mA} \quad (5.20)$$

Zudem kann durch das Zeichnen einer Kennlinie die maximal zur Verfügung stehende Leistung berechnet werden.

$$m = \frac{I_2 - I_1}{U_2 - U_1} = \frac{0 \text{ A} - 1,44 \text{ E-3 A}}{33,28 \text{ V} - 14,4 \text{ E-3 V}} = -79 \text{ E-6} \frac{\text{A}}{\text{V}} \quad (5.21)$$

$$I(U) = m \cdot (U - U_1) + I_1 \rightarrow I(U) = -79 \text{ E-6} \frac{\text{A}}{\text{V}} \cdot U + 2,598 \text{ E-3 A} \quad (5.22)$$

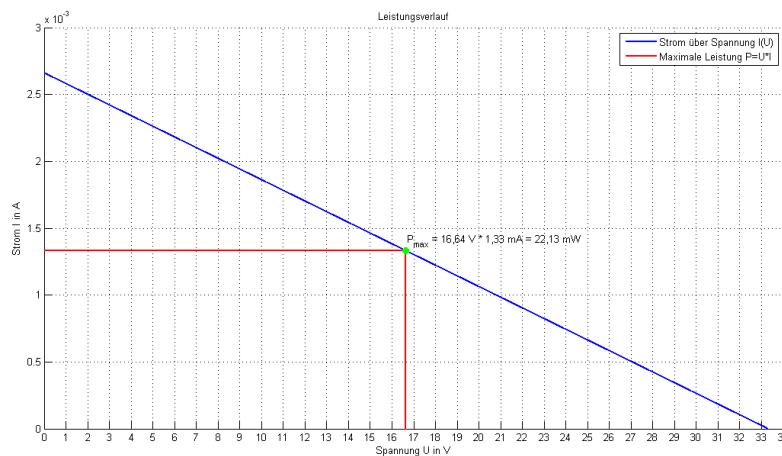


Abbildung 5.5: Verlauf der Spannung für einen Verbraucher, wie z.B. eines Piezokristallmotors mit Darstellung der maximal verfügbaren Leistung des Schwingkreises mit anschließender Gleichrichtung durch Delon-schaltung
Erstellt mit MATLAB®

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

Aus den Werten, die der Abbildung 5.5 zu entnehmen sind, ist nun die maximale Leistung von $P_{max} = 22,13 \text{ mW}$ zu berechnen. Dies ist die Leistung die einem Elektromotor, einem Piezomotor, einem Piezostapeltranslator oder einer anderen Technick, zur Verfügung stehen kann. Dabei muss bei einem Elektromotor beachtet werden, dass die angegebene Leistung nicht der maximalen Leistungsaufnahme beim Anlaufen entspricht. Günstiger ist daher ein Piezomotor.

5.6.1 Anwendung zweier Spulen

Die Idee hinter der Untersuchung, die Leistungsausbeute aus einem Gleichrichter und einer Gleichrichterschaltung mit Spannungsverdopplung, ist die, zum einen Strom für das *M24LR-Discovery Board* durch eine Spule bereitzustellen und zum anderen mit einer Spule direkt danach Energie für ein Schließsystem bereitzustellen. Werden nun die zwei Spulen, einmal vom *M24LR-Discovery Board* und einmal mit dem Gleichrichter parallel geschaltet, so verringert sich jeweils die Leistung, da jede Spule durch Gegeninduktivität das vorhandene Magnetfeld abschwächt. Dadurch bekommt keiner der beiden Verbraucher genügend Energie um funktionfähig zu bleiben.

5.7 Weitere Möglichkeiten des Energy Harvesting in Abhängigkeit von der Umgebung

Aufgrund der Gegenkopplung, die eine Spule in einem sich ändernden magnetischen Feld ausübt, kann die Energie für das Schloss nicht aus der Induktion bereitgestellt werden. Dazu werden hier weitere Möglichkeiten des Energy Harvesting aufgezeigt, die für eine energieautarke Versorgung in Frage kommen.

5.7.1 Funktionsprinzip eines Piezokristalls

Durch Einwirkung von Kraft und der daraus resultierenden Verformung von Materie entsteht bei bestimmten Materialien elektrische Spannung. Durch die Verformung

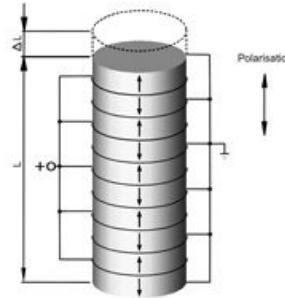


Abbildung 5.6: Aufbau eines Piezostapeltranslators
[1, Seite 121]

werden im Innern einer Elementarzelle, die Position von positiven und negativen Elektronen zueinander verschoben. Durch elektrische Polarisation entsteht dadurch an der Oberfläche des Materials Spannung. Dieses Prinzip funktioniert auch genau entgegengesetzt. So können die Kristalle mit Anlegen einer Spannung die Elektronen in eine Lage so zueinander zwingen, dass sich das Material verformt. Das Prinzip ähnelt dem eines Elektromotors. Dieser kann elektrische Energie in Bewegungsenergie umwandeln oder kinetische Energie in elektrische Energie. Der Effekt könnte zum Verriegeln eines Schlosses genutzt werden. [21, Seite 3 - 4]

Vorteile des Piezomotors gegenüber eines konventionellen Elektromotors:

Ein Vorteil der Piezokristalle ist, dass nur durch das Anlegen einer Spannung eine Bewegung verursacht wird. Dazu ist ein Piezoantrieb sehr energiesparend, da er im Haltezustand keine Energie verbraucht.

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

Weitere Vorteile sind:

- kein Getriebe
- hohe Genauigkeit
- präzise Auflösung
- reaktionsschnell

5.7.2 Einsatz von Piezokristallen als Energielieferant

Wie in Kapitel 5.7.1 beschrieben, kann ein Piezokristall nicht nur als Aktor, sondern auch als Spannungsquelle genutzt werden. Die Kraft, die dazu nötig ist muss als Druckkraft auf die Kristalle einwirken. Es gibt mehrere Möglichkeiten, je nach Standort, diese Kraft zu nutzen:

Vibrationen:

- **Standort: Zugschienen**

In der Nähe von Gleisen werden durch Zugverkehr, insbesondere Güterverkehr, starke Vibrationen verursacht. So können in der Nähe von Schienen Schwingungen in vertikaler Richtung von $1,686 \frac{\text{mm}}{\text{s}}$ auftreten[22]. Diese Schwingungen sind selbst für Menschen deutlich wahrnehmbar. Durch Einarbeiten der Piezokristalle in den Boden oder die Füße eines Gegenstandes könnte Energie aus den Vibrationen gewonnen werden.

- **Standort: hochfrequentierte Straße und Brücken**

An stark befahrenen Straßen werden durch Individualverkehr, insbesondere Lastkraftwagen, Vibrationen verursacht. Hier können Schwingungen in vertikaler Richtung von $0,493 \frac{\text{mm}}{\text{s}}$ auftreten[22].

Druck:

- **Standortunabhängig:**

Vor dem Schließfach Bodenplatten mit Piezokristallen anbringen. Der Benutzer des Schließfaches, erzeugt die Energie durch sein Gewicht.

5 Untersuchung zur Bereitstellung von zusätzlicher Energie für ein energieautarkes Schließsystem

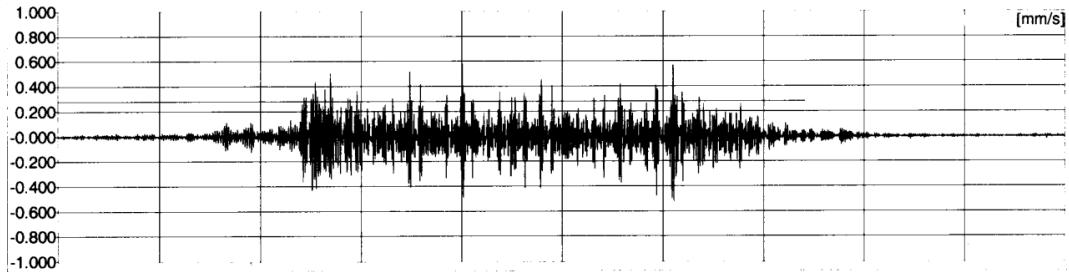


Abbildung 5.7: Hier zu sehen ist die Amplitude in mm, die ein Zug über die Zeit bei der Durchfahrt in einem Wohnhaus bei Boppard, das direkt neben den Gleisen steht, verursacht.

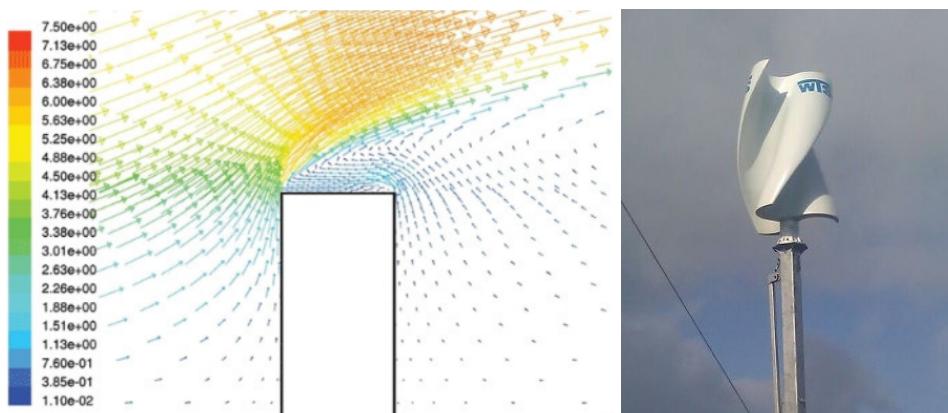
[22]

5.8 Einsatz von weiteren Energiequellen

Da es die ultimative Energy Harvesting Energiequelle nicht gibt, ist je nach Standort zu entscheiden welche Energiequelle genutzt werden sollte.

Standort: freies Feld

Wird ein Schließfach in freier Landschaft aufgestellt, d.h. in ländlichen, unbebauten und nicht bewaldeten Gebieten. So kann Wind als Energielieferant genutzt werden, dabei reichen schon kleine Windgeschwindigkeiten aus, um die benötigte Energie bereitzustellen. Wie in Abbildung 5.8(a) zu sehen ist, kann der Effekt genutzt werden, der beim Umfließen von Luft an einem Gegenstand auftritt. Bei Platzierung im offenen Terrain haben Windanlagen die höchste Effizienz. Die Masthöhe ist dann von geringer Bedeutung. Hierfür eignet sich eine vertikale Windturbine (siehe Abbildung 5.8(b)) eher als ein konventionelles Windrad. Da es Vorteile bei turbulenten Windverhältnissen bietet.



(a) Umströmung eines freistehenden Körpers [23] (b) Vertikale Windturbine [24]

Abbildung 5.8: Energie Harvesting Windkraft

6 Zusammenfassung und Ausblick

In diesem Kapitel wird dem Leser eine Zusammenfassung und eine Bewertung der Ergebnisse präsentiert. Zum Schluss wird noch darauf eingegangen, welche Aufgabenstellungen in weiteren Studien- oder Abschlussarbeiten bearbeitet werden könnten.

6.1 Zusammenfassung

Ziel dieser Arbeit war es für das Entwicklungsboard *M24LR-Discovery Board* eine Software zu schreiben, die es ermöglicht einen Pin einzulesen und den Zustand “Geschlossen” einzunehmen, der erst wieder durch erneute Eingabe des gleichen Pins in den Zustand “geöffnet” überführt wird.

Um die Aufgabe zu lösen wurden im ersten Schritt die Grundlagen und die Funktionsweise des *M24LR-Discovery Board* recherchiert, dann die notwendigen Entwicklungswerkzeuge dafür bestellt und installiert. Danach wurde ein Entwurf der Software in einem Zustandsdiagramm entwickelt und die Software anhand dieser Vorlage implementiert, getestet und verbessert, bis die Funktionsfähigkeit der Anforderungen gegeben war.

Des Weiteren wurden verschiedene Gleichrichterschaltungen mit und ohne Schwingkreis an einer selbst gewickelten Rechteckspule getestet, Messwerte ausgelesen und die maximale Leistung aus der jeweiligen Konstellation berechnet. Dabei stellte sich heraus, dass trotz Verluste über einen Spalt eine Leistung bis zu 22 mW aus der induzierten Spannung eines durch *NFC*-verbundenen Systems übertragen werden kann.

Für ein energieautarkes Schließfach sind diese geringen Leistungen jedoch nicht ausreichend, daher wurde noch nach weiteren Möglichkeiten recherchiert, die je nach Standort, genügend Energie für ein elektronisches Schloss bereitzustellen.

6.2 Ausblick

Die in dieser Arbeit entwickelte Software hat noch Verbesserungspotenzial bei der Geschwindigkeit, mit der der Pin aus dem *NFC-EEPROM* ausgelesen wird. Zur besseren Funktionsfähigkeit ist eine eigens dafür entwickelte Android App notwendig, die während ihrer gesamten Laufzeit das hochfrequente magnetische Feld erzeugt, damit der Mikrocontroller Zeit zur Verarbeitung der Daten hat. Dazu sollte die App in dem ersten Byte der Nutzdaten ein für den Benutzer unsichtbares Dollarzeichen enthalten und nur einen Pin mit vier ASCII-Zeichen verschicken (ausgenommen Leerzeichen).

Als nächster Schritt ist das Konstruieren eines Prototypen zu empfehlen, dessen Schloss vorerst noch mit Batterien betrieben wird. Während des Authentifizierungsvorganges muss es fixiert werden um nicht zu verrutschen, dadurch wäre die Energieversorgung unterbrochen.

Zusätzlich sollten Piezokristallgeneratoren, Windturbinen und ein Energiespeicher im Feldversuch getestet werden. Der Energiespeicher muss außerdem noch schnell aufladbar sein, z.B. ein Folienkondensator, der bei großen Temperaturschwankungen nicht so anfällig wie ein Akkumulator ist.

Selbstverständlich können auch effizientere elektronische Bauteile (z.B. *FRAM*) eingesetzt werden. Der Vorteil eines *FRAM* ist sein extrem niedriger Stromverbrauch.

Literaturverzeichnis

- [1] JOSEF LANGER ; MICHAEL ROLAND: Anwendung und Technik von Near Field Communication (NFC). 1. Auflage. Berlin Heidelberg : Springer, 2010. – ISBN 978-3-642-05496-9
- [2] KRÖNER, Tim: RFID Geschichte. <http://www.rfid-journal.de/rfid-geschichte.html>
- [3] WIKIPEDIA: Inter Frame Spacing. http://de.wikipedia.org/wiki/Inter_Frame_Spacing. Version: 15.05.2013
- [4] WIKIPEDIA: Carrier Sense Multiple Access/Collision Avoidance. http://de.wikipedia.org/wiki/Carrier_Sense_Multiple_Access/Collision_Avoidance. Version: 15.05.2013
- [5] WIKIPEDIA ; KLBOT2 (Hrsg.): Inter Frame Spacing. http://de.wikipedia.org/wiki/Inter_Frame_Spacing. Version: 29.07.2013
- [6] CHIPWORKS: Inside the Samsung Galaxy Nexus GT-I9250. 29.07.2013
- [7] GRÜNINGER ; GRÜNINGER, Landesbildungsserver (Hrsg.): Der Transformator. http://www.schule-bw.de/unterricht/faecher/physik/online_material/e_lehre_1/induktion/trafo1.htm. Version: 29.07.2013
- [8] EKBERT HERING ; ROLF MARTIN ; MARTIN STOHRER: Physik für Ingenieure. 10. Auflage. Heidelberg : Springer-Verlag, 2007. – ISBN 978-3-540-71855-0
- [9] KHAN, Shah G. ; KHAN, Shah G. (Hrsg.): Binary Amplitude Shift Keying. http://www.mathworks.com/matlabcentral/fx_files/30580/1/ASK.jpg. Version: 29.07.2013
- [10] STMICROELECTRONICS (Hrsg.): AN2972 Application note. Genf: STMicroelectronics, Dezember 2012
- [11] WIKIPEDIA: Harvard-Architektur. <http://de.wikipedia.org/wiki/Harvard-Architektur>. Version: 02.05.2013
- [12] WIKIPEDIA ; WOSCH21149 (Hrsg.): I2C. <http://de.wikipedia.org/wiki/I%C2%B2C>. Version: 27.05.2013
- [13] JOACHIM, Michael: I2C-Bus. http://ln.iuk.fh-dortmund.de/~gehard/STA/vortraege/I2C-Bus__Joachim.pdf. Version: 27.05.2013

Literaturverzeichnis

- [14] STMICROELECTRONICS (Hrsg.): UM1589 User manual. Genf: STMicroelectronics, März 2013
- [15] STMICROELECTRONICS (Hrsg.): M24LR-DISCOVERY. Genf: STMicroelectronics, März 2013
- [16] STMICROELECTRONICS (Hrsg.): STM8L151x4, STM8L151x6, STM8L152x4, STM8L152x6 User manual. Genf: STMicroelectronics, November 2012
- [17] STMICROELECTRONICS (Hrsg.): M24LR04E-R Datasheet. Genf: STMicroelectronics, März 2013
- [18] GOSSNER, Stefan: Grundlagen der Elektronik. 7. Auflage. München : Shaker Verlag, Juli 2008. – ISBN 978–3–8265–8825–9
- [19] PLESSIS, Nicolaas D.: SD Association Enables NFC for SD Cards. <http://www.tomshardware.com/news/SD-Association-Card-NFC-SWP,22972.html#>. Version: 22.08.2013
- [20] SCHWAGER, Prof. Dr.-Ing. J.: Informationsportal für Energy Harvesting. <http://www.harvesting-energy.de/>. Version: 30.07.2013
- [21] EKBERT HERING ; GERT SCHÖNFELDER: Sensoren in Wissenschaft und Technik. 1. Auflage. Wiesbaden : Vieweg+Teubner Verlag, 2012. – ISBN 978–3–8348–0169–2
- [22] LANDESAMT FÜR UMWELT, WASSERWIRTSCHAFT UND GEWERBEAUFSICHT RHEINLAND-PFALZ ; Bos, Manfred (Hrsg.): Erschütterungsimmissionen durch Schienenverkehr im Mittelrheintal an der linksrheinischen Bahnstrecke im Ortsbereich 56154 Boppard. <http://www.luwg.rlp.de/>. Version: 06.2011
- [23] NOW.AT/CMS/ <http://www.aee.at>: Empfehlungen für Kleinwindkraftanlagen (KWKA). <http://kleinewindkraft.files.wordpress.com/2011/02/empfehlungen-fuer-kleinwindkraftanlagen.pdf>. Version: 15..10.2010
- [24] CONSTRUCTIONS, Solar: Vertikale Windenergie. <http://www.solar-constructions.com/wordpress/vertikale-windenergie/>. Version: 2013

Listingverzeichnis

4.1	Auszug aus Copy_Data_NFC_EE_to_Int_EE() Abfrage; wenn ein Dollarzeichen vorhanden ist, wird dies sofort im <i>NFC EEPROM</i> überschrieben.	29
6.1	main.c	53
6.2	Copy_Data_NFC_EE_to_Int_EE.c	54
6.3	Copy_Data_NFC_EE_to_Int_EE_and_comp.c	56
6.4	Check_PID_Buffer_Equal.c	58
6.5	Overwrite_dollar_EEPROM.c	58
6.6	OverwriteAll_CODE_EEPROM.c	59
6.7	State_DisplayMessage.c	60
6.8	Gleichrichter.m	60
6.9	Delonschaltung.m	61
6.10	Delonschaltung mit Schwingkreis.m	61

Anhang

Listing 6.1: main.c

```
1 /* Includes -----
   */
2 #include "stm8l15x.h"
3 #include "stm8l_discovery_lcd.h"
4 #include "discover_board.h"
5 #include "vcc_measure.h"
6 #include "I2C_M24LR04E-R.h"
7 #include "I2C_STTS751.h"
8 #include "discover_functions.h"
9 #include "misc.h"
10 #include "delay.h"
11
12
13 static void Copy_Data_NFC_EE_to_Int_EE(void);
14 static uint8_t Check_PID_Buffer_Equal(void);
15 static uint8_t Check_PID_Buffer_nEqual(void);
16 static void Copy_Data_NFC_EE_to_Int_EE_and_comp(void);
17 static uint8_t Check_PID_Buffer_of_dollar(void);
18 static void State_DisplayMessage (uint8_t message[],uint8_t PayloadLength );
19 void LCD_GLASS_DisplayString_1(uint8_t* ptr);
20 static uint8_t Check_Code_Length(void);
21 static void Overwrite_dollar_EEPROM(void);
22 static void OverwriteAll_CODE_EEPROM(void);
23
24
25 EEPROM uint8_t EEInitial = 0;
26 EEPROM uint8_t EE_Buffer_m [5];
27 EEPROM uint8_t EE_Buffer_e [1];
28 EEPROM uint8_t EE_Pay_Lengt [1];
29 EEPROM uint8_t EE_Buffer_11 [1];
30 EEPROM uint8_t EE_Buffer_10 [1];
31 EEPROM uint8_t EE_Buffer_9 [1];
32 EEPROM uint8_t EE_Buffer_8 [1];
33 EEPROM uint8_t EE_Buffer_7 [1];
34 EEPROM uint8_t EE_Buffer_6 [1];
35 EEPROM uint8_t EE_Buffer_5 [1];
36 EEPROM uint8_t EE_Buffer_4 [1];
37 EEPROM uint8_t EE_Buffer_3 [1];
38 EEPROM uint8_t EE_Buffer_2 [1];
39 EEPROM uint8_t EE_Buffer_1 [1];
40 EEPROM uint8_t EE_Buf_Flag [1];
41
42
43 void main(void)
44 {
45     /* deinit I/O ports */
46     DeInitClock();
47     DeInitGPIO();
48
49     /* Select HSI as system clock source */
50     #ifdef USE_HSI
51         CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_HSI);
52         CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_16);
53     #else
54         CLK_SYSCLKSourceSwitchCmd(ENABLE);
55         /* Select 2MHz HSE as system clock source */
56         CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_HSE);
57         CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_4);
58         CLK_HSICmd(DISABLE);
```

Listingverzeichnis

```
59         #endif
60
61         // Initializes the LCD glass
62 LCD_GLASS_Init();
63
64
65         /* LED button init: GPIO set in push pull */
66 GPIO_Init( LED_GPIO_PORT, LED_GPIO_PIN, GPIO_Mode_Out_PP_Low_Fast);
67         // set to 0
68 GPIOE->ODR &= ~LED_GPIO_PIN;
69
70
71     /* Init Bar on LCD all are OFF
72 BAR0_OFF;
73 BAR1_OFF;
74 BAR2_OFF;
75 BAR3_OFF;
76
77     enableInterrupts();
78
79
80     if( Check_Code_Length() == SUCCESS)
81     {
82             if( EE_Buf_Flag[0] == 0x00 ) // geoeffnete Zustand
83             {
84                     GPIO_ResetBits(LED_GPIO_PORT,
85                                     GPIO_Pin_6);
86                     State_DisplayMessage ("OPEN", 4);
87                     Copy_Data_NFC_EE_to_Int_EE(); // Zustands Transaktion von geoffnet nach geschlossen
88
89
90             else if( EE_Buf_Flag[0] == 0xFF ) // geschlossene Zustand
91             {
92                     GPIO_SetBits(LED_GPIO_PORT,GPIO_Pin_6)
93                     ;
94                     State_DisplayMessage ("CLOSED", 6);
95                     Copy_Data_NFC_EE_to_Int_EE_and_comp(); //Zustands Transaktion von geschlossen nach geoffnet
96             }
97         }
98
99         LCD_GLASS_Clear();
100        User_DisplayMessage ("ONLY 4 CHARACTERS ENTER PIN AGAIN", 30);
101
102 }
```

Listing 6.2: Copy_Data_NFC_EE_to_Int_EE.c

```
1 static void Copy_Data_NFC_EE_to_Int_EE(void)
2 {
3     uint8_t *OneByteMemory1 = EE_Buffer_1;
4     uint8_t *OneByteMemory2 = EE_Buffer_2;
5     uint8_t *OneByteMemory3 = EE_Buffer_3;
6     uint8_t *OneByteMemory4 = EE_Buffer_4;
7     uint8_t *OneByteMemory5 = EE_Buffer_5;
8
9     uint16_t ReadAddr_Byt e1 = 0x000D;//D
10    uint16_t ReadAddr_Byt e2 = 0x000E;//E
11    uint16_t ReadAddr_Byt e3 = 0x000F;//F
12    uint16_t ReadAddr_Byt e4 = 0x0010;//10
```

Listingverzeichnis

```
13     uint16_t ReadAddr_Byt e5 = 0x0011; //11
14
15     /***** 1. Byte von EEPROM Lesen *****/
16     M24LR04E_Init();
17
18     M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER , ReadAddr_Byt e1 ,
19                           OneByteMemory1);
20     /*** I2C disable ***/
21     I2C_Cmd(M24LR04E_I2C , DISABLE);
22     CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
23     GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
24     GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
25     *****/
26     EE_Buffer_1[0] = EE_Buffer_1[0];
27
28 if ( EE_Buffer_1[0] == '$')
29 {
30
31         Overwrite_dollar_EEPROM();
32         /***** 1. Byte von EEPROM Lesen *****/
33         M24LR04E_Init();
34
35         M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
36                               ReadAddr_Byt e1 , OneByteMemory1);
37     /*** I2C disable ***/
38     I2C_Cmd(M24LR04E_I2C , DISABLE);
39     CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
40     GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
41     GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
42     *****/
43     EE_Buffer_1[0] = EE_Buffer_1[0];
44
45     /***** 2. Byte von EEPROM Lesen *****/
46     M24LR04E_Init();
47
48     M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
49                           ReadAddr_Byt e2 , OneByteMemory2);
50     //****I2C disable*****
51     I2C_Cmd(M24LR04E_I2C , DISABLE);
52     CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
53     GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
54     GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
55     //*****
56     EE_Buffer_2[0] = EE_Buffer_2[0];
57
58     //***** 3. Byte von EEPROM Lesen *****
59     M24LR04E_Init();
60
61     M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
62                           ReadAddr_Byt e3 , OneByteMemory3);
63     //****I2C disable*****
64     I2C_Cmd(M24LR04E_I2C , DISABLE);
65     CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
66     GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
67     GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
68     //*****
69     EE_Buffer_3[0] = EE_Buffer_3[0];
70
71     //***** 4. Byte von EEPROM Lesen *****
72     M24LR04E_Init();
73
74     M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
75                           ReadAddr_Byt e4 , OneByteMemory4);
76     //****I2C disable*****
```

Listingverzeichnis

```
73         I2C_Cmd(M24LR04E_I2C, DISABLE);
74         CLK_PeripheralClockConfig(CLK_Peripheral_I2C1, DISABLE);
75         GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT, M24LR04E_I2C_SCL_PIN);
76         GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT, M24LR04E_I2C_SDA_PIN);
77         //*****
78         EE_Buffer_4[0] = EE_Buffer_4[0];
79
80
81         //***** 5. Byte von EEPROM Lesen *****
82         M24LR04E_Init();
83
84         M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER,
85                               ReadAddr_Byt5, OneByteMemory5);
86         //****I2C disable*****
87         I2C_Cmd(M24LR04E_I2C, DISABLE);
88         CLK_PeripheralClockConfig(CLK_Peripheral_I2C1, DISABLE);
89         GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT, M24LR04E_I2C_SCL_PIN);
90         GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT, M24LR04E_I2C_SDA_PIN);
91         //*****
92         EE_Buffer_5[0] = EE_Buffer_5[0];
93
94         //***** Speichern in 2. STM8 Internen EEPROM *****
95         EE_Buffer_10[0] = EE_Buffer_5[0];
96         EE_Buffer_9[0] = EE_Buffer_4[0];
97         EE_Buffer_8[0] = EE_Buffer_3[0];
98         EE_Buffer_7[0] = EE_Buffer_2[0];
99
100        //*** Status Flag in EEPROM Speicher auf Geschlossen Setzen ****
101        EE_Buf_Flag[0] = 0xFF;
102
103        OverwriteAll_CODE_EEPROM();
104    }
```

Listing 6.3: Copy_Data_NFC_EE_to_Int_EE_and_comp.c

```
1 static void Copy_Data_NFC_EE_to_Int_EE_and_comp(void)
2 {
3     uint8_t *OneByteMemory1 = EE_Buffer_1;
4     uint8_t *OneByteMemory2 = EE_Buffer_2;
5     uint8_t *OneByteMemory3 = EE_Buffer_3;
6     uint8_t *OneByteMemory4 = EE_Buffer_4;
7     uint8_t *OneByteMemory5 = EE_Buffer_5;
8
9     uint16_t ReadAddr_Byt1 = 0x000D;
10    uint16_t ReadAddr_Byt2 = 0x000E;
11    uint16_t ReadAddr_Byt3 = 0x000F;
12    uint16_t ReadAddr_Byt4 = 0x0010;
13    uint16_t ReadAddr_Byt5 = 0x0011;
14
15 //***** 1. Byte von EEPROM Lesen *****
16
17     M24LR04E_Init();
18
19     M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER, ReadAddr_Byt1,
20                           OneByteMemory1);
21
22     /** I2C disable ***/
23     /*I2C_Cmd(M24LR04E_I2C, DISABLE);
24     CLK_PeripheralClockConfig(CLK_Peripheral_I2C1, DISABLE);
25     GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT, M24LR04E_I2C_SCL_PIN);
26     GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT, M24LR04E_I2C_SDA_PIN);
27     */
28     EE_Buffer_1[0] = EE_Buffer_1[0];
```

Listingverzeichnis

```
29         if ( EE_Buffer_1[0] == '$' )
30     {
31
32         Overwrite_dollar_EEPROM();
33         //***** 1. Byte von EEPROM Lesen damit Dollar in Intern
34         gesloescht ist*****
35         M24LR04E_Init();
36
37         M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
38             ReadAddr_Byt e1 , OneByteMemory1);
39             /* *** I2C disable *** */
40             /* I2C_Cmd(M24LR04E_I2C , DISABLE);
41             CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
42             GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
43             GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
44             *****/
45             EE_Buffer_1[0] = EE_Buffer_1[0];
46
47         //***** 2. Byte von EEPROM Lesen *****
48         M24LR04E_Init();
49
50         M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
51             ReadAddr_Byt e2 , OneByteMemory2);
52             //*****I2C disable*****
53             I2C_Cmd(M24LR04E_I2C , DISABLE);
54             CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
55             GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
56             GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
57             //***** ****
58             EE_Buffer_2[0] = EE_Buffer_2[0];
59
60         //***** 3. Byte von EEPROM Lesen *****
61         M24LR04E_Init();
62
63         M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
64             ReadAddr_Byt e3 , OneByteMemory3);
65             //*****I2C disable*****
66             I2C_Cmd(M24LR04E_I2C , DISABLE);
67             CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
68             GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
69             GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
70             //***** ****
71             EE_Buffer_3[0] = EE_Buffer_3[0];
72
73         //***** 4. Byte von EEPROM Lesen *****
74         M24LR04E_Init();
75         M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
76             ReadAddr_Byt e4 , OneByteMemory4);
77             //*****I2C disable*****
78             I2C_Cmd(M24LR04E_I2C , DISABLE);
79             CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
80             GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT , M24LR04E_I2C_SCL_PIN);
81             GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT , M24LR04E_I2C_SDA_PIN);
82             //***** ****
83             EE_Buffer_4[0] = EE_Buffer_4[0];
84
85         //***** 5. Byte von EEPROM Lesen *****
86         M24LR04E_Init();
87         M24LR04E_ReadOneByte (M24LR16_EEPROM_ADDRESS_USER ,
88             ReadAddr_Byt e5 , OneByteMemory5);
89             //*****I2C disable*****
90             I2C_Cmd(M24LR04E_I2C , DISABLE);
91             CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
```

Listingverzeichnis

```
89         GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SCL_PIN);
90         GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SDA_PIN);
91         //*****
92         EE_Buffer_5[0] = EE_Buffer_5[0];
93
94         //***** Lesen Status Byte *****
95
96         if ( Check_PID_Buffer_Equal() == SUCCESS )
97         {
98             EE_Buf_Flag[0] = 0x00;
99             OverwriteAll_CODE_EEPROM();
100        }
101    else
102    {
103        // CODE nicht gueltig.
104    }
105}
106 }
```

Listing 6.4: Check_PID_Buffer_Equal.c

```
1 static uint8_t Check_PID_Buffer_Equal(void)
2 {
3
4     if((EE_Buffer_10[0] == EE_Buffer_5[0]) &&
5         (EE_Buffer_9[0] == EE_Buffer_4[0]) &&
6         (EE_Buffer_8[0] == EE_Buffer_3[0]) &&
7         (EE_Buffer_7[0] == EE_Buffer_2[0]))
8     )
9     {
10        return SUCCESS;
11    }
12    else return ERROR;
13 }
```

Listing 6.5: Overwrite_dollar_EEPROM.c

```
1 static void Overwrite_dollar_EEPROM(void)
2 {
3     uint8_t *OneByteMemory1 = EE_Buffer_1;
4     uint8_t *OneByteMemory2 = EE_Buffer_2;
5     uint8_t *OneByteMemory3 = EE_Buffer_3;
6     uint8_t *OneByteMemory4 = EE_Buffer_4;
7     uint8_t *OneByteMemory5 = EE_Buffer_5;
8
9     uint16_t ReadAddr_Byt1 = 0x000D;
10    uint16_t ReadAddr_Byt2 = 0x000E;
11    uint16_t ReadAddr_Byt3 = 0x000F;
12    uint16_t ReadAddr_Byt4 = 0x0010;
13    uint16_t ReadAddr_Byt5 = 0x0011;
14
15
16    //***** 1. Byte in EEPROM ueberschreiben *****
17    M24LR04E_Init();
18    M24LR04E_WriteOneByte (M24LR16_EEPROM_ADDRESS_USER ,
19                           ReadAddr_Byt1, (uint8_t)( 0x20 ) );
20    //****I2C disable*****
21    I2C_Cmd(M24LR04E_I2C, DISABLE);
22    CLK_PeripheralClockConfig(CLK_Peripheral_I2C1, DISABLE);
23    GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SCL_PIN);
24    GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SDA_PIN);
25 }
```

Listingverzeichnis

Listing 6.6: OverwriteAll_CODE_EEPROM.c

```
1 static void OverwriteAll_CODE_EEPROM(void)
2 {
3
4     uint8_t *OneByteMemory2 = EE_Buffer_2;
5     uint8_t *OneByteMemory3 = EE_Buffer_3;
6     uint8_t *OneByteMemory4 = EE_Buffer_4;
7     uint8_t *OneByteMemory5 = EE_Buffer_5;
8
9     uint16_t ReadAddr_Byte2 = 0x000E;
10    uint16_t ReadAddr_Byte3 = 0x000F;
11    uint16_t ReadAddr_Byte4 = 0x0010;
12    uint16_t ReadAddr_Byte5 = 0x0011;
13
14    delayLFO_ms (1);
15    //*****1. Byte in EEPROM ueberschreiben *****
16    M24LR04E_Init();
17    M24LR04E_WriteOneByte (M24LR16_EEPROM_ADDRESS_USER ,
18                           ReadAddr_Byte2, (uint8_t)( 0x20 ) );
19    //****I2C disable*****
20    I2C_Cmd(M24LR04E_I2C , DISABLE);
21    CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
22    GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SCL_PIN);
23    GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT ,M24LR04E_I2C_SDA_PIN);
24    //*****2. Byte in EEPROM ueberschreiben *****
25
26    delayLFO_ms (1);
27
28    //*****2. Byte in EEPROM ueberschreiben *****
29    M24LR04E_Init();
30    M24LR04E_WriteOneByte (M24LR16_EEPROM_ADDRESS_USER ,
31                           ReadAddr_Byte3, (uint8_t)( 0x20 ) );
32    //****I2C disable*****
33    I2C_Cmd(M24LR04E_I2C , DISABLE);
34    CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
35    GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SCL_PIN);
36    GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT ,M24LR04E_I2C_SDA_PIN);
37
38    delayLFO_ms (1);
39    //*****2. Byte in EEPROM ueberschreiben *****
40    M24LR04E_Init();
41    M24LR04E_WriteOneByte (M24LR16_EEPROM_ADDRESS_USER ,
42                           ReadAddr_Byte4, (uint8_t)( 0x20 ) );
43    //****I2C disable*****
44    I2C_Cmd(M24LR04E_I2C , DISABLE);
45    CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
46    GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SCL_PIN);
47    GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT ,M24LR04E_I2C_SDA_PIN);
48
49    delayLFO_ms (1);
50
51    M24LR04E_Init();
52    M24LR04E_WriteOneByte (M24LR16_EEPROM_ADDRESS_USER ,
53                           ReadAddr_Byte5, (uint8_t)( 0x20 ) );
54    //****I2C disable*****
55    I2C_Cmd(M24LR04E_I2C , DISABLE);
56    CLK_PeripheralClockConfig(CLK_Peripheral_I2C1 , DISABLE);
57    GPIO_HIGH(M24LR04E_I2C_SCL_GPIO_PORT ,M24LR04E_I2C_SCL_PIN);
58    GPIO_HIGH(M24LR04E_I2C_SDA_GPIO_PORT ,M24LR04E_I2C_SDA_PIN);
59
```

Listingverzeichnis

```
60 }
```

Listing 6.7: State_DisplayMessage.c

```
1 static void State_DisplayMessage (uint8_t message[], uint8_t PayloadLength )
2 {
3
4     //Switch the clock to LSE and disable HSI
5     #ifdef USE_LSE
6         CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_1);
7         CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_LSE);
8         CLK_SYSCLKSourceSwitchCmd(ENABLE);
9         while (((CLK->SWCR)& 0x01)==0x01);
10        CLK_HSICmd(DISABLE);
11        CLK->ECKCR &= ~0x01;
12    #else
13        CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_1);
14        CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_LSI);
15        CLK_SYSCLKSourceSwitchCmd(ENABLE);
16        while (((CLK->SWCR)& 0x01)==0x01);
17        CLK_HSICmd(DISABLE);
18        CLK->ECKCR &= ~0x01;
19    #endif
20
21    LCD_GLASS_DisplayString_1(message);
22
23    #ifdef USE_HSI
24        //Switch the clock to HSI
25        CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_16);
26        CLK_HSICmd(ENABLE);
27        while (((CLK->ICKCR)& 0x02)!=0x02);
28        CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_HSI);
29        CLK_SYSCLKSourceSwitchCmd(ENABLE);
30        while (((CLK->SWCR)& 0x01)==0x01);
31    #else
32        CLK_SYSCLKDivConfig(CLK_SYSCLKDiv_2);
33        // Select 2MHz HSE as system clock source
34        CLK_SYSCLKSourceConfig(CLK_SYSCLKSource_HSE);
35        // wait until the target clock source is ready
36        while (((CLK->SWCR)& 0x01)==0x01);
37        // wait until the target clock source is ready
38        CLK_SYSCLKSourceSwitchCmd(ENABLE);
39    #endif
40 }
```

Listing 6.8: Gleichrichter.m

```
1 clear all; close all; clc;
2
3 x = 0:0.0001:15;
4 y= -0.000187*x + 0.002705;
5
6 x1 = [0,7.25]; % p1x/p2x
7 y1 = [0.001353,0.001353];% p1y/p2y
8
9 x2 = [7.25,7.25];
10 y2 = [0,0.001353];
11
12 figure(1)
13 hold on
14 plot(x,y,'LineWidth',2)
15 plot(x1,y1,'LineWidth',2,'Color','red');
16 plot(x2,y2,'LineWidth',2,'Color','red'); %[7.25,7.25],[0,0.001353]
17 plot(7.25,.001353,'*','LineWidth',4,'Color','green');
```

Listingverzeichnis

```
18 set(gca,'XTick',0:.5:15)
19 grid on
20 axis([0 15 0 0.003]) %range
21 xlabel('Spannung U in V');
22 ylabel('Strom I in A');
23 title('Leistungsverlauf')
24 legend('Strom ueber Spannung I(U)', 'Maximale Leistung P=U*I')
25 text(7.27 , 0.00275/2,'P_m_a_x = 7,25 V * 1,4 mA = 10,15 mW')
```

Listing 6.9: Delonschaltung.m

```
1 clear all; close all; clc;
2
3 x = 0:0.0001:27;
4 y= -0.000077*x + 0.002002;
5
6 x1 = [0,26/2]; % p1x/p2x
7 y1 = [0.002002/2,0.002002/2];% p1y/p2y
8
9 x2 = [26/2,26/2];
10 y2 = [0,0.002002/2];
11
12 figure(1)
13 hold on
14 plot(x,y,'LineWidth',2)
15 plot(x1,y1,'LineWidth',2,'Color','red');% Horizontal
16 plot(x2,y2,'LineWidth',2,'Color','red'); %[7.25,7.25],[0,0.001353]Vertikal
17 plot(26/2,0.002002/2,'*','LineWidth',4,'Color','green');
18 set(gca,'XTick',0:1:27)
19 grid on
20 axis([0 27 0 0.003]) %range
21 xlabel('Spannung U in V');
22 ylabel('Strom I in A');
23 title('Leistungsverlauf')
24 legend('Strom ueber Spannung I(U)', 'Maximale Leistung P=U*I')
25 text(26.2/2 , 0.0022/2,'P_m_a_x = 13 V * 1 mA = 13 mW')
```

Listing 6.10: Delonschaltung mit Schwingkreis.m

```
1 clear all; close all; clc;
2
3 x = 0:0.0001:33.28;
4 y= -0.0000799*x + 0.00266;
5
6 x1 = [0 , 33.28/2]; % p1x/p2x
7 y1 = [0.00266/2 , 0.00266/2];% p1y/p2y
8
9 x2 = [33.28/2 , 33.28/2];
10 y2 = [0 , 0.00266/2];
11
12 figure(1)
13 hold on
14 plot(x,y,'LineWidth',2)
15 plot(x1,y1,'LineWidth',2,'Color','red');% Horizontal
16 plot(x2,y2,'LineWidth',2,'Color','red'); %[7.25,7.25],[0,0.001353]Vertikal
17 plot(33.28/2,0.00266/2,'*','LineWidth',4,'Color','green');
18 set(gca,'XTick',0:1:34)
19 grid on
20 axis([0 34 0 0.003]) %range
21 xlabel('Spannung U in V');
22 ylabel('Strom I in A');
23 title('Leistungsverlauf')
24 legend('Strom ueber Spannung I(U)', 'Maximale Leistung P=U*I')
25 text(33.4/2,0.0028/2,'P_m_a_x = 16,64 V * 1,33 mA = 22,13 mW')
```

Listingverzeichnis

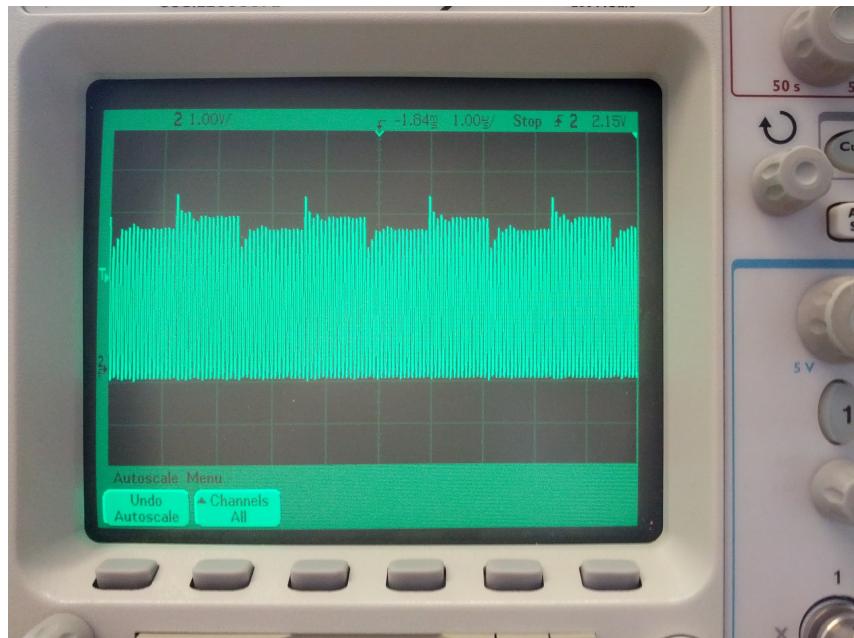


Abbildung 6.1: *Amplitude Shift Keying*

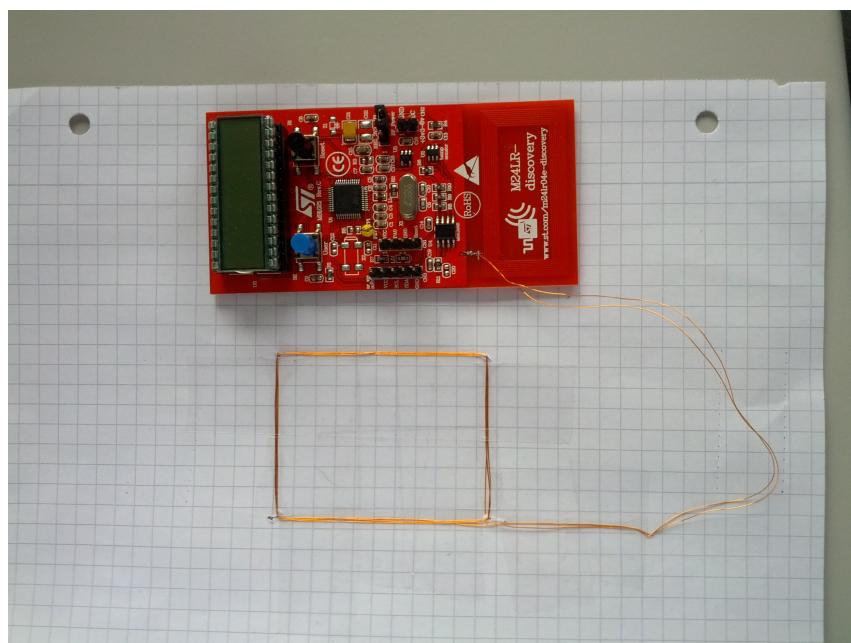


Abbildung 6.2: *M24LR-Discovery* mit externer Spule

Listingverzeichnis

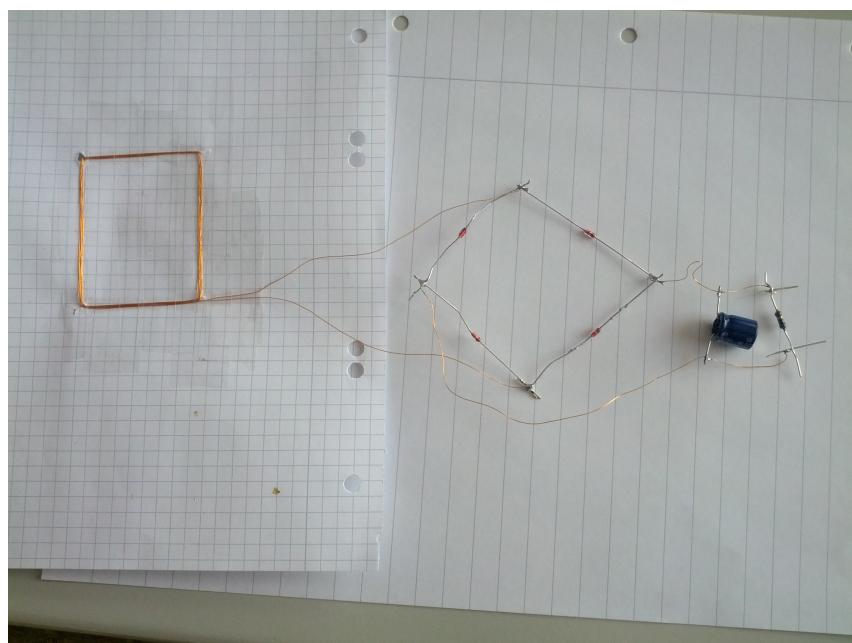


Abbildung 6.3: Gelötete Greatz-Gleichrichterschaltung

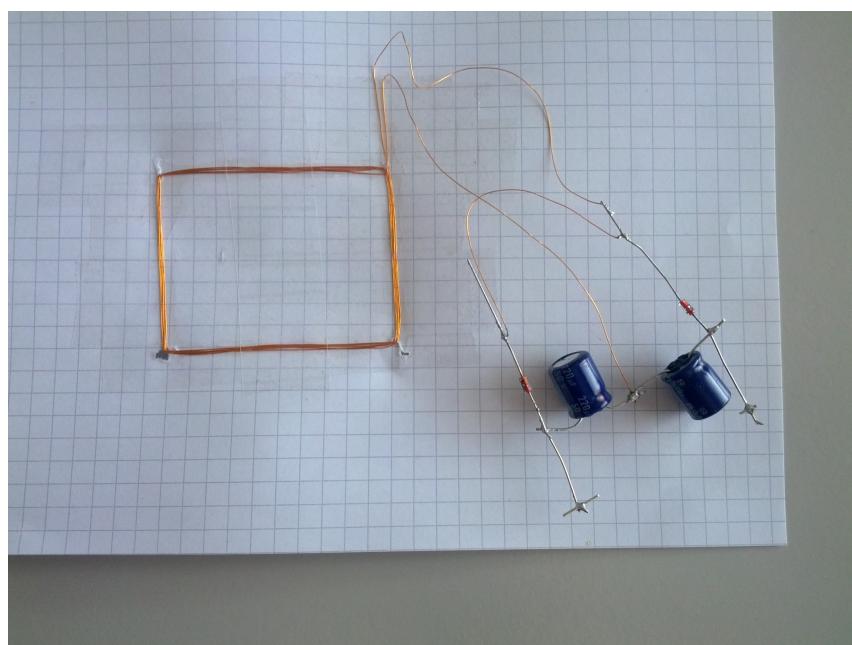


Abbildung 6.4: Gelötete Delonschaltung