DIPLOMARBEIT

Energieautarkes Schlosssystem mittels kontaktloser Nahfunkkommunikation

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Diplomingenieurs unter der Leitung von

Univ. Prof. Dr. habil. Christoph Grimm
Dipl.-Ing. Dr. techn. Stefan Mahlknecht

eingereicht an der
Fakultät für Elektrotechnik und Informationstechnik
Institut für Computertechnik (Nr. 384)
der Technischen Universität Wien
durch

B.Sc. Lukas Obletter
Matr.Nr. 0425222
Thalhaimergasse 47/29
1160, Wien

Wien, am 30. Oktober 2011 _____

Kurzfassung

In der vorliegenden Arbeit wird ein Gesamtkonzept für ein Fahrradschlosssystem für öffentliche Fahrradabstellplätze ausgearbeitet. Es werden die Ansprüche an das System bezüglich Sicherheit, Benutzerfreundlichkeit, Schlüsselsynchronisierung und Administration ausfindig gemacht und ein Lösungsvorschlag zur Umsetzung des Konzeptes vorgestellt. Die Realisierbarkeit des Schlosssystems hängt dabei maßgeblich vom Energieverbrauch der batteriebetriebenen NFC-Lesegeräte (Near Field Communication) ab, die mit tragbaren Wartungskosten betrieben werden sollten. Unter Berücksichtigung der Systemanforderungen werden verschiedene Energiesparmaßnahmen ausgearbeitet, welche ein wartungsfreies Betreiben der Schlösser über mehrere Jahre ermöglicht.

Abstract

In the present work an overall concept for a bike lock system for bicycle parking lots is worked out. With the focus on safety, usability, key synchronization and administration a possible solution is worked out. The feasibility of the lock system depends on the energy consumption of the NFC readers (Near Field Communication), because these are battery powered and should be operated with portable maintenance costs. Complying with the requirements, methods to save energy will be developed, which allow an operation of the locks for several years.

Inhaltsverzeichnis

Abkürzungen

ASK	Amplitude Shift Keying
CMSIS	Cortex Microcontroller Software Interface Standard
CSS	Cascading Style Sheets
EAS	Electronic Article Surveillance
ECMA	European Computer Manufacturers Association
ELF	Executable and Linkable Format
ER	Entity-Relationship
FDMA	Frequency Domain Multiple Access
FDX	Full Duplex
FTDI	Future Technology Devices International
FTP	File Transfer Protocol
GDB	GNU Debugger
GPL	General Public License
GSM	Global System for Mobile Communications
HAL	Hardware Abstraction Layer
HDX	Half Duplex
HF	High Frequency
HSU	High Speed Usart
HTML	Hyper Text Markup Language
ID	Identifier, Identification
IDE	Integrated Development Environment
JTAG	Joint Test Action Group
JRE	Java Runtime Environment
LAMP	Linux Apache MySQL PHP
LED	Light Emitting Diode
LLCP	Logical Link Control Protocol
MSB	Most Significant Bit
MVC	Model View Controller
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
NFCIP	Near Field Communication Interface and Protocol

NRZ	Non Return to Zero
OCD	On Chip Debugger
P2P	Peer to Peer
PCD	Proximity Coupling Device
PHP	PHP Hypertext Preprocessor
PICC	Proximity Integrated Circuit Card
RFID	Radio Frequency Identification
SDMA	Spatial Division Multiplex Access
SEQ	Sequential Multiplex
SHA	Secure Hash Algorithm
SQL	Structured Query Language
TDMA	Time Domain Multiple Access
URL	Uniform Resource Locator
USART	Universal Asynchronous Receiver / Transmitter
XML	Extensible Markup Language

1. Einleitung

1.1 Zusammenfassung

Ziel dieser Arbeit ist es, ein Gesamtkonzept für ein Schlosssystem mit kontaktlosen Schlüsseln zu entwerfen. Im Speziellen wird auf ein Fahrradschlosssystem für öffentliche Fahrradabstellplätze eingegangen, da diese Spezialanwendung mit den meisten Herausforderungen konfrontiert ist.

Es wird angenommen, dass die Schlösser in einer Stadt verteilt an den bestehenden Fahrradständern montiert werden. Die Bürger können diese sicheren Schlösser für das eigene Fahrrad verwenden. So kann dieses mit einer kontaktlosen RFID-Karte (Target) oder mit einem NFC-Handy abgesperrt werden. Nur jenes Target, welches das Schloss abgesperrt hat, ist dazu berechtigt, dieses auch wieder zu öffnen.

Analysiert wird dabei die Machbarkeit eines solchen Systems. Hohe Ansprüche werden hier auf die Energieeffizienz gesetzt, da die NFC-Schlösser über keine Stromzufuhr verfügen. Voraussetzung für einen Einsatz mit tragbaren Kosten ist die Ausrüstung der Schlösser mit einem kleinen Batteriepack, was dessen wartungsfreie Nutzung über mehrere Jahre erlaubt. Damit gilt es, die Hardware so energiesparend wie möglich zu realisieren und die Authentifizierungsvorgänge, sowie die verschiedenen Prozessabläufe so effizient wie möglich durchzuführen.

Weiters hat die Sicherheit hohe Priorität. Sabotage sollte weitestgehend ausgeschlossen werden. Zusätzlich muss der Administrator des Systems neue Benutzer anlegen dürfen und diese im Fall von Missbrauch auch wieder deaktivieren können. Dabei muss sichergestellt werden, dass alle Schlüssel auf allen Schlössern über das Handynetz richtig aktualisiert werden.

1.2 Historische Entwicklung

Während des zweiten Weltkrieg wurde erstmals die Radio-Frequenz-Identifikation (RFID) zum Erkennen von Freund- bzw. Feindflugzeugen eingesetzt. Die Technologie wurde 1937 vom United States Naval Research Laboratory entwickelt. Dabei wurde eine Yagi-Antenne auf einer Schusswaffe montiert. Nach dem Senden einer Identifikationsanfrage an des Flugzeug antwortete dieses mit einem hellen Lichtstrahl. Dabei wurde allerdings nur sichergestellt, dass es sich nicht um ein feindliches Flugzeug handelte. Eine vollständige Identifikation des Flugzeuges war zu dieser Zeit noch nicht möglich. Ein großer Nachteil dieses Systems war, dass bei einem Defekt des Senders bzw. des Transponders, ein Flugzeug sofort als Feind eingestuft wurde.

Das in Großbritannien 1939 entwickelte „IFF Mark III“ wurde in Kombination mit den bestehenden Radargeräten für eine genaue Ortsangabe verwendet. Dabei kam erstmals eine Dipolantenne zum Einsatz und die Flugzeuge antworteten per Funk auf einer Frequenz zwischen 157 MHz und 187 MHz.

Das 1944 vorgestellte „Mark V“ zeigte deutliche Verbesserungen in der Reichweite und der Genauigkeit. Außerdem konnten damit auch schnelle Flugzeuge mit über 280km/h erkannt werden. Durch mehrere Funkkanäle wurde es möglich, mehrere Flugzeuge gleichzeitig zu erkennen. Auch wurde erstmals mit einem „Code-of-the-day“ eine sicherere Identifikation gewährleistet.

1967 erschien zum ersten Mal ein EAS-System (Electronic Article Surveillance), wobei ein 1-bit Transponder in einem magnetischen Feld erkannt wird. In den darauffolgenden Jahren wurde das EAS-System auf sichere Detektion verbessert, wobei auch versucht wurde Fehldetektionen durch Verwenden weiterer harmonischer Felder soweit wie möglich auszuschließen. Dieses System wird bis heute als Diebstahlschutz in Geschäften eingesetzt.

1973 wurden erste Mautsysteme mit RFID Technologie vorgestellt. 1987 wurde diese erstmals in Norwegen für Busse im Nahverkehr eingesetzt.

Im Jahr 1979 wurde die RFID-Technologie zum Erkennen von Wild- und Zuchttieren eingesetzt. Dabei setzte man bereits passive Transponder ein, welche über eine eindeutige Kennzahl verfügten.

Durch die Halbleiterindustrie wurde das Verkleinern der passiven Transponder und eine billige Herstellung möglich. Damit wurden ab etwa 1990 viele neue Einsatzgebiete erkannt und der Einsatz der RFID-Technologie wirtschaftlich interessant.

Die ersten vollständigen ISO-Standards wurden 1996 veröffentlicht. So wurde die Grundlage für Interoperabilität geschaffen.

2004 wird durch die Gründung des NFC-Forums (Near Field Communication) die RFID-Technologie erweitert (siehe Kapitel 2.1). Dabei wurde die Kompatibilität zu den bestehenden Standards erhalten. Erstmals wurde durch die ECMA-Standards auch eine aktive Kommunikation zwischen zwei Kommunikationspartnern festgehalten. Auf diese Weise kann jedes NFC-Gerät auch als

Lesegerät oder als passiver Transponder fungieren. Ein NFC-Gerät kann mehrere Transponder beinhalten. Somit fallen Kosten für die Produktion von Transpondern komplett weg. Es muss lediglich eine Transponderkonfigurationsdatei auf das NFC-Gerät geladen werden. Die Grundlage für eine sichere und kostengünstige Identifikationstechnologie für die Zukunft wurde geschaffen und die möglichen Einsatzgebiete schienen schier unendlich.

Verschiedene Versuche von Nokia, Samsung, Motorola und LG die NFC-Technologie in Handys einzubauen waren bis 2010 nicht erfolgreich und fanden keinen Durchbruch.

Wieso dieser Durchbruch allerdings ab dem Jahr 2011 bis 2013 gelingen könnte wird im Kapitel 1.3 diskutiert [FRA07] [LANDT05].

1.3 Zukunftsentwicklung

Obwohl es über die genaue zukünftige Entwicklung natürlich nur Vermutungen gibt, stehen allerdings bereits einige Tatsachen fest. Die Preise der RFID-Targets werden künftig weiter stark sinken und die Chips werden immer kleiner werden. Durch die erhöhte Nachfrage werden auch größere Mengen produziert, was ein weiteres Sinken der Kosten in der Produktion der Targets zur Folge haben wird.

Sobald es für größere Handelsketten wirtschaftlich interessant ist die aktuellen Barcodes mit RFID-Targets zu ersetzen, wird die Technologie ihren großen Durchbruch feiern. Kleinere Geschäfte werden gezwungen sein mit dem Trend mitzuhalten. Die Realisierung automatischer Kassen in Supermärkten wären kein Problem mehr. Viele, bis heute unbekannte Einsatzgebiete für die RFID-Technologie werden somit erschlossen werden. Die Vorteile die durch die Automatisierung mithilfe der RFID-Technologie für Großunternehmen entstehen, werden über die Skepsis der Datenschützer erhaben sein. Machtpositionen am Markt werden sich nicht vermeiden lassen, da ein Gerät beispielsweise so gebaut werden kann, dass seine Funktionalität nur mit originalen Ersatzteilen gegeben ist. Die RFID Technik wird eine großartige Chance in der Bekämpfung von Produktfälschungen sein. Auf diese Weise wird auch das Einlösen einer Garantie auf ein Produkt vereinfacht.

Die größten Probleme werden in Zukunft die Anpassung der gesetzlichen Grundlage, sowie die Standardisierung sein. Als einzige Lösung für eine gute Interoperabilität sehen hier viele eine „Open Source Middle Ware“ Lösung. Die „Libnfc.org Community“ arbeitet an einem solchen Projekt. Ziel ist es eine plattformunabhängige Library zu erstellen, welche die meisten NFC-Chips unterstützt und ein vollständiges Programmierinterface für die Applikationsentwicklung bietet.

Die RFID und die NFC-Technologie werden nicht nur für Konzerne interessant sein, sondern für jede Privatperson. Sie wird den Alltag erobern. Es ist absehbar, dass die NFC-Technologie am Handy die traditionelle Brieftasche, sowie den Schlüsselbund ablösen wird, da das NFC am Handy im

Vergleich insgesamt viel sicherer eingestuft werden kann. Schlüssel, Tickets und Dokumente können über Internet einfach und sicher mit Familienmitgliedern und Freunden geteilt werden. Die grenzenlosen Möglichkeiten werden eine Revolution in der Zahlungsart, im Ticketing und bei Zutrittsbeschränkungen hervorrufen. Bedingung dafür ist, dass die Handyhersteller einen NFC-Chip in die Geräte einbauen. Nokia, BlackBerry, HTC, Samsung, Sony Ericsson, Motorola, Apple und weitere Handyhersteller haben für die kommenden Handys den Einsatz eines NFC-Chips bestätigt. Erstmals wurde im Sommer 2011 in den Testmärkten New York und San Francisco die Bezahlung über das Handy erfolgreich durch Google Wallet eingesetzt.

Ein großes Problem für ein komplettes NFC-Ökosystem ist sicherlich die Energieeffizienz der Lesegeräte. Da NFC auch für entlegene Ortschaften einsatzfähig sein soll, müssen Lesegeräte mit einem Minimum an Energie auskommen und wartungsfrei über mehrere Jahre funktionieren. Die folgende Arbeit beschäftigt sich unteranderem mit genau dieser Problematik, wobei eine konzeptuelle Lösung ausgearbeitet wird, die für viele Bereiche das Energieproblem tragbar macht [SUTH05] [JANNASCH04] [ONDRUS06].

1.4 Datenschutz und soziale Akzeptanz

Durch die weite Verbreitung der RFID-Technologie und der kommenden NFC-Technologie werden die Proteste der Datenschützer immer lauter.

Da mindestens die Identifikationsnummer eines Targets unbemerkt ausgelesen werden kann, besteht die Gefahr, dass ein Benutzer unwillkürlich Information über sich Preis gibt. Für Marktzwecke können Datenmissbraucher Gewohnheitsprofile von Bürgern erstellen, da diese mit dem gleichen Target womöglich öfters unbewusst erfasst werden. Da Targets nicht nur in Form von Kreditkarten, sondern ebenfalls unwissentlich in den verschiedensten Produkten, wie Kleidungsstücken oder Schlüsselanhängern enthalten sind, hat der Datenmissbraucher durch das Kombinieren verschiedener Identifikationsnummern eine weitere Sicherheit, dass es sich immer um die gleiche Person handelt. Gibt dieser Benutzer noch persönliche Information frei, möglicherweise durch eine Kreditkartenzahlung, so sind fast alle europäischen Datenschutzbestimmungen verletzt.

Für den Bürger besteht keine Möglichkeit sich dagegen zu schützen. Die RFID-Targets müssten lokalisiert und anschließend zerstört werden. Wobei Targets, welche in Produkte integriert sind, sehr schwer entfernt werden können, ohne dem Produkt Schaden zuzufügen. Die Targets müssten beim Verkauf der Ware vom Verkäufer deaktiviert werden, was allerdings nur selten der Fall ist.

Es ist absehbar, dass in naher Zukunft in allen Produkten bzw. Verpackungen aus logistischen Gründen ein RFID-Target enthalten sein wird. So können Optimierungen bei Transport und Vertrieb leicht durchgeführt werden. Nebenbei kann damit die Echtheit und die Herkunft des Produktes

nachgewiesen werden. Für den Bürger wird es unausweichlich, Produkte mit integrierten RFID-Targets zu erwerben und bei sich zu haben [SUHR04] [KERN07, S.7-8].

Der Unterschied zu einem NFC-Gerät ist, dass durch eine Sperre der Digitalen Brieftasche am Handy mit einem PIN Code keine Information über die beinhalteten Tag-Files frei geben wird.

Mit einer zusätzlichen Applikation am Handy könnte der Benutzer die Targets lokalisieren und deaktivieren, falls vom Hersteller eine öffentliche Deaktivierung implementiert ist.

Die Targets in den Waren könnten auch sofort beim Kauf an der Kasse deaktiviert werden.

Demnach sollte der Benutzer nur dann Information über sich preisgeben, wenn er es willkürlich über die digitale Brieftasche am Handy betätigt.

Laut Umfragen schätzen Unternehmen, welche in der RFID-Branche tätig sind, Datenschutz zu 41% als „sehr wichtig“ und 44% als „wichtig“ ein [ANDR07].

Laut einer Umfrage, welche in Österreich im Jahr 2006 mit 400 Teilnehmern verschiedener Kategorien durchgeführt wurde, ist der Begriff RFID nur unter 11,5% der Befragten bekannt. Nach Erläuterung der RFID-Technik wurden die verschiedenen Anwendungen vor allem in der Automatisierung mit der Gesamtnote 1,42 beurteilt. 30% der Befragten ist es wichtig, dass die Targets bei der Weitergabe an den Kunden deaktiviert werden, 40% sind unentschlossen und weitere 30% sind dagegen. Nachdem die Befragten über die Nachteile der Technik, sowie über die Probleme mit dem Datenschutz aufgeklärt wurden, korrigierten 66% die Gesamtnote von 1,42 um einen ganzen Notengrad nach unten. Männer stehen allgemein skeptischer als Frauen zur RFID-Technologie. Auch nimmt die Skepsis mit höherem Ausbildungsgrad zu [ENGELH06, S.104-122].

Damit ist die soziale Akzeptanz nur teilweise gegeben. Man erkennt aus der Studie, dass viel Aufklärungsarbeit notwendig ist. Das Vertrauen der Bürger muss für die NFC-Technik erst gewonnen werden. Dies wird in Zukunft von wenigen Unternehmen wie Google, Apple und HP abhängig sein. Es hängt davon ab, wie diese die Softwareanwendungen sowie die SaaS (Software as a Service) gestalten und wie sehr bei der Implementierung die Sicherheit und der Datenschutz in den Vordergrund gestellt werden.

Im Bezug auf die folgende Arbeit ist zu erwähnen, dass die Implementierung der Fahrradschlösser so erfolgen muss, dass nicht authentifizierte Targets nicht gespeichert werden dürfen. Authentifizierte Targets werden am Zentralserver gespeichert, sollten allerdings nur für statistische Zwecke zur Verbesserung des Angebot verwendet werden.

2. Stand der Technik – Near Field Communication

2.1 Standardisierung

Mit der Gründung des NFC Forums www.nfc-forum.org durch die Unternehmen Sony, Nokia und NXP Semiconductors im Jahr 2004 wurde die Basis zur Standardisierung der NFC-Technologie geschaffen. Im März 2011 waren es bereits 140 Forum-Mitglieder mit dem gemeinsamen Interesse, eine einheitliche interoperable Technologie für Nahfunkkommunikation zu gewährleisten [LANGER2010, S88].

Die größten Ziele des NFC Forums sind damit eine modular definierte standardisierte Architektur zu schaffen, die Entwicklung von NFC Anwendungen zu verbreiten, NFC Produkte zu zertifizieren und die Unternehmen sowie Private über NFC gründlich zu informieren [NFCFORUM, aboutus].

Die Standardisierung geht dabei von den physikalischen Eigenschaften bis zum Format der ausgetauschten Daten NDEF (NFC Data Exchange Format).

Die ECMA International wurde somit beauftragt, den Standard für NFC auszuarbeiten.

Die Standardisierung von NFC versucht, auf bestehende Technologien aufzubauen und die Kompatibilität zu älteren kontaktlosen Datenübertragungstechniken für kurze Distanzen zu erhalten. Dazu gehören die RFID-Systeme MIFARE (ISO/IEC 14443 Typ A), ISO/IEC 14443 Typ B und FeliCa (JIS X 6319-4) [LANGER2010, S89-90].
Der Unterschied zwischen Typ A und Typ B liegt bei der Kommunikationsschnittstelle. Nur eines der beiden muss durch ein Target unterstützt werden, während ein Lesegerät beide Typen unterstützen muss.

Beim ISO/IEC 14443 Typ A werden die Daten im Basisband millercodiert und dann vom Lesegerät zum Target mit einer 100%-igen ASK (Amplitude Shift Keying) auf dem HF-Signal moduliert. Vom Target zum Lesegerät passiert die Kommunikation durch binäre Lastenmodulation auf einem Hilfsträger mit der Frequenz f_h=847 kHz.

ISO/IEC 14443 Typ B hingegen hat eine NRZ (Non Return to Zero) Codierung und eine ASK-Modulation von 10%. Vom Target zum Lesegerät wird auch hier mit Lastenmodulation gearbeitet. Der Hilfsträger mit der Frequenz fh=847 kHz wird mit einer 180°-Phasenumtastung moduliert.

Auch im Antikollisionsmechanismus und bei der Kommunikationsinitialisierung sind bei beiden Typen Unterschiede in der Norm zu finden [FINKENZELLER2002, S.245-S.250].

NFCIP-2 (Near Field Communication Interface and Protocol): Im Standard ECMA-352 (NFCIP-2), welcher dem ISO/IEC21481 entspricht, wird der Mechanismus, wie der Kommunikationsmodus ausgewählt und zwischen ECMA-340, ISO/IEC 14443 oder ISO/IEC 15693 (Vicinity Card System) selektiert wird, spezifiziert [ECMA352, S.1].

Vom NFC-Forum ist aber keine Interoperabilität mit dem Vicinity Card System geplant [NFCFO-RUM].

NFCIP-1: Wird im Standard ECMA-340 festgehalten und entspricht dem Standard ISO/IEC18092. Hier werden die Kommunikationsmodi, darunter der Frequenzbereich, die Modulationsarten, Codierungen, Datenübertragungsgeschwindigkeiten und Initialisierungsschemas für die NFC-Kommunikation spezifiziert. In diesem Standard werden auch die Zusammenschlüsse mit den speziellen Testmethoden beschrieben, die in den Standards ECMA-356 (RF Interface Test Methods) und ECMA-362 (Protocol Test Methods) festgehalten sind [ECMA340, S.1].

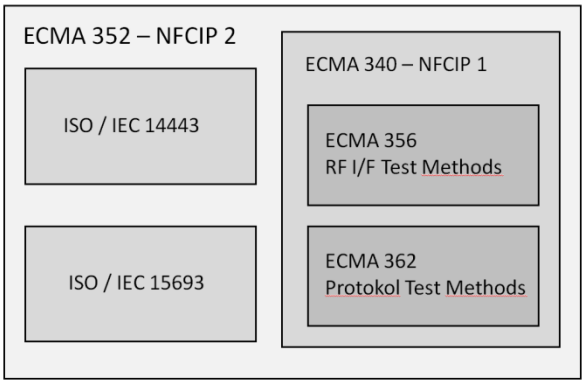


***Abbildung 1**: Zusammenspiel der RFID-Standards ISO / IEC 14443 und ISO/IEC 15693 mit der NFC-Standardisierung ECMA 340 und ECMA 352 [ECMA340]*

Die NFC-Technologie unterstützt insgesamt vier Tagtypen. Dabei wurde darauf geachtet, dass keine von diesen auf proprietäre Lösungen zurückgreift und dass jeder Hersteller alle Tagtypen herstellen kann. Alle basieren auf dem Standard ISO 14443 Typ A bzw. Typ B und Sony FeliCa, wie im Standard ISO 18092 festgehalten wird. Die vier Tagtypen unterscheiden sich im Format, der Kapazität, bei den Ansprechbefehlen und im Speicheraufbau:

Technologie	Hersteller	Normiert durch	Speicherkapazität
Topaz	Innovision	ISO 14443A	Max. 2048 Byte
MIFARE Ultralight	NXP	ISO 14443A	Max. 2048 Byte
FeliCa	Sony	JIS X 6319-4	Max. 1 MByte
Smartcards, APDU-basiert	-	ISO 14443A, ISO 7816-4	Max. 512 MByte

***Tabelle 1**: Vergleich von Smartcards verschiedener Technologieen*

NDEF (NFC Data Exchange Format) beschreibt hingegen ein binäres Format wie Nutzdaten übertragen werden. Dabei kann in einem Datenpaket eines oder mehrere Datentypen enthalten sein. Diese können URIs, MIME Media, Smart Poster Type, Text Record, Generic Control, Signatur oder andere definierte und benutzerdefinierte Daten sein. Dabei muss nicht von Anfang an klar sein, wie groß die Daten sind, die einzukapseln sind.

Eine NDEF-Mitteilung beginnt immer mit MB (Message Begin) und endet schließlich mit ME (Message End). Zwischen MB und ME werden beliebig viele Records eingefügt [NFCFORUM-TS-NDEF].

NDEF Message					
R_1 MB=1	R_2	...	R_s	...	R_n ME=1

***Abbildung 2**: Aufbau eines NDEF-Frames zum Austauschen von standardisierten Datenformaten. Eine NDEF-Message kann beliebig viele Records enthalten [NFCFORUM-TS-NDEF]*

LLCP (Logical Link Control): LLCP beschreibt, wie die verschiedenen Schichten des Protokolls aufgeteilt sind. In LLCP nicht enthalten sind Sicherungsmechanismen, Multicastfunktionalität und Timinggenauigkeit.

LLCP ist dafür verantwortlich:

- Verbindungen auf- und abzubauen und während der Kommunikation den Link aufrechtzuerhalten.
- Verbindungsorientierte bzw. auch verbindungslose Kommunikation.
- Protokoll Multiplexing: damit mehrere Protokolle höherer Ordnung gleichzeitig den Kommunikationskanal verwenden können [NFCFORUM-TS-LLCP, S4-5].

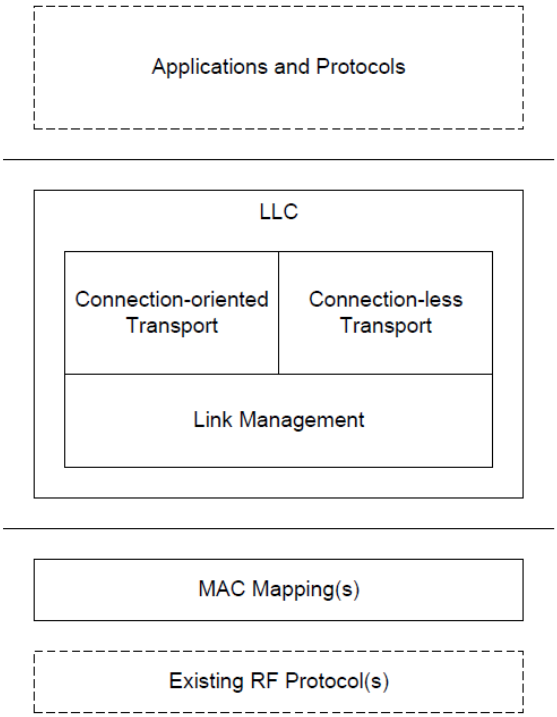


***Abbildung 3**: Aufbau der Protokollschichten. Die Schnittstelle zu den unteren Schichten kann von mehreren Applikationen oder Protokollen der oberen Schicht gleichzeitig benutzt werden [NFCFORUM-TS-LLCP, S11].*

2.2 Frequenzen und Übertragungsgeschwindigkeiten

Laut dem ECMA340-2nd Edition Standard ist die Frequenz des Trägersignals mit 13,56 MHz definiert und die magnetische Feldstärke beträgt von $H_{min,rms}$ = 1,5 A/m bis $H_{max,rms}$ = 7,5A/m. Auf diesem Feld werden die Daten für die Kommunikation moduliert. Targets sollen das Feld bei einer magnetischen Feldstärke von $H_{threshold}$ = 0,1875A/m erkennen.

Die Übertragungsgeschwindigkeit beträgt 106 kbps, 212 kbps oder 424 kbps [ECMA340].

Man findet auch Übertragungsgeschwindigkeiten bis zu 847 kbit/s. Diese sind aber proprietäre Lösungen und sind im Standard nicht enthalten.

Weitere RFID Frequenzen, wie LF (120-135kHz), 868 Mhz, 915 Mhz, 2.45GHz und UHF (5.5GHz) werden im NFC-Standard nicht berücksichtigt. Auch konzentriert sich NFC ausschließlich auf Nahfeldkommunikation bis zu 10 cm. Fernfeld bzw. Backscattersysteme sind von dieser Norm ausgeschlossen [KERN2007, S.34].

2.3 NFC-Betriebsmodi

Bei NFC unterscheidet man zwischen drei Betriebsmodi, mit denen die Kompatibilität zu älteren Technologien erhalten bleibt. Ein NFC-Gerät kann mit einem anderen NFC-Gerät kommunizieren, kann als Lesegerät arbeiten und es kann eine Smartcard emulieren. Es können nicht mehrere Modi gleichzeitig eingesetzt werden. Die Auswahl des richtigen Betriebsmodus ist im Standard NFCIP-2 (ECMA352, ISO/IEC21481) als „Mode Switch Procedure“ definiert [LANGER2010, S90-91].

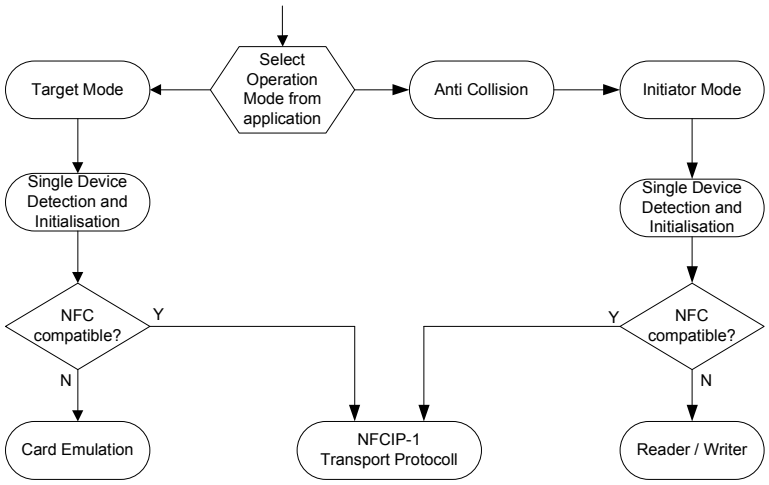


***Abbildung 4**: Vorgang zur Auswahl des Betriebsmodus eines NFC-Gerätes. Das NFCIP-1 Protokoll wird nur dann verwendet, wenn beide kommunizierenden Geräte den NFC-Standard unterstützen, sonst verhält sich ein NFC-Gerät beim Auslesen von passiven Targets identisch zu einem traditionellen Lesegerät [LANGER2010, S93].*

Damit unterscheiden wir zwischen:

- **P2P (Peer to Peer) Modus:**
  Ein NFC-Gerät ist standardmäßig immer im passiven Modus und untersucht, ob ein anderes eine Verbindung initiiert. In diesem Fall erzeugt das NFC-Gerät von sich aus kein RF-Feld. Ist kein anderer Initiator in der Nähe, so kann durch eine Applikation selbst auf Initiator geschaltet werden. Der Initiator entscheidet, ob die Kommunikation im aktiven oder passiven Modus geschehen soll, und schreibt die Übertragungsgeschwindigkeit vor. Dann wird noch überprüft, ob das Target das NFC-Protokoll unterstützt. Je nachdem wird dann der richtige Betriebsmodus ausgewählt.

    - **P2P passive Modus:** In diesem Modus stellt der NFC-Initiator die Energie für den

Datentransport zum NFC-Target und auch zum Datentransport vom NFC-Target zurück zur Verfügung. Der NFC-Initiator macht eine ASK-Modulation (Amplitude Shift Keying) auf dem hochfrequenten Trägersignal. Das NFC-Target übermittelt die Daten durch Lastenmodulation in der gleichen Geschwindigkeit wie der Initiator [ECMA340].

- **P2P active Modus:** Diese Variante ist für den Initiator energiesparender. Auf das RF-Feld, das abwechselnd vom Initiator bzw. Target erzeugt wird, werden die Daten ASK (Amplitude Shift Keying) moduliert. Die Kommunikationsparameter müssen im Initiator und im Target identisch sein. Damit die Kommunikation ohne Unterbrechungen funktioniert, darf die Feldstärke, sei dies beim NFC-Initiator, als auch beim NFC-Target, $H_{min,rms}$ = 1,5 A/m nicht unterschreiten [ECMA340].

- **Reader / Writer-Modus:** In diesem Modus abreitet das NFC-Gerät als ein Lese- bzw. Schreibgerät für herkömmliche Smartcards bzw. RF-Transponder der kompatiblen Standards ISO/IEC 14443, JIS X 6319-4 und ISO/IEC 15693 (Vicinity-Systeme). Letztere werden zwar vom Standard NFCIP-2 vorgesehen, sind aber nicht Teil der NFC-Forum-Spezifikation [LAMARCA2007, S.20].
  Während des Kommunikationsaufbaus im Antikollisionsmechanismus wird vom passiven Tag bekannt gegeben, dass kein NFC-P2P (Peer to Peer) Modus unterstützt wird und somit ein traditionelles RFID-Protokoll eingesetzt wird.
  Das NFC-Gerät ist in diesem Fall dafür verantwortlich die Energieversorgung über das HF-Feld zur Verfügung zu stellen, um die RFID-Transponder zu versorgen [LANGER2010, S99].
- **Card-Emulation-Modus:** In diesem Modus verhält sich das NFC-Gerät wie eine passive Smartcard von einem der Smartcardprotokolle ISO/IEC 14443 Typ A, ISO/IEC 14443 Typ B und JIS X 6319-4. In Verbindung mit NFC werden diese auch als NFC-A, NFC-B und NFC-F genannt.
  Da das NFC-Gerät immer einen Mikroprozessor hat, kann es auch alle Karten, von normalen Speicherkarten bis zu Mikroprozessorkarten emulieren. Es können sogar mehrere Karten gleichzeitig emuliert werden. Ein großer Vorteil dieses Modus ist, dass dieser auch ohne zusätzliche Energieversorgung funktioniert. So kann das NFC-Gerät als Kreditkarte oder für Zutrittskontrollen auch bei ausgeschaltetem Gerät funktionieren. Ein Lesegerät kann in diesem Fall den Unterschied zwischen einer Smartcard und dem NFC-Gerät im Card-Emulations-Modus nicht unterscheiden [LANGER2010, S100,101] [IEEEMAN2008].

Zur Übertragung von Daten werden Vollduplex (FDX), Halbduplex (HDX) und sequenzielle (SEQ) Betriebsarten verwendet.

Bei **FDX (Vollduplex)** wird das Feld am Lesegerät nie abgeschaltet. Es können zugleich Daten vom Lesegerät zum Transponder und vom Transponder zum Lesegerät strömen.

Bei **HDX (Halbduplex)** wird vom Lesegerät abwechselnd geschrieben und gelesen, wobei das HF-Feld immer für die Energieversorgung der Transponder präsent ist. Die Transponder verwenden Lastenmodulation zur Datenübertragung.

Bei **SEQ (sequenziell)** wird die Energie im Transponder zwischengespeichert. Sobald das Lesegerät das Feld abschaltet, verwendet der Transponder die begrenzte Energie, um die Daten aktiv zu übertragen. Nachteil ist hier die geringe Autonomie und die kompliziertere Bauweise der Transponder mit Zwischenspeicher [KERN2007, S.59-61].

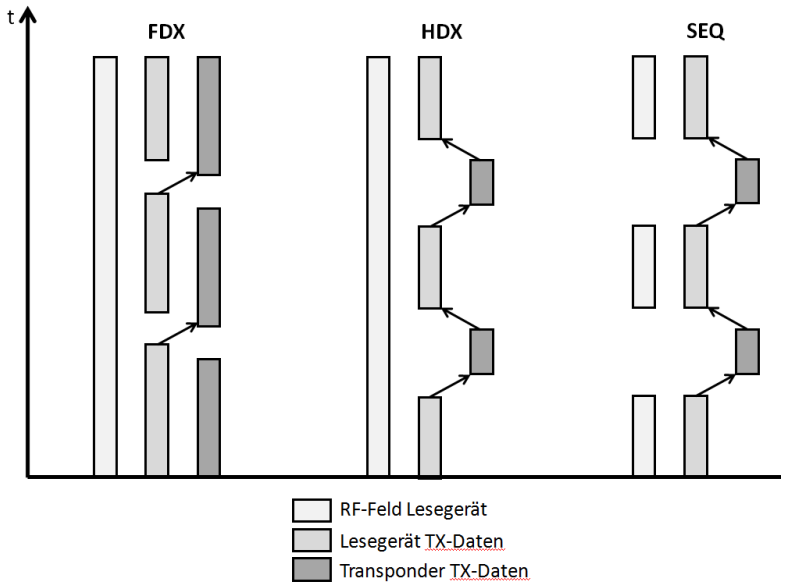


***Abbildung 5**: Energiezufuhr und Datenstrombetriebsarten bei RFID-Systemen und bei NFC-Geräten im Lesegerätmodus oder Target-Emulationsmodus [Kern, S60].*

2.4 Antikollisionsmechanismen und Initialisierung

Es werden in RFID und NFC Systemen verschiedene Antikollisionsmechanismen eingesetzt. Die Idee ist, dass jeder Transponder den Datenkanal nur so lange reserviert, bis alles abgewickelt ist. Dabei ist für den Transponder irrelevant, wie viele andere Transponder im Ansprechfeld sind. Sie bemerken die Präsenz anderer nicht und können auch die Daten nicht mitlesen, die von anderen Transpondern zum Lesegerät gesendet werden.

TDMA (Time Domain Multiple Access) ist jener Antikollisionsmechanismus, der in RFID bzw. NFC-Systemen am meisten eingesetzt wird. Dabei wird vom Initiator (Lesegerät) jeweils ein Transponder selektiert und so die Kommunikation aufrechterhalten, bis die Authentifizierung und alle Lese- und Schreibvorgänge abgewickelt sind. Nachdem wird der Transponder wieder freigegeben und mit dem nächsten die Kommunikation aufgenommen. Ein Beispiel dafür ist das ALOHA Verfahren. Hier senden die Transponder sobald die Daten zur Verfügung stehen und wiederholen dies nach einer bestimmten Zeit mit etwas Abweichung. Damit entstehen viele Kollisionen. Dieses Ver-

fahren funktioniert ausschließlich mit Readonly-Transponder, welche wenig Daten zum Übertragen haben. Besser funktioniert das Slotted ALOHA Verfahren. Hier werden Zeitfenster zur Verfügung gestellt, in denen die Transponder sich mit einer eindeutigen Seriennummer identifizieren können. Hierbei können Kollisionen passieren, die allerdings erkannt werden. Sobald eine eindeutige Seriennummer bekannt ist, kann diese selektiert werden, womit anschließend beliebig viele Lese- und Schreibzyklen durchgeführt werden können. Daraufhin wird der Transponder wieder freigegeben und erneut ein REQUEST-Broadcast zum Einlesen der Seriennummern ausgesendet [FINKENZELLER2002, S.213-S.215].

Ob Kollisionen erkannt werden, ist von der Codierung im Basisband abhängig. Bei einer NRZ (Non Return to Zero) ist es nicht möglich Kollisionen zu erkennen, während bei einer Manchestercodierung bereits Kollisionen der einzelnen Bits erkannt werden können. Es ist aber wichtig, dass alle Transponder synchronisiert und mit der gleichen Übertragungsgeschwindigkeit zum Senden anfangen.

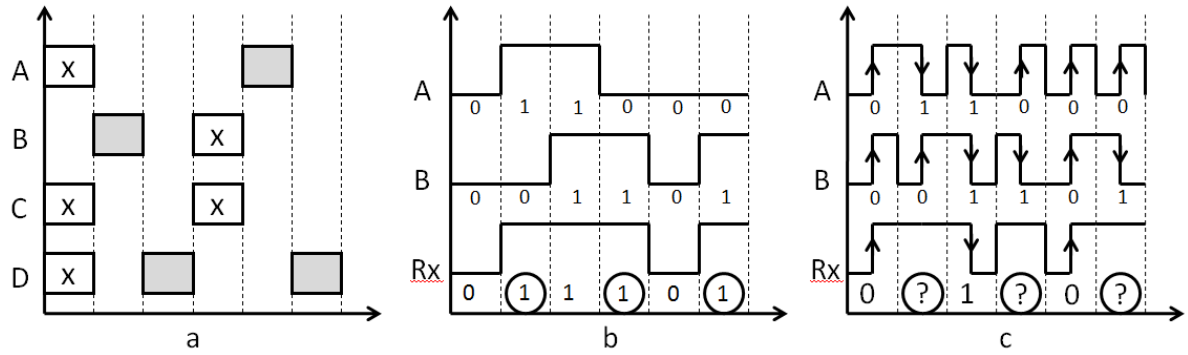


***Abbildung 6**: Kollisionserkennung im Basisband.*
a) Slotted Aloha Verfahren: Fixe Zeitfenster werden den Transpondern zur Verfügung gestellt, um sich zu identifizieren.
b) NRZ (Non Return to Zero): Kollisionen können nicht erkannt und einzelne Bits fehlinterpretiert werden.
c) Manchestercodierung: Die Kollisionen einzelner Bits werden erkannt [finkenzeller2010, S205]

Es wird auch oft ein binärer Suchbaum eingesetzt, um einzelne Transponder zu finden. Gestartet wird mit dem vollen möglichen Adressbereich (Broadcasting). Alle Transponder im Lesebereich antworten mit der eindeutigen Seriennummer. Antwortet mehr als ein Transponder, so wird die Kollision bei einem spezifischen Bit erkannt. Ein neuer Adressbereich wird abgefragt. Dabei wird der MSB (Most Significant Bit), der in der Vorrunde kollidierte, als ‚1' oder als ‚0' fixiert. Dies geschieht solange, bis eine eindeutige Seriennummer ohne Kollisionen richtig ausgelesen wurde. Die durchschnittliche Anzahl an Iterationen um eine eindeutige Seriennummer zu finden ist damit

$$L(N) = \log_2(N) + 1 . \qquad 2.1$$

N ist die Anzahl der Transponder im Lesebereich [FINKENZELLER2010].

Eine weitere Methode ist SDMA (Raummultiplexing). Dabei werden die Transponder so bewegt, dass sich immer nur einer im Ansprechfeld befindet. Möglich ist es auch, mit einer Richtantenne immer nur einen Transponder auf einmal mit einem Feld über dem Schwellwert für die Energiezufuhr zu bestrahlen. Wegen der speziellen Anforderungen der Positionierung der Transponder ist diese Methode nur bei wenigen Anwendungen einsetzbar.

FDMA (Frequency Domain Multiple Access) verwendet eine Frequenz, die ideal für die Energieversorgung ist. Die Transponder antworten auf verschiedenen zur Verfügung gestellten Kanälen. Für die Lastenmodulation lässt sich für jeden Transponder eine andere Hilfsträgerfrequenz verwenden. Auch hier ist der größere Aufwand bei der Erstellung der Hardware des Lesegeräts (Empfänger für jeden Kanal) der springende Punkt, warum dieser Antikollisionsmechanismus nur in Spezialanwendungen verwendet wird [FINKENZELLER2002, S.203-S.210].

2.5 Energieversorgung des Transponders

Die Targets sind meist passive Elemente und entnehmen die gesamte Energie, um den internen Controller zu betreiben aus dem externen magnetischen Feld, welches vom Lesegerät erzeugt wird.

Die Energieversorgung der Targets basiert auf dem Prinzip eines Transformators. Am Lesegerät wird ein magnetisches Feld durch eine Antenne aufgebaut und am Target wird die Energie über eine Antenne gewonnen.

Die Induktivität einer Leiterschleife mit N Windungen ist

$$L = \frac{N \cdot \Phi}{I} = \frac{N \cdot \mu \cdot H \cdot A}{I} = N^2 \cdot \mu_0 \cdot R \cdot \ln\left(\frac{2 \cdot R}{d}\right). \qquad \text{[FINKENZELLER2010]} \qquad 2.2$$

Befindet sich in der Nähe der stromdurchflossenen Lesegerätantenne die Antenne eines Transponders, so wird diese von einem Teil des erzeugten magnetischen Feldes durchsetzt. Damit werden die beiden Stromkreise miteinander gekoppelt. Wie stark diese miteinander gekoppelt sind, hängt vom geometrischen Aufbau sowie von der Entfernung und der Anordnung der Antennen zueinander ab. Diese Größe wird durch die Gegeninduktivität ausgedrückt, welche sich durch

$$M_{21} = M_{12} = \frac{\Psi_{A2}(I_1)}{I_1} = \frac{\mu_0 \cdot H(I_1) \cdot N_2 \cdot A_2}{I_1} \qquad \text{[FINKENZELLER2010]} \qquad 2.3$$

berechnen lässt.

Aus der Gegeninduktivität und Eigeninduktivität der beiden Antennen wird der Kopplungsfaktor

$$k = \frac{M_{21}}{\sqrt{L_1 \cdot L_2}} \qquad \text{[FINKENZELLER2010]} \qquad 2.4$$

bestimmt. Dabei ist k immer zwischen 0 und 1. Bei $k = 0$ sind die zwei Antennen möglicherweise durch große Distanz oder Abschirmung komplett entkoppelt. Bei $k = 1$ fließt der gesamte magnetische Fluss durch die sekundäre Antenne und man spricht dabei von einer totalen Kopplung.

Durch die Änderung des magnetischen Flusses in der Transponderantenne wird eine Spannung induziert. Bei einem sehr niedrigen Kopplungsfaktor ist die induzierte Spannung nicht ausreichend, um den Transponder-Chip zu versorgen. Der Wirkungsgrad wird hier durch einen Resonanzschwingkreis maximiert. Parallel zur Transponderantenne (Spule + Widerstand) wird ein Kondensator geschaltet. Dieser wird so abgestimmt, dass die Resonanzfrequenz der Betriebsfrequenz des RFID-Lesegerätes entspricht:

$$C_p = \frac{1}{4 \cdot \pi^2 \cdot f^2 \cdot L_2}. \qquad 2.5$$

Eine Ersatzschaltung des Gesamtsystems wird in Abbildung 7 dargestellt.

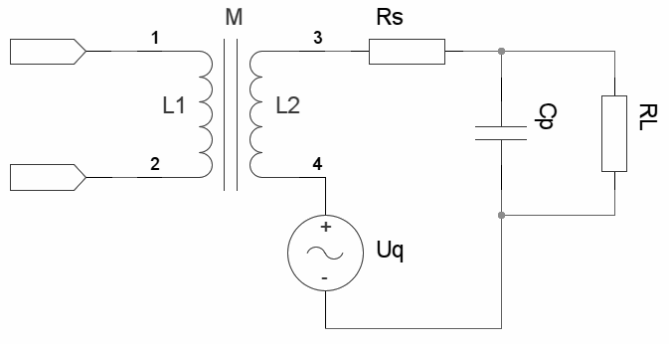


***Abbildung 7**: Ersatzschaltung Gesamtsystem Lesegerät und Target*

Damit ergibt sich für die Versorgungsspannung des Target-Chips

$$u(R_L) = \frac{j\omega \cdot k \cdot i_{L1} \cdot \sqrt{L_1 \cdot L_2}}{1 + (j\omega \cdot L_2 + R_2) \cdot \left(j\omega \cdot C_p + \frac{1}{R_L}\right)}. \qquad \text{[FINKENZELLER2010]} \qquad 2.6$$

Je nach Kopplungsfaktor kann diese Spannung auch einige hundert Volt betragen. Damit der Transponder-Chip nicht durch Überspannungen beschädigt wird, wird parallel zu R_L ein spannungsabhängiger Widerstand (Varistor) geschaltet. Damit ist eine stabile Spannungsversorgung des Transponders sichergestellt [SCHOBLICK05] [FINKENZELLER2010].

2.6 Transponder Modulationsarten und Demodulation im Lesegerät

Befindet sich ein Transponder im Nahfeld des Lesegerätes, so haben wir eine Transformatorkopplung zwischen den beiden Antennen, wie es unter Punkt 2.5 beschrieben ist. Der Resonanzschwingkreis und die Last im Target lassen sich durch die Rückwirkung der Kopplung mit dem Lesegerät als transformierte Impedanz Z_T darstellen. Durch das Zu- und Wegschalten eines Widerstandes zu R_L im Target variiert auch die transformierte Impedanz Z_T. Auf die Spannung an der Antenne im Lesegerät wirkt sich das als eine Amplitudenmodulation aus.

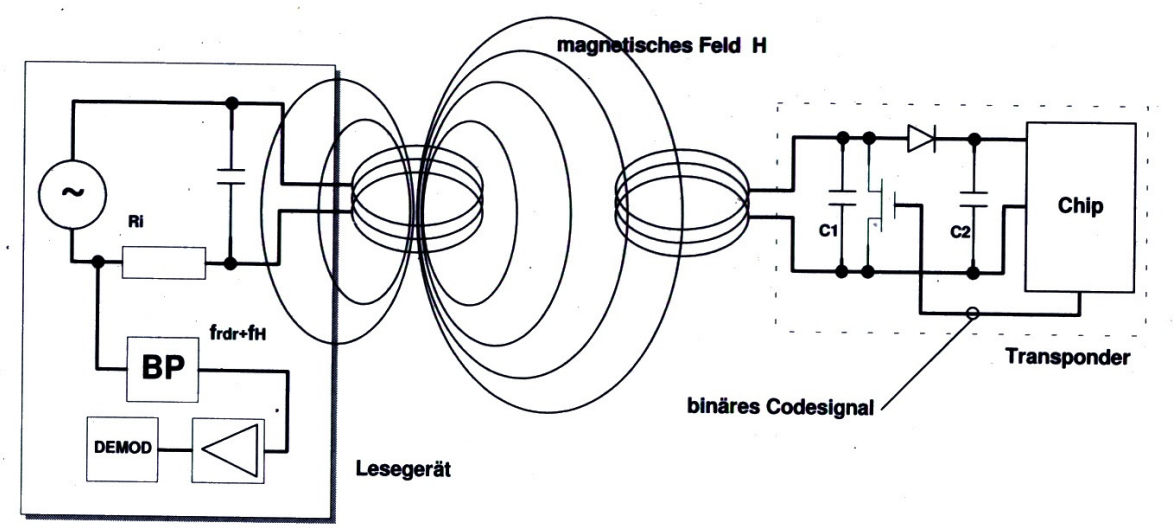


***Abbildung 8**: Demodulation im Lesegerät. Die Lastenmodulation wirkt sich im Lesegerät als amplitudenmoduliertes Signal aus [FINKENZELLER2010].*

Das Verhältnis zwischen Trägersignal und Nutzsignal des amplitudenmodulierten Signals kann bei kleinem Kopplungsfaktor sehr groß sein und ist somit nicht einfach zu detektieren. Zur leichteren Detektion wird die Lastenmodulation mit einem Hilfsträger durchgeführt. Die Frequenz des Hilfsträgers muss dabei größer als die Datenrate und kleiner als die Lesegerätfrequenz sein. Bei einem 13 MHz RFID-System werden die Daten typischerweise auf einem 212 kHz-Hilfsträger moduliert. Im Lesegerät kann dann durch einen Bandpassfilter leicht auf die Nutzdaten geschlossen werden [FINKENZELLER2010] [DOBKIN08].

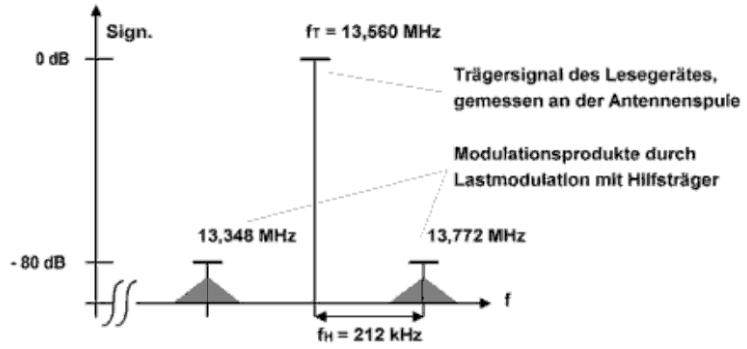


***Abbildung 9**: Lastenmodulation auf Hilfsträger. Der Störabstand im Lesegerät wird deutlich verbessert [FINKENZELLER2010].*

3. Problemanalyse und Lösungsvorschlag

In diesem Kapitel folgt eine Konzeptausarbeitung eines synchronisierten Schlosssystems, wobei die Schlösser in einer entlegenen Ortschaft angenommen werden. Dabei sind vor allem Maßnahmen für die Energieeffizienz und Sicherheit zu treffen, welche die Realisierung eines solchen Systems überhaupt ermöglichen. So ein System ist nur dann realisierbar, wenn die Betriebskosten durch Batterien tragbar sind und die Sicherheit der Schlüssel und Schlösser garantiert ist. Als Beispielanwendung wird hier speziell auf ein Fahrradschlosssystem im öffentlichen Interesse eingegangen. Vom Prinzip her kann das Schlosssystem aber auch für Hotelzimmer oder Bürogebäude verwendet werden.

3.1 Anforderungen an das Gesamtsystem

Die Verwaltung einer Stadt könnte sichere Schlösser für Fahrräder bereitstellen. Diese würden sehr robust gebaut werden und würden auf den bestehenden Fahrradständern fixiert werden. Jede Person, die nun so ein Schloss verwenden möchte, braucht einen Schlüssel. Jeder Benutzer kann ein Schloss schließen, aber nur jener Schlüssel, der ein Schloss versperrt hat, kann dieses auch wieder öffnen. Mit traditionellen Schlössern ist dies nicht realisierbar. Um das Problem mit der Schlüsselvergabe zu lösen, eignet sich die NFC-Technologie. Damit können Bürger eine Smartcard oder auch das eigene Handy (NFC Unterstützung vorausgesetzt) verwenden, um ein Schloss zu betätigen. Nur jene Karte oder jenes Handy, die ein Schloss verriegelt haben, haben auch die Berechtigung dieses wieder zu öffnen. Beim Benützen des Handys gibt es die Möglichkeit, den Schlüssel durch eine Applikation an einen Freund weiterzuschicken.

Ein **Schlossbenutzer** braucht folgende Funktionalität:

- Schloss verriegeln
- Schloss öffnen
- Schlüssel an andere Person weitergeben
- einmalige Registrierung und Aktivierung eines Accounts
- Neuen Schlüssel anfordern (im Fall von Verlust bzw. Zerstörung)

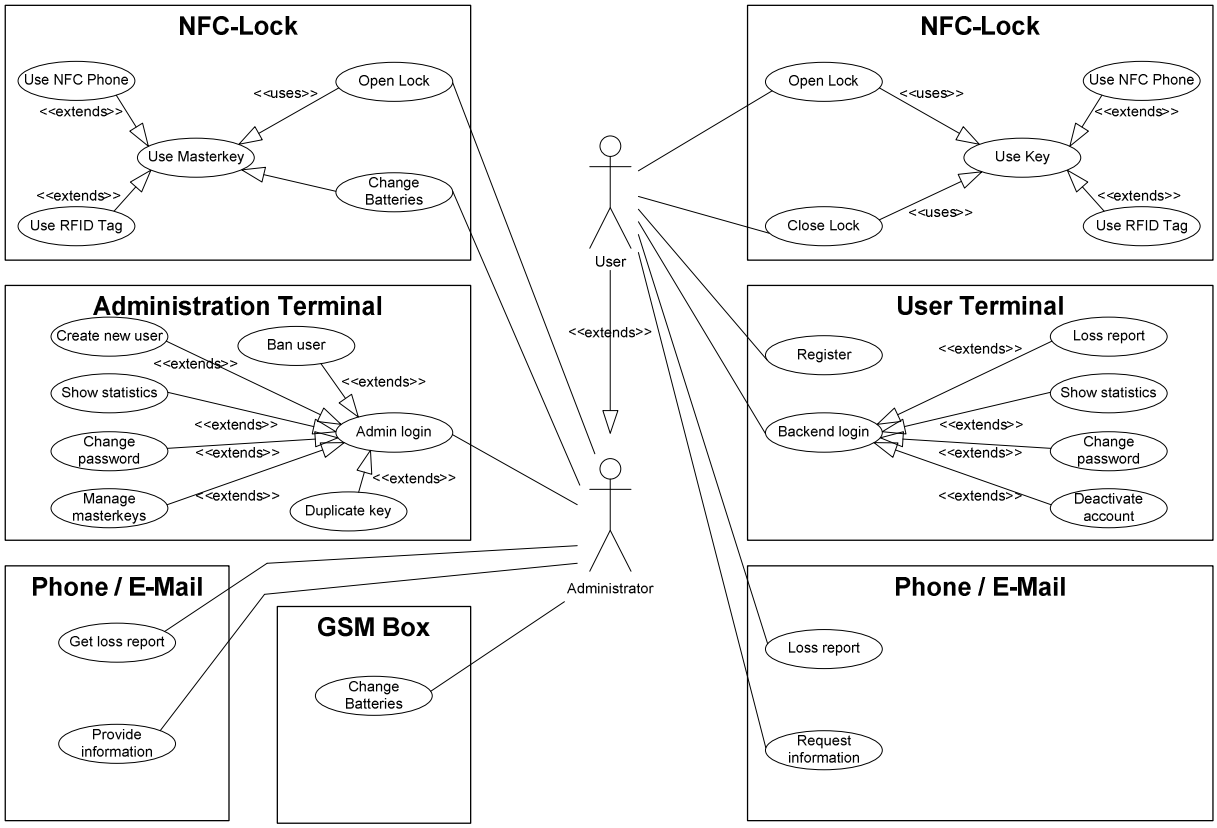


***Abbildung 10**: Use Case Diagramm aus der Sicht des Schlosssystembenutzers und aus der Sicht des Administrators.*

Ein **Administrator** wird mit den folgenden Aufgaben konfrontiert:

- neuen Benutzer registrieren
- gegebenenfalls einen Benutzer sperren (im Fall von Missbrauch)
- Generalschlüssel (darf ein Schloss immer öffnen)
- Batterien austauschen
- Karte duplizieren im Fall von Verlust

Vom **Gesamtsystem** wird also verlangt:

- nur registrierte Karten bzw. Handys können die Schlösser verriegeln und öffnen
- ein Generalschlüssel kann ein Schloss immer öffnen
- Datenbank mit gesperrten Benutzern muss auf allen Schlössern täglich aktualisiert werden
- bei niedriger Batteriestufe einer Schlosseinheit wird dem Administrator eine Meldung geschickt
- bei einer Fehlfunktion wir ebenfalls eine Meldung an den Benutzer und an den Administrator weitergeleitet
- elektronisch muss alles so energieeffizient sein, dass es mit einem relativ kleinen Batteriepack für mehrere Jahre betrieben werden kann
- Kälte, Nässe und Schmutz dürfen die Funktionalität nicht beeinflussen
- mechanische Belastbarkeit gegen Diebe
- Sabotagemöglichkeiten sollten ausgeschlossen werden. Auch wenn der Angreifer weiß, wie das System gebaut ist, darf keine Möglichkeit bestehen, dieses zu manipulieren.

Um diese Systemmerkmale zu gewährleisten, ist eine Vernetzung aller Schlösser mit einer Zentraleinheit unvermeidlich. Die Schlüsselsynchronisation mit der Zentraleinheit ist nur über das Handynetz realisierbar, da die Schlösser außer Reichweite von Strom- und Datenkabel sind. Da an einem Fahrradständer immer mehrere Schlösser montiert werden, ist es sinnvoll, nur eine Box am GSM-Netz anzuschließen und die anderen mit einer energiesparenden Funktechnik mit der GSM-Box zu synchronisieren. Über das GSM-Netz kann man dann die Datenbank über eine passwortgeschützte FTP-Verbindung synchronisieren. Damit nicht immer alle gesperrten Schlüssel neu übertragen werden müssen, werden nur jene aktualisiert, die seit dem letzten Update hinzugefügt oder entfernt wurden. Die Synchronisation wird im Punkt 3.4 detailliert beschrieben.

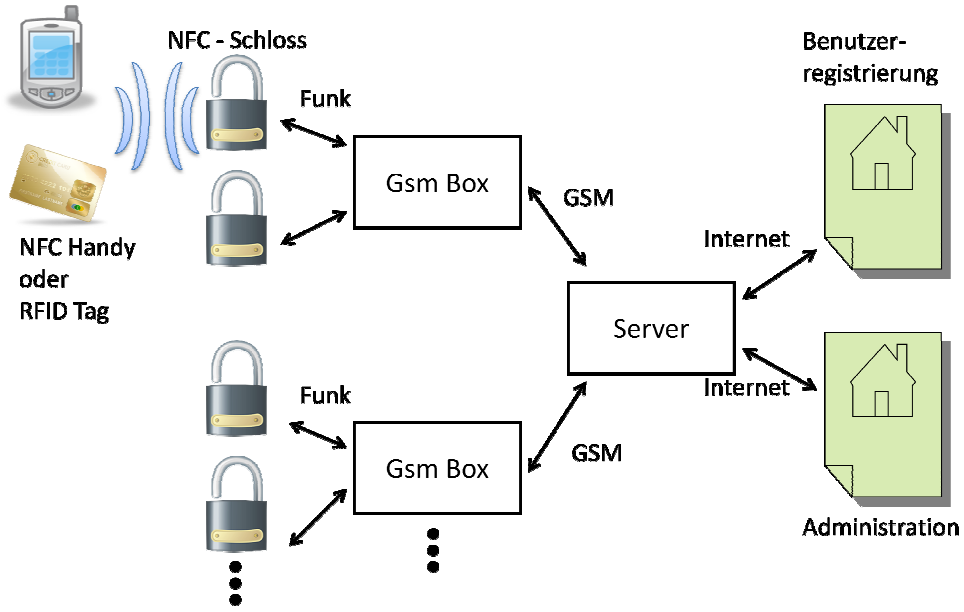


***Abbildung 11**: Lösungsvorschlag eines Fahrradschlosssystems zum Öffnen und Schließen mittels NFC aktiviertem Gerät oder mit einer kontaktlosen Chipkarte. Die Schlüssel müssen immer über die GSM-Box auf den Schlössern synchronisiert werden. Über die Internetseiten können Änderungen der Konfiguration der Schlösser vorgenommen werden.*

Damit können wir das Gesamtsystem in die folgenden Module unterteilen, welche unter Punkt 3.3 näher untersucht werden:

- elektronisches NFC Schloss
- GSM-Box
- Zentralserver
- Administrationsterminal
- Registrierungsterminal

3.2 Benutzerinterface

In diesem Kapitel wird die Benutzung des Schlosssystems aus der Sicht des Benutzers analysiert. Dabei wird die Usability in den Vordergrund gestellt. Die Benützung sollte intuitiv und ohne Schulung der Benutzer möglich sein. Außerdem werden verschiedene organisatorische Abläufe ausgearbeitet, womit ein komplettes System mit den gesamten erforderlichen Dienstleistungen entsteht.

3.2.1 Öffnen und Schließen eines NFC-Schlosses

Das Öffnen und Schließen eines NFC Schlosses ist jene Dienstleistung, welche ein Benutzer vom System am öftesten bezieht. Deshalb sollte diese Aufgabe so einfach, schnell und zuverlässig wie möglich durchgeführt werden. Dabei kann immer nur jener Schlüssel ein Schloss öffnen, der es auch versperrt hat.

Beim Zusperren eines Schlosses wird der Benutzer zuerst das eigene Fahrrad am richtigen Ort platzieren, so wie er das schon von traditionellen Fahrradständern gewohnt ist. Anschließend gibt es zwei Möglichkeiten. Bei einer Variante wird der Nutzer daraufhin den Schlüssel betätigen, das Schloss schließen und somit dieses verriegeln. Außerdem kann der Nutzer zuerst das Schloss schließen und dann erst den Schlüssel zum Verriegeln benutzen. Beide Fälle müssen berücksichtigt werden. Benützt der Benutzer die Karte als Schlüssel bereits vor dem Schließen, wartet das Schloss eine bestimmte Zeit auf das Schließen des Schlosses. Diese Zeit sollte so lang sein, dass der Benutzer nicht unter Zeitdruck kommt, das Schloss zu schließen. Sollte es der Fall sein, dass das Schloss von diesem Benutzer trotzdem nicht benützt wird, wird ein Time-out ausgelöst. Damit soll garantiert werden, dass ein Fahrrad nicht mit einem fremden Schlüssel abgesperrt wird und danach mit dem eigenen Schlüssel nicht mehr entriegelt werden kann. So ist es sinnvoll, dieses Zeitlimit bei circa zehn Sekunden anzusetzen.

Beim Öffnen des Schlosses gibt es nur ein Szenario. Der Benutzer öffnet mit dem Schlüssel das Schloss. Es muss berücksichtigt werden, dass sich das Schloss nicht öffnen darf, wenn es unmittelbar zuvor geschlossen wurde. Auch hier ist es sinnvoll, nach dem Abschließen einige Sekunden Wartezeit zu gewähren, bis sich der Schlüssel außerhalb des Lesebereichs des Schlosses befindet. Zehn Sekunden sind auch hier gewiss ein gutes Maß.

Damit der Benutzer weiß, in welchem Zustand sich das Schloss gerade befindet, kann ein Signalton oder eine LED-Leuchte benützt werden. Ein Signalton ist für Menschen mit Hörbeeinträchtigung ungeeignet. Die Wahrscheinlichkeit, dass Menschen mit einer schweren Sehbehinderung Fahrradfahren, ist sehr gering. Damit wird die LED-Leuchte bevorzugt.

Mit einer LED-Leuchte kann durch verschiedene Blinkfrequenzen und Farben die Information so codiert werden, dass sie intuitiv vom Benutzer aufgenommen wird. Die LED-Signalisierung wird so lange angezeigt, bis die Wartezeit, die zwischen dem Öffnen und Schließen des Schlosses abgewartet wird, abgelaufen ist.

- Schloss wurde zugesperrt (ROT, kein Blinken)
- Schloss wartet auf eine Benutzeraktion, z. B. schließen des Schlosses (GRÜN, langsames Blinken)
- Schloss wurde geöffnet (GRÜN, kein Blinken)
- Zutritt nicht gewährleistet (ROT, schnelles Blinken)

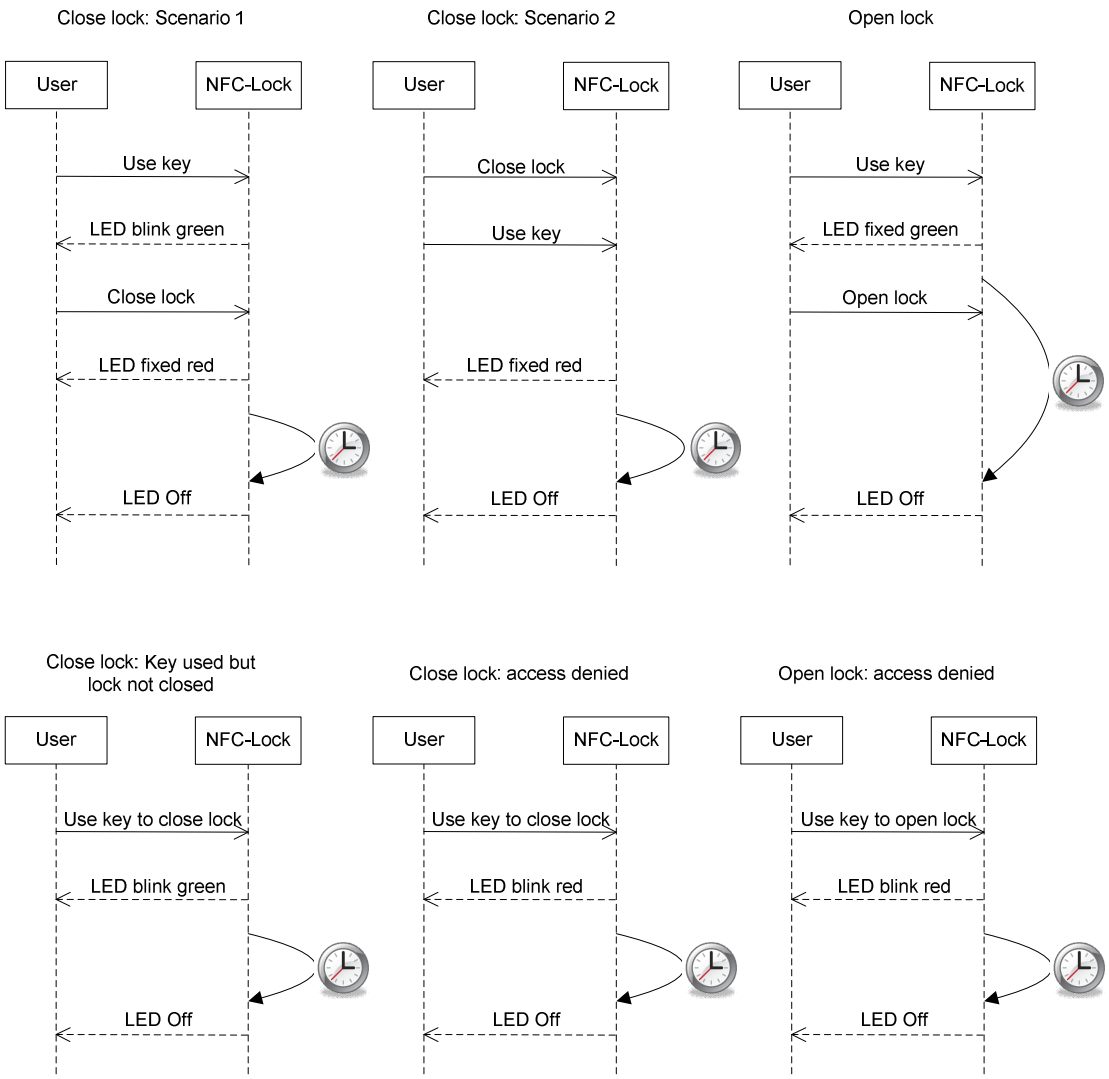


Abbildung 12: *Sequenzdiagramm verschiedener Szenarien der Interaktion zwischen Schlossbenutzer und Schloss. Das Schloss kommuniziert mit dem Benutzer über eine LED-Leuchte bezüglich des Öffnens und Schließens des Schlosses. Die Codierung mit Farben und Blinken wurde so gewählt, dass der Benutzer die Mitteilung intuitiv wahrnimmt.*

3.2.2 Registrierung, Kontoverwaltung und Abmeldung

Um einen Missbrauch vorzubeugen, darf ein Benutzer erst nach einer einmaligen Registrierung das Schlosssystem benützen. Dabei muss er die persönlichen Daten angeben. Es werden Name, Nachname, Adresse, Telefonnummer, E-Mail-Adresse und ein Password benötigt. Bei Problemen jeglicher Art ist somit eine Kontaktaufnahme zu jedem Benutzer möglich. Danach bekommt dieser eine

Smartcard oder eine Smartcard-NFC-Datei für das eigene Handy, die zum Beispiel in „Google Wallet“ gespeichert werden kann.

Ein Schlüssel kann somit:

- vom Benutzer über eine Internetseite bestellt und dann per Post zugeschickt werden
- bei einer Servicestelle ausgestellt werden
- am Handy durch eine Applikation heruntergeladen werden

Der erhaltene Schlüssel ist sofort auf allen Schlossstationen einsetzbar.

Nach der einmaligen Registrierung wird jedem Benutzer auf einer Internetseite die Möglichkeit angeboten, sich einzuloggen. Die E-Mail-Adresse und das ausgewählte Password gelten als Anmeldedaten. Hier werden die eigenen Daten und eine Statistik über die Schlossbenutzung angezeigt. Die Adresse, Telefonnummer, E-Mail-Adresse und Passwort kann auch jederzeit geändert werden.

Möchte ein Benutzer das Schlosssystem nicht mehr benützen, so steht ihm die Möglichkeit offen, das eigene Konto zu deaktivieren. Eine weitere Möglichkeit ist es das Schlosssystem einfach nicht mehr zu benützen, und den Schlüssel zu entsorgen.

3.2.3 Schlüsselverlustmanagement

Im Fall von Schlüsselverlust muss kompetent und schnell gehandelt werden. Dabei sind zwei Fälle zu unterscheiden:

- Der Schlossbenützer verliert den Schlüssel und hat gerade kein Fahrrad abgeschlossen
- Der Benützer hat das Fahrrad an einer Station abgesperrt und verliert dann den Schlüssel

In beiden Fällen ist eine Sperre des Schlüssels notwendig und ein neuer wird erstellt.

Im ersten Fall kann es passieren, dass ein unbefugter Benutzer den verlohrenen Schlüssel findet und das Schlosssystem mit dem eigenen Fahrrad benutzt. Was zwar unerwünscht aber nicht weiter schlimm ist.

Im zweiten Fall hingegen kann nur gehofft werden, dass der Finder des Schlüssels nicht weiß, wo das Fahrrad abgestellt ist. Zusätzlich zur Sperre des verlorenen Schlüssels, muss hier das richtige Schloss für den neuen Schlüssel zum Aufsperren zugänglich gemacht werden. So kann der Benutzer dann nach dem nächsten Update das Fahrrad einfach wieder abholen.

Zum Sperren eines Schlüssels reicht ein Telefonat mit der Zentrale. Hier werden mündlich Daten abgefragt, wie Name, Nachname, Geburtsdatum und Adresse. Die neue Karte wird per Post an die in der Datenbank gespeicherte Adresse geschickt. So kann sichergestellt werden, dass keine falsche Person oder ein Betrüger den neuen Schlüssel erhält.

Beim Sperren eines Schlüssels an einer Servicestelle muss die Richtigkeit der Daten durch einen Personalausweis überprüft werden. In diesem Fall kann der neue Schlüssel sofort ausgehändigt werden. Der neu ausgestellte Schlüssel ist sofort funktionsfähig. Ein mit dem vermissten Schlüssel abgeschlossenes Fahrrad kann erst am nächsten Tag nach dem Update abgeholt werden.

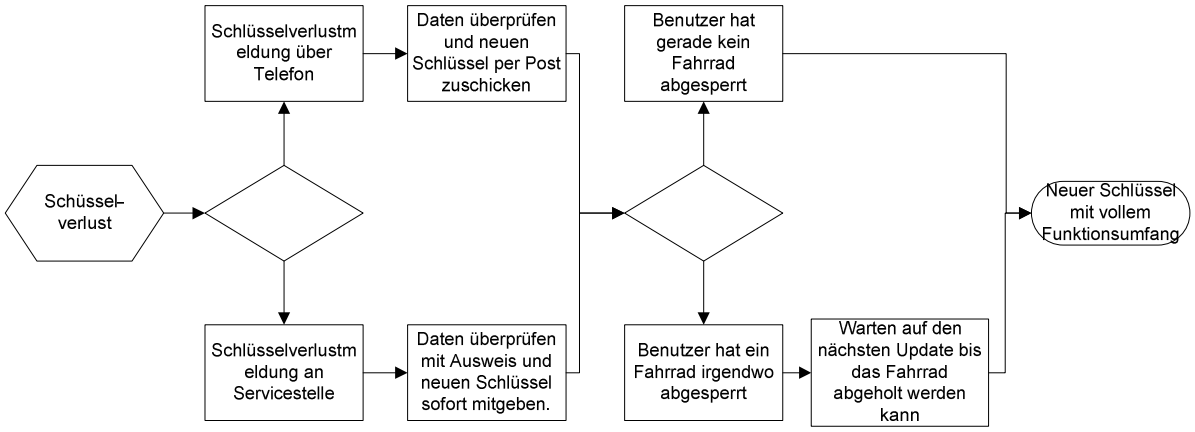


***Abbildung 13**: Schlüsselverlustmanagement. Der organisatorische Ablauf, wie der Administrator den Schlüsselverlust eines Schlossbenutzers handhaben soll.*

3.3 Systemkomponenten

In diesem Kapitel wird die Funktionalität der einzelnen Systemkomponenten beschrieben. Dabei wurden die verschiedenen Abläufe so optimiert, dass in den batteriebetriebenen Komponenten nur die notwendigsten Aktionen durchgeführt werden und alle möglichen Schritte und Berechnungen auf die Zentraleinheit ausgelagert werden.

3.3.1 Schloss

In diesem Unterkapitel wird genauer auf die Systemkomponenten eingegangen, die mit dem Benutzer am öftesten in Kontakt kommt. Es werden die Abläufe ausgearbeitet, welche zum Öffnen bzw. Schließen des Schlosses notwendig sind. Welche Maßnahmen getroffen werden, um am meisten Energie in dieser Komponente zu sparen, wird in Kapitel **Error! Reference source not found.** behandelt.

Diese Komponente muss die Funktion, wie in Kapitel 3.2.1 beschrieben, bereitstellen. Somit werden folgende Komponenten gebraucht: Mikrocontroller, NFC-Kommunikationsteil, elektrischer Riegel, ein Speicher und ein lokales Funksystem.

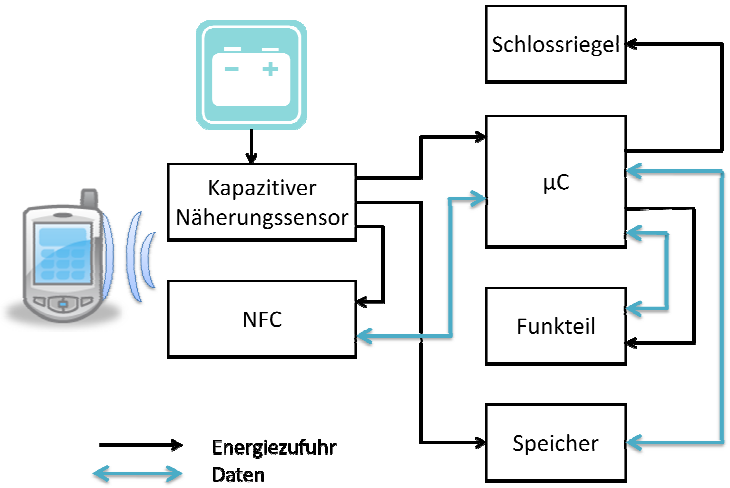


***Abbildung 14**: Architektur des NFC-Schlosses. Die Bauteile werden über den Näherungssensor mit Strom versorgt.*

Der allgemeine Ablauf funktioniert so, dass der Mikrocontroller versucht über die NFC-Komponente ein Target im Lesebereich zu finden und dessen ID auszulesen. Sobald eine ID ausgelesen wurde, wird diese mit den Sicherheitsmaßnahmen, wie im Kapitel 0 beschrieben, authentifiziert. Nach einer sicheren Authentifizierung muss sichergestellt werden, dass der Schlüssel nicht gesperrt ist. Damit muss der interne Speicher nach der aktuellen CardID durchsucht werden. Ist diese CardID im Speicher nicht enthalten, so kann das Schloss benutzt werden. Wenn das Schloss gerade offen ist, so wird es abgesperrt, ist es abgesperrt, so wird es geöffnet.

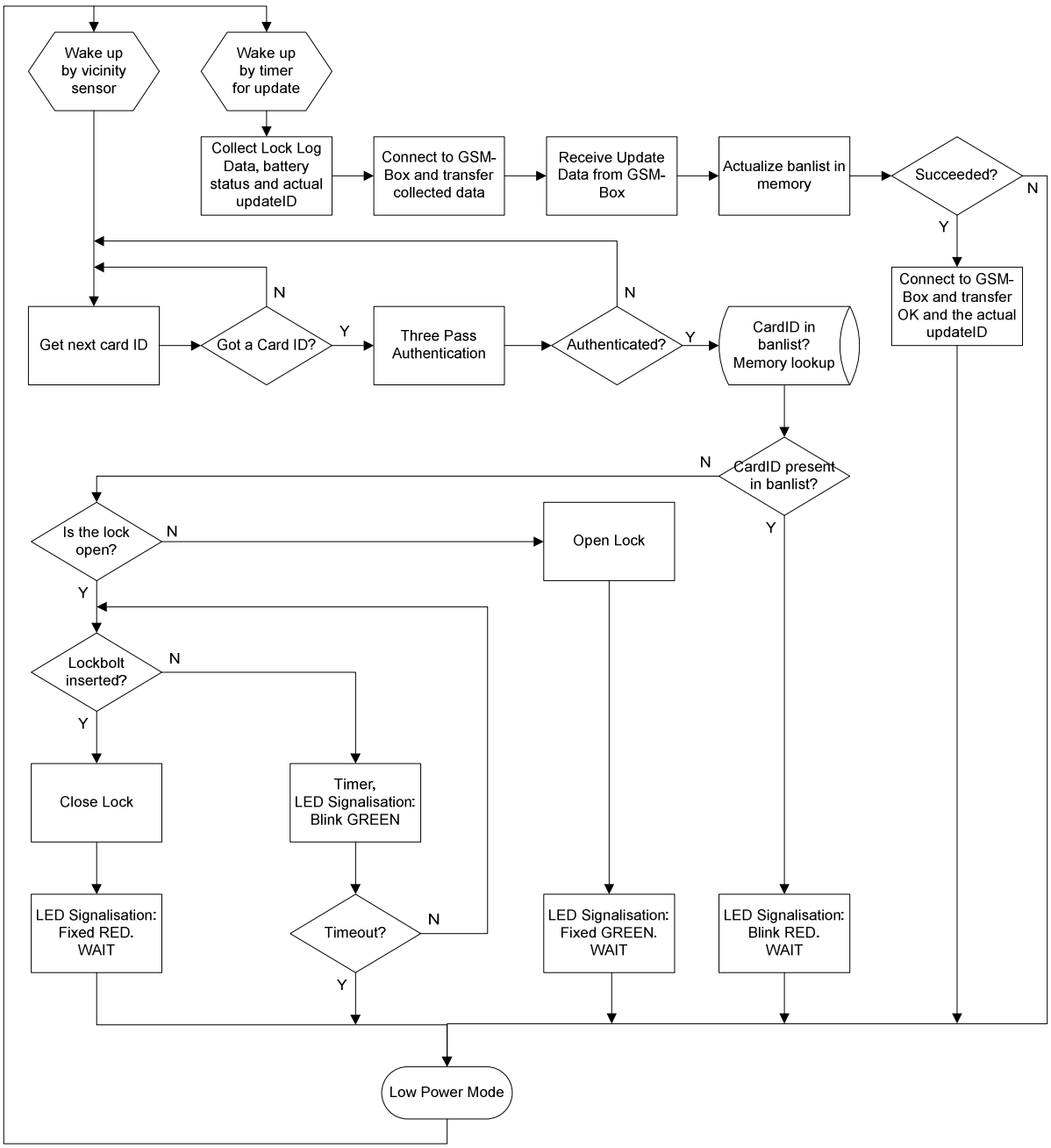


***Abbildung 15**: Ablaufdiagramm des Verhaltens der Software im NFC-Schloss zum sicheren Öffnen und Schließen sowie Durchführen von Updates und Mitteilen von Schlossbenutzungslogs bzw. des Schlossstatus.*

Um den Speicher nach einer ID zu durchsuchen, brauchen wir einen Algorithmus, der so schnell wie möglich aus dem Array mit tausenden Schlüsseln, den richtigen findet. Mit einem binären Suchbaum:

```
min := 1;
max := N; {array size: var A : array [1..N] of integer}
repeat
  mid := (min+max) div 2;
  if x > A[mid] then
    min := mid + 1;
  else
    max := mid - 1;
until (A[mid] = x) or (min > max);
```

kann man eine sortierte Liste auf einen Eintrag überprüfen. Wenn N die Anzahl der Einträge im Speicher ist, dann braucht man maximal $\log_2(N)$ Abfragen, bis man ein Resultat hat. Das ergibt bei 10^6 Schlüsseln aufgerundet 20 (19,931) IF Abfragen auf dem Speicher. Dies wäre für einen Mikrocontroller leicht verträglich.

Ein Mal pro Tag wacht das Schloss auf und sendet gesammelte Informationen über die Schlossbenutzung (CardID, Uhrzeit, aufgesperrt / zugesperrt), Batteriestatus und Fehlermeldungen zur GSM-Box weiter.

3.3.2 GSM-Box

Die GSM-Box hat vor allem eine Gatewayfunktion. Sie leitet die Daten aller NFC-Schlösser einer Parkstation zur Zentraleinheit weiter. Die Daten werden einmal pro Tag von den Schlosseinheiten über Funk empfangen, zusammengefügt und über GSM weitergeschickt. Von der Zentraleinheit werden diese Daten aufbereitet und ein Update wird zurückgeschickt, welches somit auf die einzelnen Schlösser übertragen wird.

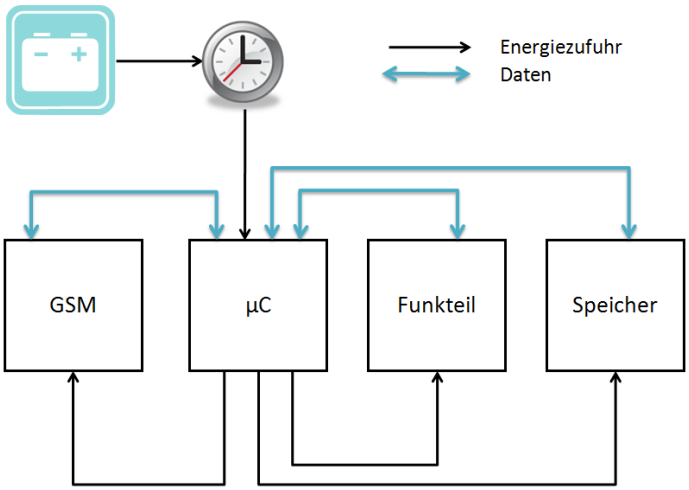


***Abbildung 16**: Architektur GSM-Box.*

Ist an einem Ort nur ein Schloss angebracht, so kann die GSM-Funktion direkt im Schloss eingebaut werden. Eine eigenständige GSM-Box Einheit hat nur einen Sinn, wenn an einer Schlossstation mehrere Schlösser angebracht sind. Damit kann man Telefonkosten und Energie sparen. Nachteil ist die indirekte Kommunikation, wie es im Kapitel 3.4 detailliert beschrieben ist.

Die GSM-Box besteht aus einem energiesparenden Funkteil, der mit den Schlössern kommuniziert, einem GSM-Modul, das mit der Zentraleinheit kommuniziert, einem Speicher, auf der die Daten der Schlösser und die Updates zwischengespeichert werden können und einem Mikrocontroller, der alles koordiniert.

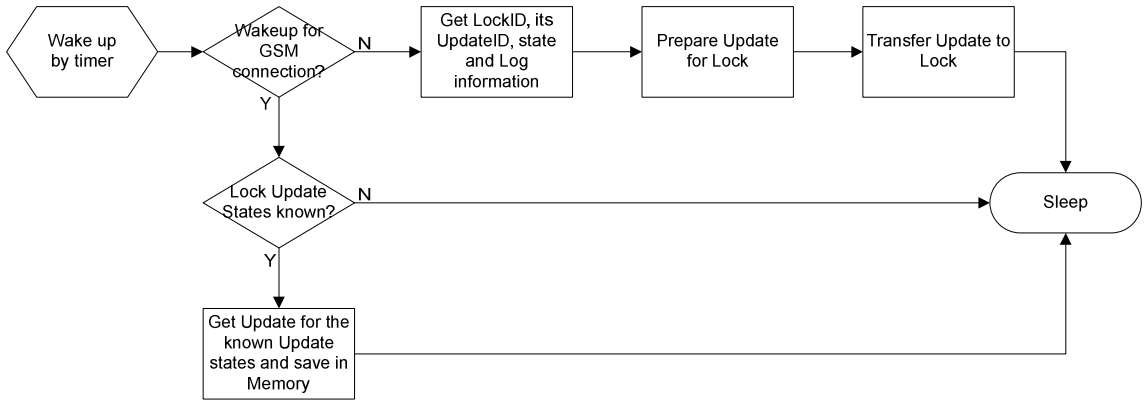


***Abbildung 17**: Ablaufdiagramm der GSM-Box zum Koordinieren der Kommunikation zwischen Server und NFC-Schlössern. Dabei müssen Log-Daten und Statusinformation von den Schlössern zum Server weitergeleitet werden und Schlüsselupdates vom Server zu den Schlössern geschickt werden.*

3.3.3 Zentraleinheit

Auf einem zentralen Rechner müssen wir den Schlössern immer eine aktuelle Datenbank mit den gesperrten Schlüsseln zur Synchronisation zur Verfügung stellen. Außerdem sollte dieser auch über fehlerhafte Aktualisierungen von Schlössern oder allgemeiner Fehlfunktionen in einer bestimmten Einheit informiert sein.

Bei dieser Systemeinheit haben wir kein Energieproblem. Darum sollten möglichst alle Berechnungen und Datenaufbereitungen hier passieren, um den Schlössern das Umgehen mit Updates und Daten zu erleichtern.

Dieser Rechner muss eine Datenbankfunktion, einen FTP-Server und einen Web Server zur Verfügung stellen. Mit einem Linux Rechner haben wir alle Voraussetzungen.

In der Datenbank haben wir also eine Tabelle mit allen Schlüsseln „key“ und einem Eintrag für jedes Schloss in der Tabelle „lock“, wobei über die Tabelle „update“ auf die Schlüsselliste Bezug genom-

men wird. Damit ist immer bekannt, welche Schlüssel auf welchem Schloss für das Zutrittsverbot hinzugefügt werden und welche entfernt werden müssen. Jedes Schloss in der Tabelle „lock“ hat zudem eine Beziehung zu einem Eintrag in der „GSM-Box“ Tabelle. Dadurch weiß die zentrale Einheit, über welche GSM-Box sie kommunizieren muss, um ein bestimmtes Schloss zu aktualisieren und die letzten Daten zu erhalten. Bildlich werden die Zusammenhänge im ER (Entity-Relationship) Modell in Abbildung 18 dargestellt.

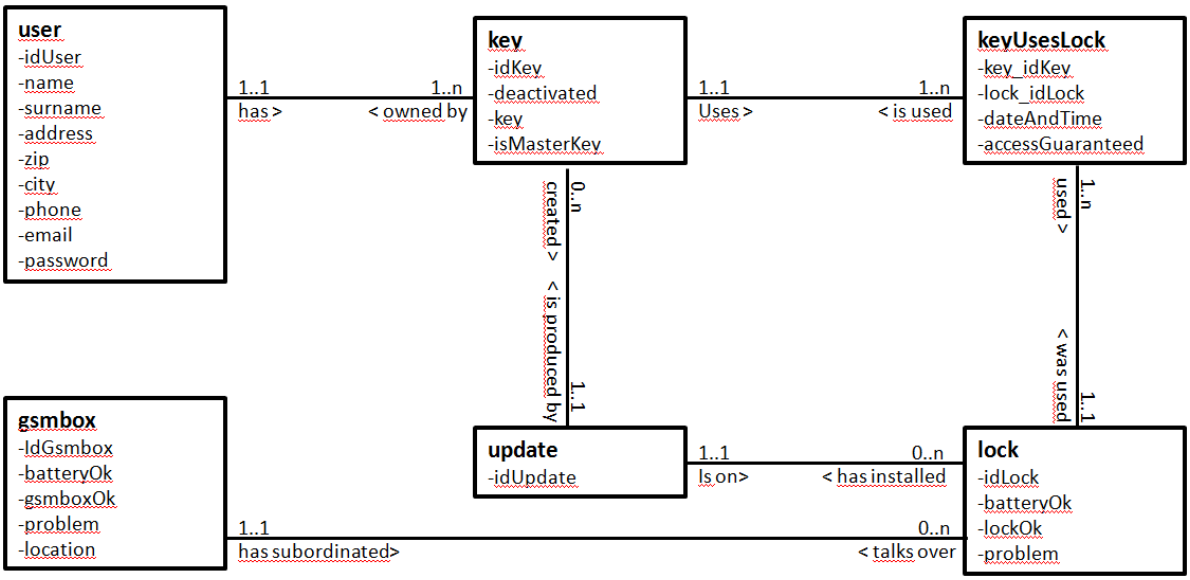


***Abbildung 18**: Aufbau der persistenten Daten angezeigt als Entity-Relationship-Modell. Diese Daten werden in dieser Form auf der Zentraleinheit in einer SQL-Datenbank (Structured Query Language) gespeichert.*

Sobald eine GSM-Box den aktuellen Status der untergeordneten Schlösser der Zentraleinheit mitteilt, werden diese in der Datenbank aktualisiert. Dabei werden die Daten, wie in Punkt 3.4 dargestellt, empfangen. Jedes Schloss wird dann mit folgender SQL-Abfrage in der Datenbank aktualisiert:

```
UPDATE lock
SET update_idupdate=$installedUpdateId,
      batteryOk=$batteryOk,
      lockOk=$lockOk
WHERE idLock=$idLock
```

Auch die Tabelle „keyUsesLock“ wird aktualisiert. Hierbei werden jedem Schloss die verwendeten Schlüssel zugeordnet. Für jeden Log-Eintrag wird folgender Aufruf durchgeführt:

```
INSERT INTO keyUsesLock
SET lock_idLock=$idLock,
       key_idKey=$idKey,
       dateAndTime=$time,
       accessGuaranteed=$allowed
       openLock=$openLock
```

Damit stehen alle Daten zur Verfügung, die der Administrator braucht und die für die Erstellung des Updates, wie unter Punkt 3.4 beschrieben, gebraucht werden. Mit folgender SQL-Abfrage werden jene Schlüssel gefiltert, welche sich seit dem letzten Update geändert haben oder hinzugefügt wurden:

```
SELECT key.*, update.idUpdate
FROM key, update, lock
WHERE update.idUpdate > lock.update_idUpdate
AND update.key_idKey=key.idKey
AND lock.idLock=$idLock
```

Mit der Datenbankstruktur, wie in Abbildung 18 gezeigt, werden außerdem alle Voraussetzungen, die vom Administrationsterminal und vom Benutzerterminal gebraucht werden, erfüllt.

3.3.4 Administrationsterminal

Die gesamte Administration erfolgt direkt auf der Zentraleinheit. Die gesamte Information wird über eine passwortgeschützte Internetseite dem Administrator zur Verfügung gestellt. Damit muss die gesamte Verwaltung des Systems über diesen Terminal erfolgen. Der Systemadministrator muss hier den Betrieb der Schlösser überwachen können. Fehlfunktionen einzelner Schlösser oder einer GSM-Box müssen dem Administrator im Terminal aufscheinen. Genauso muss Information über den Batteriestatus der Schlösser angezeigt werden.

Aber nicht nur Systemmerkmale sollten hier verwaltbar sein, sondern auch die gesamte Benutzerverwaltung, Benutzerregistrierung und das Schlüsselverlustmanagement muss hier durchgeführt werden können. Genaueres über die Implementierung ist im Kapitel 0 beschrieben.

3.3.5 Benutzerterminal

Diese Systemkomponente bietet dem Benutzer jenes Service, das unter Punkt 3.2.2 beschrieben ist. Damit kann durch einen http-Zugriff durch den Browser eine Neuregistrierung durchgeführt, ein neues Passwort über E-Mail angefordert, oder ein Benutzer mit E-Mail und Passwort angemeldet werden. Nach der Anmeldung wird dem Benutzer die Möglichkeit geboten, das eigene Passwort zu

ändern, den Verlust des Schlüssels zu melden und die Statistik über die Schlossbenutzung aufzurufen.

Diese Applikation wird direkt auf der Zentraleinheit ausgeführt. Hierfür eignet sich am besten ein LAMP (Linux Apache MySQL PHP) Server, womit die Sicherheit und die Funktionalität garantiert sind.

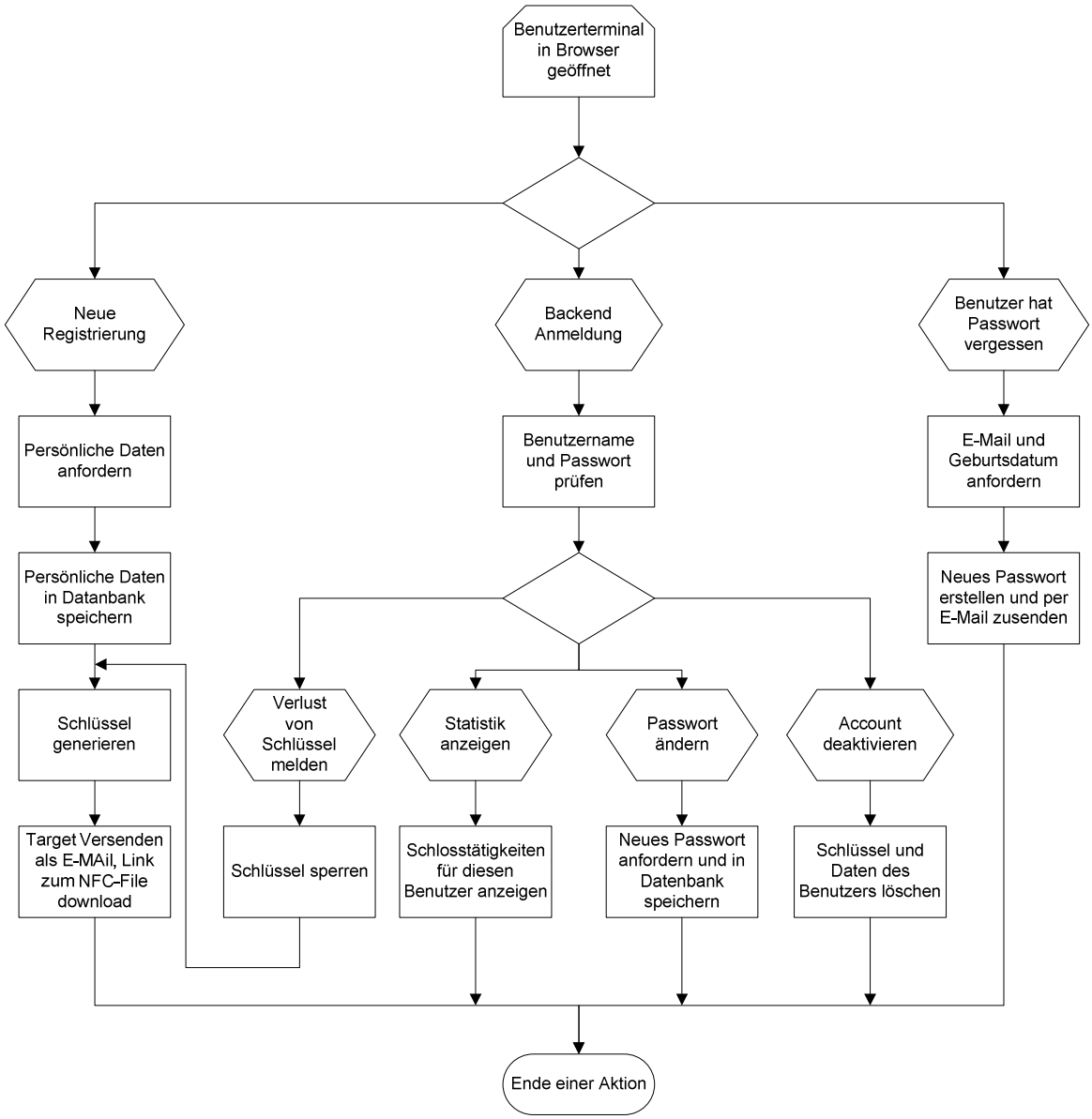


***Abbildung 19**: Funktionalität für den Benutzer. Durch die Verwaltung im Backend kann der Benutzer die wichtigsten Aktionen selbst durchführen, ohne eine Servicestelle zu kontaktieren. Darunter gehören die Registrierung, Schlüsselverlustmeldung und Account deaktivieren.*

3.4 Schlüsselsynchronisierung und Datenaustauschformat

Die Synchronisierung der Schlüssel ist besonders heikel. Man muss immer sicherstellen, dass alle gesperrten Schlüssel auf allen Schlössern richtig synchronisiert wurden. Sollte die Synchronisierung auf einem Schloss einmal fehlschlagen, so müssen bei der nächsten Synchronisierung für dieses bestimmte Schloss andere Daten übermittelt werden. Die Synchronisation geschieht nicht direkt zwischen der Zentraleinheit und dem Schloss. Zuerst wird die GSM-Box synchronisiert, von wo aus anschließend die betreffenden Schlösser synchronisiert werden. Ein Schloss wacht selbstständig auf und kontrolliert auf der GSM-Box, ob es Änderungen der Schlüsseldatenbank gibt. Auch die GSM-Box sucht von sich aus nach Updates auf der Zentraleinheit. Die GSM-Box muss also der Zentraleinheit mitteilen, ob eine Synchronisation mit den Schlössern erfolgreich zustande gekommen ist oder nicht.

Damit werden Schlösser berücksichtigt, die möglicherweise einige Updates ausgelassen haben. Ein Schloss muss sich damit nur die ID des letzten Updates merken und diese beim nächsten Update der GSM-Box mitteilen. Die UpdateID wird bei jeder Änderung auf der Schlüsseldatenbank der Zentraleinheit um eins erhöht. Dem Schloss werden dann nur jene Datensätze übermittelt, bei denen die UpdateID größer ist als jene des Schlosses.

Die GSM-Box teilt dem Server den Updatestatus der Schlösser mit. Dementsprechend wird dann jeder GSM-Box eine eigne, auf die Schlösser angepasste Updatedatei zur Verfügung gestellt. Daraufhin wird diese über FTP heruntergeladen und in der GSM-Box zwischengespeichert, bis ein Schloss ein Update durchführen will.

Man nimmt an, dass einige Schlösser das Update mit der UpdateID X und einige Schlösser das Update mit der UpdateID Y erfolgreich durchgeführt haben und die Zentraleinheit ein Update mit der UpdateID Z zur Verfügung stellt. Es gilt Z > Y > X. Von der Zentraleinheit werden dann die Daten in folgender Form zubereitet, und als XML-Datei der GSM-Box weitergeleitet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<data updateid=Z>
	<bankey>
		<masterkey>E2F12255</masterkey>
		<key>F2E4C332 </key>
		<key>11DDCCCC</key>
	</bankey>
	<allowkey>
		<key>11DD FFFF</key>
		<masterkey>E5556688</masterkey>
	</allowkey>
</data>

<data updateid=Y>
	<bankey>
		<key>11DD AAAA</key>
	</bankey>
```

```
        <allowkey>
               <key>F2E4 C332</key>
        </allowkey>
        <lostkey>
               <oldkey>CCCCCCCC</oldkey>
               <newkey>AAAAAAAA</newkey>
        </lostkey>
</data>
```

In dieser XML-Datei werden folgende Daten übertragen:

- <bankey> Schlüssel, sowie Masterschlüssel, welche gesperrt werden müssen
- <allowkey> Schlüssel, welche gesperrt wurden und wieder freigegeben werden müssen
- <lostkey> verlorene Schlüssel, welche durch einen neuen Schlüssel ersetzt werden. Damit kann ein eventuell mit dem verlorenen Schlüssel abgesperrtes Fahrrad mit dem neuen Schlüssel geöffnet werden.

Sobald sich ein Schloss dann mit der GSM-Box verbindet und die UpdateID mitteilt, werden jene Datenpakete zurückgeschickt, die eine größere UpdateID enthalten. Die Aufgabe die Datenpakete richtig aufzuspalten übernimmt die GSM-Box.

Im Schloss setzt man voraus, dass die Schlüsselliste sortiert ist, um möglichst schnell einen Schlüssel im Speicher zu finden. Nach dem Hinzufügen und Entfernen von Schlüsseln ist es problematisch, die Liste möglichst effizient zu sortieren. Wird ein Element mitten in der Liste eingefügt, so müssen die weiteren Einträge im Speicher um eins weiterverschoben werden. Bei einer Million Einträge kann da ein Update mit möglicherweise 50 neuen Schlüsseln zu einem Zeit- und Energieproblem werden. So muss ausgeschlossen werden, dass unnötig viele Schlüssel gesperrt werden müssen. Müssen also zu viele Targets gesperrt werden, ist man gezwungen, allen Benutzern einen neuen Schlüssel auszuhändigen, der mit einem anderen Passwort geschützt ist (siehe Punkt 3.5.6).

Nach dem Herunterladen des Updates von der GSM-Box auf das Schloss muss für jeden Schlüssel die Position ermittelt werden, an der er im Speicher eingefügt werden soll. Wenn M die Anzahl der Schlüssel ist, die einzufügen ist und N die Anzahl der Schlüssel darstellt, welche sich schon im Speicher befinden, so ist der Aufwand zum Finden des jeweiligen Index:

$$M \cdot \log_2(N) \tag{3.1}$$

So wird beim Einfügen von Schlüsseln am Ende der Liste angefangen den Eintrag in der Liste um so viele Speicherstellen zu verschieben, wie Schlüssel einzufügen sind. Sobald ein Schlüssel in eine durchlaufene Position gehört, wird er eingefügt und die nächsten Einträge werden somit um eine Position weniger verschoben. Beim Löschen von Schlüsseln passiert das Gleiche, allerdings vom Anfang der Liste beginnend.

Im schlimmsten Fall müssen alle hinzuzufügenden Schlüssel am Anfang eingefügt werden und alle zu entfernenden Schlüssel auch vom Anfang des Speichers entfernt werden. Dabei müssen N Speicherlesezyklen und $N+M$ Speicherschreibzyklen durchgeführt werden.
Das Ganze kann man optimieren, indem die Liste mit den Updates bereits sortiert zur Verfügung gestellt wird.

```
//Pseudocode to delete the Keys from the List
Start by the first Element
        X=1
        Go to the first Element that have to be deleted
        while(notEndOfList)
        {
                Take the next Element in the List
                        Needs this Element to be deleted too?
                                Yes: X++
                                No: Shift it back by X
        }

//Pseudocode to add the Keys from the sorted List
Start by the last Element
        X=number of keys to be inserted
        while(notBeginningOfList && X not 0)
        {
                New Element is Bigger than Element in List?
                        Yes: Insert new Element behind Element in List
                                Shifted by X, X--
                        No: Shift Element in List by X, then point to the
                                next Element
        }
```

Auch von den Schlössern werden Informationen zur Zentraleinheit verschickt. Wichtig sind dabei vor allem Fehlermeldungen wie eine fast leere Batterie. Interessant für den Betreiber des Schlosssystems ist auch eine möglichst genaue Statistik. Damit müssen alle Schlüssel übertragen werden, die das Schloss benutzt haben und ob der Zugriff erfolgreich war oder nicht. Auch diese Daten lassen sich als XML-Format gut darstellen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<lockdata>
       <idLock>158</idLock>
       <installedUpdateId>75634</installedUpdateId>
       <batteryOk>1</batteryOK>
       <lockOk>1</lockOk>
       <allowedLock><key>11DD FFFF</key><time>1310365108</time></allowedLock>
       <allowedOpen><key>11DD FFFF</key><time>1310380092</time></allowedOpen>
       <deniedLock><key>11DDCCCC</key><time>1310395000</time></deniedLock>
<allowedLock><key>11DD FFFF</key><time>1310400000</time></allowedLock>
       <deniedOpen><key>11DDCCCC</key><time>1310450000</time></deniedOpen>
</lockdata>
```

Von allen Schlössern fließen diese Daten zuerst auf der GSM-Box zusammen und werden dann zur Zentraleinheit geschickt. Wird von einem Schloss über mehrere Synchronisationszyklen keine Mitteilung empfangen, so wird das als Fehlfunktion des Schlosses eingestuft.

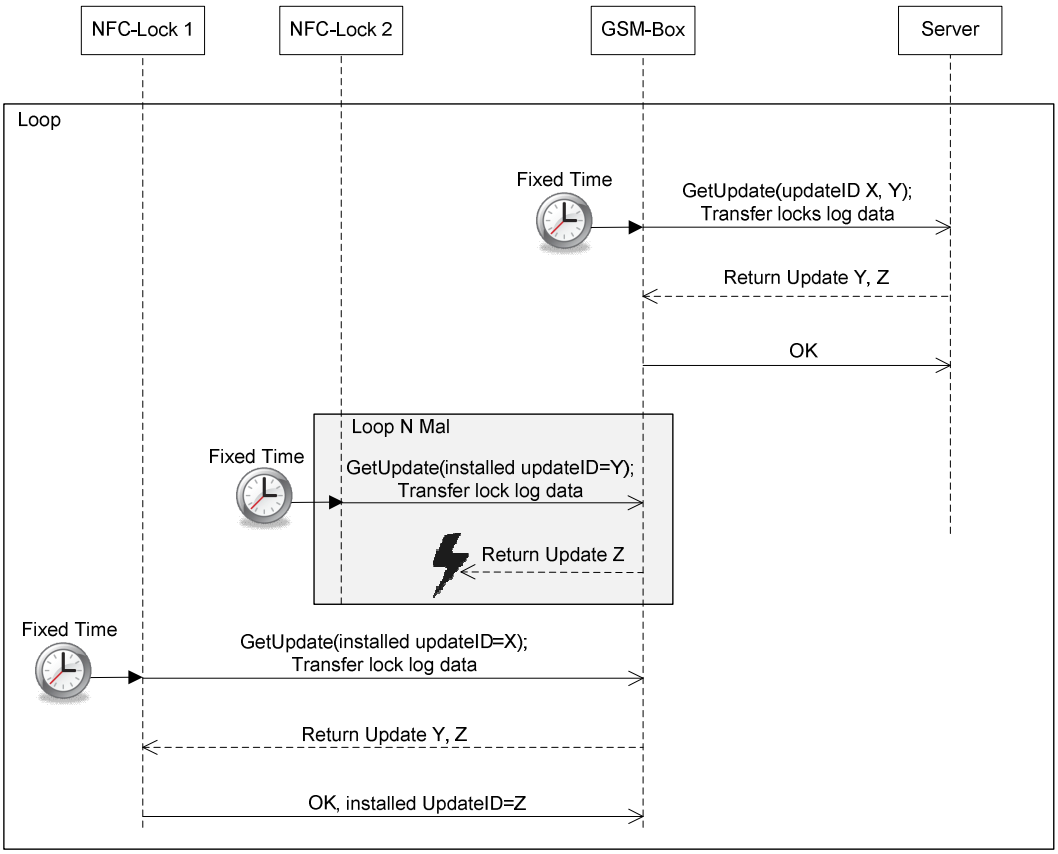


***Abbildung 20**: Sequenzdiagramm der Kommunikation zwischen NFC-Schloss, GSM-Box und Server. Die neue XML-Updatedatei wird vom Server auf Anfrage der GSM-Box mit den bekannten Status der Schlösser erstellt. Für jede GSM-Box wird also eine auf die unterliegenden Schlössern angepasste XML-Datei erstellt.*

3.5 Sicherheit

In diesem Kapitel werden die Risiken ausgewertet und Entscheidungen getroffen, welche Schutzmechanismen gegen unerlaubte Zugriffe vorgenommen werden müssen, damit eine Manipulation des Systems bestmöglich ausgeschlossen werden kann. Das System ist für diverse Angriffsmöglichkeiten anfällig. Um Sicherheit zu garantieren, große Schäden und unnötige Kosten zu vermeiden, muss der Betreiber genaue Kenntnisse über die Schwachstellen haben. Aus diesem Grund konzentriert sich dieses Kapitel vor allem auf die Sicherheit zwischen Lesegerät am Schloss und Targets, die sich über Funk austauschen müssen.

3.5.1 Datenkorruption

Bei diesem Manipulationsversuch hat der Angreifer im Sinn die Kommunikation zwischen Lesegerät und Target zu stören. Es wird aktiv ein Feld so angelegt, dass ein Target die ASK-Modulation des Lesegerätes nicht mehr interpretieren kann und das Lesegerät die Lastenmodulation des Targets nicht mehr wahrnimmt. Damit wird die Funktionalität des Systems verhindert. Man spricht auch von einem „Denial of Service attack“ [SECURITY].

Dies ist im Fall des Fahrradschlosses kein Sicherheitsproblem. Es kann aber passieren, dass ein Schlossbenutzer ein Fahrrad nicht abschließen kann oder auch nicht wieder aufsperren kann. Ein Diebstahl eines Fahrrades ist mit dieser Methode nicht möglich.

3.5.2 Datenmodifikation

Hier versucht der Angreifer das Feld so zu manipulieren, indem er den beiden Kommunikationsparteien glauben macht, die Daten seien echt. Bei einer 100%-igen oder auch 10%-igen ASK-Modulation kann man problemlos die Lücken mit einem ähnlichen Feld auffüllen. Damit wird statt einer logischen ‚0‘ eine ‚1‘ verstanden. Schwieriger wird es, auf dem bestehenden Feld eines so zu überlagern, dass sich diese am Empfänger auslöschen und statt eine logische ‚1‘ eine logisch ‚0‘ interpretiert wird. Dies ist theoretisch zwar möglich, kann in der Praxis aber so gut wie ausgeschlossen werden, da es technisch kaum realisierbar ist [SECURITY].

Hier könnte die Sicherheit in bestimmten Fällen nicht sicher gewährleistet werden. Die einzige Möglichkeit um sich vor solchen Attacken zu schützen besteht darin, eine sichere, verschlüsselte Kommunikation zwischen Lesegerät und Target aufzubauen, wie dies unter Punkt 3.5.5 beschrieben ist.

3.5.3 Dateninsertion

Diese Art von Angriff ist nur möglich, wenn ein Target sehr lange braucht, um zu antworten. Damit hat ein Angreifer Zeit, nach einer Anfrage durch das Lesegerät sofort eine Antwort zurückzuschicken. Die Antwort muss aber vollständig übertragen werden, bevor das angesprochene Target anfängt eine Lastenmodulation durchzuführen und den Angriff stört. Auch dieser Angriff kann mit den Maßnahmen wie unter 3.5.5 beschrieben unterbunden werden.

3.5.4 Authentifizierung durch die CardID

Jede Karte hat eine eindeutige ID-Nummer. Es wird vom Hersteller garantiert, dass eine ID nicht doppelt vergeben wird. Man kann aber die ID allein nicht als sichere Authentifizierung gelten lassen, da man diese ohne Speicherzugriffskontrolle auslesen kann. Ein Dieb könnte durch das Annähern an eine Benutzerkarte mit einem Lesegerät die ID auslesen und dann beim Fahrradschloss, an dem der Benutzer geparkt hat, diese vortäuschen und damit problemlos ein Fahrrad stehlen. Die notwendigen Maßnahmen werden im Kapitel 3.5.5 ausgearbeitet.

3.5.5 Sicherungsmechanismus durch gegenseitige symmetrische Authentifizierung

Damit so ein Fahrradschlosssystem einsetzbar ist, muss die Sicherheit soweit erhöht werden, dass man vor systematischer Manipulation und Sabotage abgesichert ist. Im ISO 9798-2 Standard wird die „Three Pass Mutual Authentication“ festgehalten. Damit hat man auf der Funkstrecke keine Klartextdaten, sondern immer nur verschlüsselte Daten. Transponder und Lesegerät kennen einen gemeinsamen kryptografischen Schlüssel, der jedoch nie über die Funkstrecke übertragen wird.

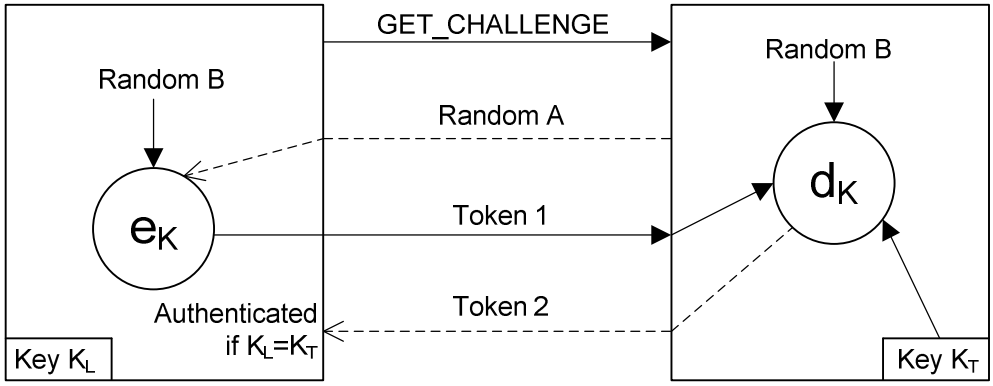


***Abbildung 21**: Symmetrische Authentifizierung. Alle Targets beinhalten den gleichen Schlüssel K_T. Die Authentifizierung ist dann erfolgreich, wenn $K_T = K_L$. Die Sicherheit basiert auf der Geheimhaltung dieses Schlüssels. Wird dieser ausfindig gemacht, so ist das gesamte System manipulierbar. Für ein System mit tausenden von Schlüsseln ist der Einsatz dieser Authentifizierungsmethode zu riskant.*

Die Aufgabe des Lesegeräts und des Targets ist dabei, zu überprüfen, ob beide den gleichen kryptografischen Schlüssel K haben. Sobald das Target vom Lesegerät den Authentifizierungsbefehl „GET_CHALLENGE“ empfängt, schickt er dem Lesegerät unverschlüsselt eine Zufallszahl Z_T. Das Lesegerät verschlüsselt mit dem im ISO 9798-2 beschriebenen Verschlüsselungsalgorithmus e_K und dem Schlüssel K_L die empfangene Zufallszahl Z_T und eine vom Lesegerät erstellte neue Zufallszahl Z_L zu einem Datenpaket.

Nachdem dieses Datenpaket dem Target geschickt wurde, entschlüsselt dieses das Paket mit dem Schlüssel K_T und überprüft, ob sich die gleiche Zufallszahl Z_T ergibt. Wenn die Schlüssel K_T im Transponder und K_L im Lesegerät identisch sind, so ist dies der Fall. Damit auch das Lesegerät die Sicherheit hat, dass die Schlüssel übereinstimmen, verschlüsselt auch das Target die Zufallszahl Z_L und überträgt sie dem Lesegerät. Nach der Entschlüsselung am Lesegerät und der Übereinstimmung der empfangenen Daten mit der vorher verschickten Zufallszahl Z_L ist die Authentifizierung abgeschlossen. Nach der Authentifizierung werden alle übertragenen Daten mit dem gemeinsamen Schlüssel verschlüsselt.

Diese Methode bietet einen sicheren Schutz gegen alle von Punkt 3.5.2 bis 3.5.3 beschriebenen Angriffs-, Manipulations- und Sabotagemöglichkeiten.

Da aber in allen Targets der gleiche Schlüssel K_T verwendet wird, bleibt ein Risiko, dass dieser von jemandem ausgeforscht wird. Das wäre möglich, indem ein Dieb ein Schloss stielt und die Daten zwischen Mikrocontroller und dem PN532 NFC Chip abhört. Hier wird der Schlüssel in Klartext über USART übertragen und stellt damit eine potenzielle Gefahr dar. Dann ist das Schlosssystem unbegrenzten Manipulationsmöglichkeiten ausgesetzt. Wenn einige hunderttausend Targets im Umlauf sind, kann es also passieren, dass man aus Sicherheitsgründen früher oder später erneut alle Targets mit einem neuen Passwort verteilen muss, was unnötige Kosten verursachen würde.

Dafür ist es besser aus der CardID und einem Master Passwort einen abgeleiteten Schlüssel zu erstellen, welcher bei der Ausstellung des Schlüssels einmalig im Transponder programmiert werden muss [WANG09] [HUNT07].

3.5.6 Authentifizierung mit abgeleiteten Schlüsseln

Eine andere Möglichkeit, durch die die Sicherheit noch einmal verbessert wird, ist den Speicher jedes Transponders mit einem anderen kryptografischen Schlüssel zu schützen.

Es kann durch die eindeutige ID-Nummer des Transponders und einem Masterpasswort ein abgeleiteter Schlüssel berechnet werden. Der Algorithmus zum Berechnen abgeleiteter Schlüssel muss garantieren, dass aus der Transponder ID und dem abgeleiteten Schlüssel nicht das private Master-

passwort hergeleitet werden kann. Aus der ID des Transponders mit dem privaten Masterpasswort wird jedoch ein eindeutiger öffentlicher Schlüssel generiert.

Dafür eignet sich als Verschlüsselungsalgorithmus e_K eine SHA-1 kryptografische Hashfunktion. Als Input werden dabei die CardID und das Masterpasswort zu einem String vereint. Der 160bit SHA-Wert, der dabei entsteht, muss auf die Länge des Schlüssels K_T gebracht werden. Dies kann realisiert werden, indem das 160 Bit lange Wort in Teile aufgespalten wird, welche der Länge des Schlüssels K_T entsprechen und diese schließlich ohne Übertrag addiert werden. Damit hat man eine Einwegfunktion, mit der man mit sehr geringer Wahrscheinlichkeit aus der CardID und dem Schlüssel K_T auf das Masterpasswort schließen kann.

Nachdem die Authentifizierung erfolgt ist, können die Daten mittels eines symmetrischen Schlüsselalgorithmus verschlüsselt werden. Damit wird zum Verschlüsseln und Entschlüsseln jener Schlüssel verwendet, welcher sich durch das private Masterpasswort hergeleitet hat.

Die Sicherheit hängt jetzt allein davon ab, wie sicher das private Masterpasswort aufbewahrt wird. Die einzige Möglichkeit für einen Angreifer, den privaten Schlüssel auszuforschen, ist die Firmware des Mikrocontrollers auszulesen. Daher ist es wichtig, die Lesegeräte ohne Debuginformation zu programmieren und das Auslesen des Programmspeichers zu blockieren.

Damit ist die Sicherheit garantiert.

Die Sicherheit kann weiters erhöht werden, indem man im geschützten Speicher des Transponders nochmals einen öffentlichen Schlüssel speichert, welcher aus einer unterschiedlichen Verschlüsselungsfunktion und einem weiteren privaten Schlüssel hergeleitet wird. Dieser wird nach erfolgreicher Authentifizierung ausgelesen. Außerdem wird überprüft, ob er mit dem im Mikrocontroller berechneten Schlüssel übereinstimmt.

Damit können ein unbefugter Zutritt und vor allem eine systematische Manipulation des Systems ausgeschlossen werden.

Nachteil dieser Methode ist, dass hier keine billigen „readonly“-Transponder verwendet werden können und dass alle Transponder einmalig programmiert werden müssen [WANG09] [HUNT07].

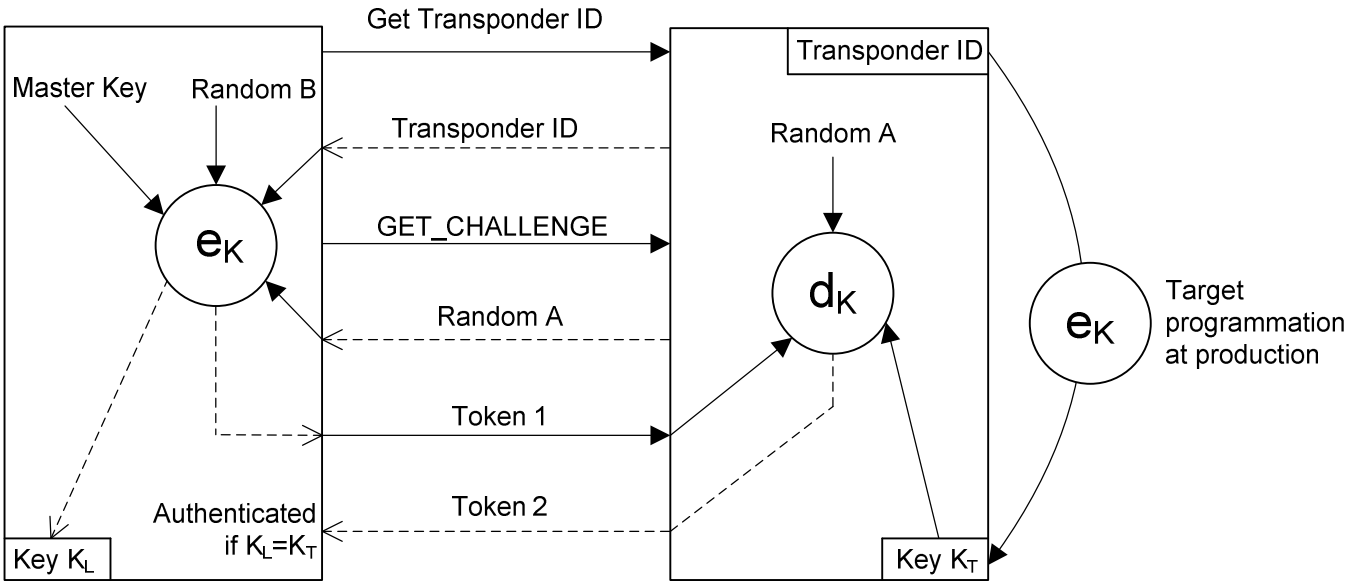


***Abbildung 22**: Durch einen öffentlichen und einen privaten Schlüssel kann die Sicherheit deutlich verbessert werden. Der Schlüssel K_T ist in jedem Target ein anderer. Die Sicherheit basiert auf der Geheimhaltung des Schlüssels „Master Key". Dieser ist in allen Schlössern enthalten. Um somit die Sicherheit zu garantieren, muss das Auslesen der Firmware eines Schlosses unterbunden werden. Damit ist ein systematisches Manipulieren bei der Authentifizierung sehr unwahrscheinlich.*

4. Implementierung eines NFC Fahrradschlosses

4.1 Hardware

Es wurde versucht, die Entwicklungsboards so zu wählen, dass von diesen ausgehend möglichst billig und schnell eine Entwicklungsumgebung zum Programmieren konfiguriert und in weiterer Folge daraus möglichst einfach ein Prototyp gebaut werden kann. Aus diesem Grund wurde untersucht, ob die Schaltpläne und Beispielcodes verfügbar sind und unter welcher Lizenz diese weiterverwendbar sind. Hier wird versucht den Vorteil von Open Source auszunützen und möglichst Schaltpläne, welche unter die Lizenz CC BY-SA, und Software, welche unter GPL veröffentlicht wurden, zu verwenden.

Die Auswahl des NFC-Chips erfolgte über eine Studie, die sich damit beschäftigt, welche Chips in den meisten zukünftigen Geräten eingebaut werden. Dabei finden die Chips von NXP PN532 in Mobiltelefonen weite Verbreitung. Von microBuilder.eu gibt es ein „PN 532 NFC / RFID Breakout Board“ mit der Grundbeschaltung des PN532-Chips samt RF-Antenne bei dem die wichtigsten Pins auf externe Stifte geführt sind.

Als Mikrocontroller wurde, wegen der Energiesparmaßnahmen und Berechnungsfähigkeit, ein STM32 gewählt.

Aus diesem Grund eignete sich das Entwicklungsboard von Olimex STM32-P103 am Besten.

Die beiden Platinen wurden über die USART-Schnittstelle miteinander verbunden und an die Spannungsversorgung gekoppelt. Zum einfachen Programmieren des Mikrocontrollers und Debuggen der Software wurde die STM32-P103 Entwicklungsplatine über den JTAG-Adapter ARM-USB-TINY an die USB-Schnittstelle des Entwicklungsrechners angeschlossen. Somit wurde eine Hardwarekonfiguration zum Programmieren der Software bereitgestellt.

Um den GSM- und Funkteil simulieren zu können, wird die STM32-P103 Entwicklungsplatine über USART mit dem Entwicklungsrechner angeschlossen.

Das PN532 Breakoutboard wurde, um mit der STM32-P103 Entwicklungsplatine zu kommunizieren, folgendermaßen konfiguriert:

- Pin I0 und Pin I1 auf Masse hängen, indem man die Jumper SEL0 und SEL1 einstellt. So wird am PN532 Chip der Kommunikationsmodus auf USART mit standardmäßig eine Baudrate von 115200 bps (bit per second), 1 Startbit, 8 Bit Daten und 1 Stopbit eingestellt.
- eine Spannungsversorgung von 3,3V anschließen
- SSEL/SCL/RX am UEXT-Anschluss 3 und MOSI/SDA/TX am UEXT-Anschluss 4 der STM32-P103 Entwicklungsplatine anschließen
- Weiters brauchen keine weiteren Pins am PN532-Breakoutboard angeschlossen werden.

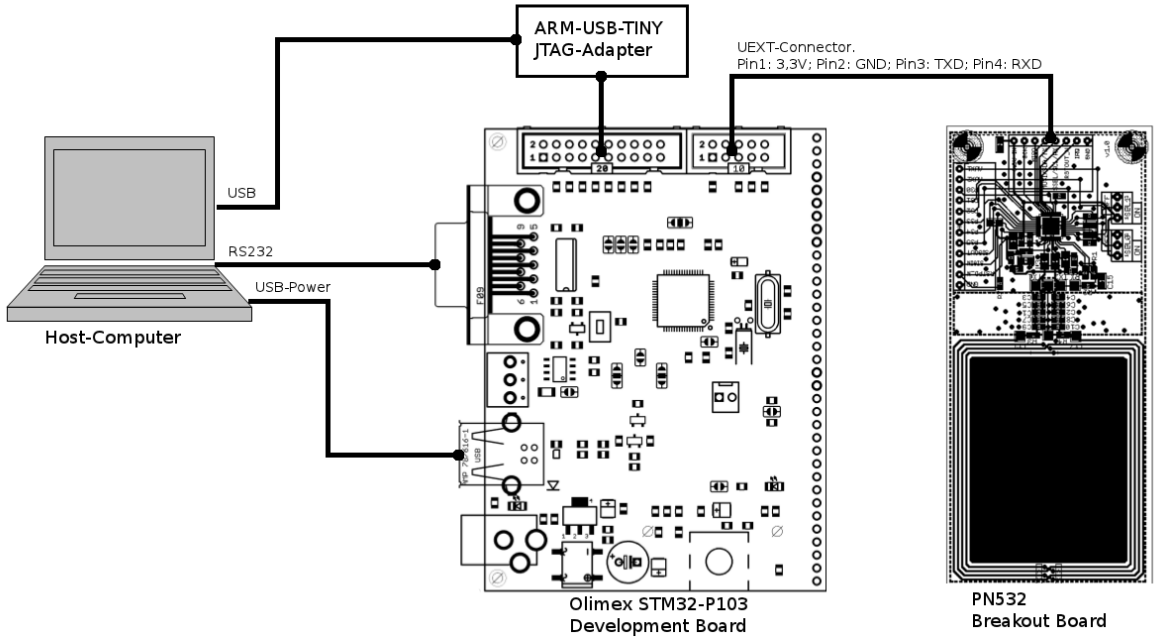


***Abbildung 23**: Laboraufbau zum Entwickeln eines NFC-Schlosses. Zusammenschluss der Entwicklungsplatine STM32-P103 von Olimex und PN532-Breakoutboard. Die Spannungsversorgung der STM32-P103 Platine wird über USB und für das PN532-Board über den UEXT-Connector sichergestellt.*

Der Teil des **Näherungssensors** wurde separat behandelt, da im Entwicklungsstadion das Abschalten der STM32-P103 Platine und des PN532-Breakoutboards nicht erwünscht ist. Zum Testen des Näherungssensors wurde die Entwicklungsplatine „Capacitive Proximity Switch Evaluation Kit OM11055“ verwendet. Diese ist mit dem Chip NXP PCF8883 bestückt. Der Chip ist ein integrierter kapazitiver Näherungssensor mit automatischer Kalibrierung. Langsame Änderungen der Kapazität an der Sensorplatte werden automatisch ausgeregelt. Zusätzlich können die Sensibilität, die Schaltzeit und der Schaltmodus dieses Sensors durch äußere Beschaltung genau eingestellt werden. Die

Sensibilität kann sehr hoch eingestellt werden, was das Erfühlen einer sich annähernden Hand auch durch ein Gehäuse ermöglicht.

Die Grundbeschaltung des Näherungssensors NXP PCF8883 ist in Abbildung 24 dargestellt:

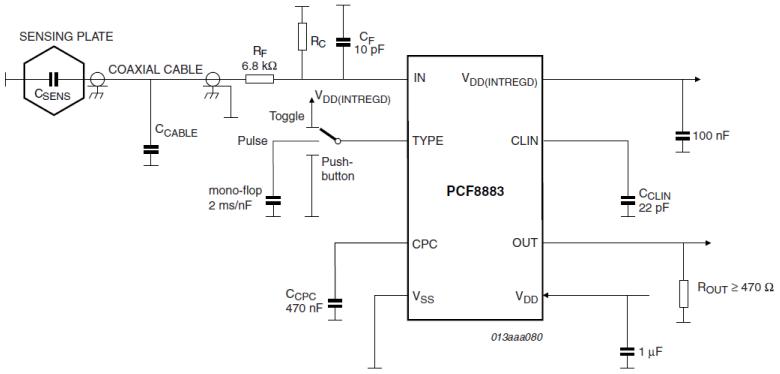


***Abbildung 24**: Grundbeschaltung des Näherungssensors PCF8883. Man kann die Schaltcharakteristik, die Geschwindigkeit und Sensibilität durch Variieren der diskreten Bauelemente anpassen [NXPPCF].*

Durch den Kondensator C_{CPC} kann die Sensitivität des Sensors eingestellt werden. Je größer dabei C_{CPC} gewählt wird, desto sensibler ist der Sensor eingestellt. Das heißt, die Größe der Sensorplatte kann verkleinert werden, oder aber auch, dass der Sensor bereits in einer größeren Entfernung geschaltet werden kann. C_{CPC} kann laut Spezifikation zwischen 90 nF und 2500 nF gewählt werden.

Über R_C wird die Gesamtkapazität des Sensors kalibriert. Dieser wird nur gebraucht, falls die Verbindung zwischen Sensorplatte und Sensorbaustein mehr als 0,5 m beträgt. Die Gesamtkapazität sollte laut Datenblatt zwischen 10 pF und 60 pF liegen.

Mit C_{CLIN} wird die Schaltgeschwindigkeit eingestellt. Dadurch können schnelle Störungen unterdrückt werden und Fehldetektionen vermieden werden. Über C_{CLIN} kann die Abtastfrequenz am Input Pin IN eingestellt werden. Je höher die Abtastfrequenz, desto schneller schaltet der Näherungssensor und desto schneller wird automatisch nachkalibriert. Um C_{CLIN} zu dimensionieren, ist es wichtig zu wissen, wie schnell sich das Triggerobjekt der Sensorplatte annähert. Eine sich zu langsam annähernde Hand könnte automatisch ausgeregelt werden. Die Abtastfrequenz kann für den NXP PCF8883 folgendermaßen berechnet werden:

$$T(fk)[\mu s] = 300\mu s + C_{CLIN}[pF] \cdot 33\frac{\mu s}{pF}$$

C_{CLIN} kann je nach gewünschter Abtastfrequenz zwischen 0 nF und 100 nF gewählt werden [OLIMEXSTM08] [NXPPN532] [NXPPCF].

4.2 Aufsetzen der Entwicklungsumgebung

Auf dem Entwicklungsrechner wurde eine Entwicklungsumgebung so konfiguriert, dass das Programmieren und Debuggen möglichst einfach durchgeführt werden kann. Notwendig dabei war eine integrierte Entwicklungsumgebung, ein Crosscompiler, ein On-Chip Debugger und ein Treiber für den JTAG-Adapter.

Die Installation des Treibers für den JTAG-Adapter ARM-USB-TINY erwies sich dabei als etwas problematisch, da auf dem Entwicklungsrechner eine Windows-Version mit 64 Bit installiert war. Dabei war es notwendig Windows im Test Mode zu starten und den „Integrity Check“ der Treiber zu deaktivieren. Damit war es möglich, mit dem von Olimex zur Verfügung gestellten Treiber, den JTAG-Adapter in Windows als FTDI-Port unter den Geräten erfolgreich aufzulisten.

Als On-Chip Debugger und Programmierer wurde die Open Source Lösung OpenOCD verwendet. OpenOCD ermöglicht für eingebettete Systeme, durch einen umfangreichen Befehlssatz, eine direkte Kontrolle über die Hardware. Es stellt die Verbindung zwischen JTAG-Interface und Debugger her. OpenOCD wir als Dienst im Hintergrund auf dem Host Computer gestartet und wartet standardmäßig auf Port 3333 auf eine Verbindung vom Debugger. Auf der anderen Seite wartet er auf Events des Targets, fängt diese ab und verwaltet sie. Durch eine Konfigurationsdatei können verschiedene Hardwareparameter wie der genaue CPU-Typ und die Variante des Targets eingestellt, sowie die Konfiguration des JTAG-Adapters angepasst werden. Weiters kann durch OpenOCD der Flash Speicher des Targets gelöscht, gelesen oder geschrieben werden. In der Konfigurationsdatei muss also enthalten sein, an welcher Speicherstelle OpenOCD zum Schreiben anfangen sollte [OPENOCD08] [ECLIPSEOPENOCD].

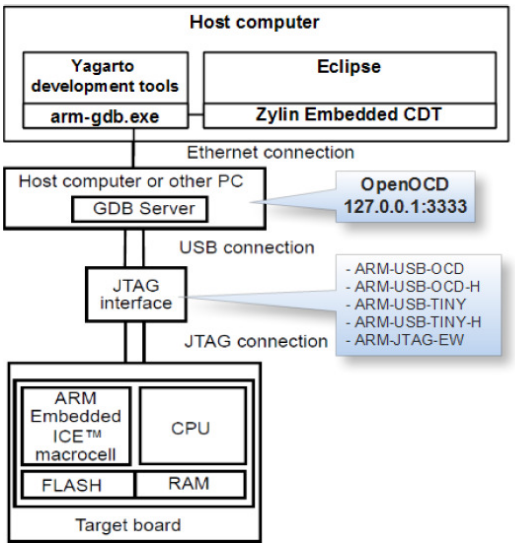


***Abbildung 25**: Entwicklungsumgebungsaufbau. Zusammenspiel der ARM-Toolchain, Eclipse-Entwicklungsumgebung, On-Chip-Debugger OpenOCD und JTAG Interface zwischen Hostplatform und Target [ECLIPSEOPENOCD].*

Zum Cross-Kompilieren der Software für den STM32 Mikrocontroller wurde die Yagarto Toolchain verwendet. Diese beinhaltet den benötigten Compiler, Bibliotheken und GDB-Debugger und steht frei zur Verfügung. Vom Cross-Compiler oder auch Target-Compiler allgemein spricht man dann, wenn man auf einer Hostplattform die ausführbaren Dateien für eine andere Plattform erstellt. In diesem Fall wird der Code auf dem Entwicklungsrechner für den STM32-Mikrocontroller kompiliert [YAGARTO].

Als integrierte Entwicklungsumgebung wurde Eclipse IDE verwendet. Die Gründe dafür sind die große Flexibilität und Konfigurierbarkeit sowie die einfache Integration externer Toolchains. Um diese Entwicklungsumgebung mit dem ARM-Cross-Compiler von Yagarto und OpenOCD zu verknüpfen, waren folgende Schritte notwendig:

1. Als erster Schritt wurde die Java Runtime Environment (JRE) von Sun Microsystems installiert. Diese ist zum Installieren von Eclipse IDE selbst notwendig.
2. Die Eclipse-Version mit C/C++ Unterstützung wurde heruntergeladen und installiert. Dann wurde ein „Workspace“ eingerichtet.
3. CDT-Master herunterladen. Anschließend unter Eclipse unter „Help->Install New Software“ ein Repository hinzufügen und den Pfad zum heruntergeladenen CDT-Master angeben.
4. Nun werden alle Pakete, die mit CDT zusammenhängen, angezeigt. Diese werden alle installiert. Damit wird sichergestellt, dass Eclipse den Support für den C/C++ Debugger und Cross Compiler unterstützt.
5. Jetzt kann ein neues C-Projekt angelegt werden. Dabei sollte nicht die MinGW oder Cross GCC als Toolchain verwendet, sonder die „Other Toolchain“ ausgewählt werden. Als Projekttyp sollte ein „Empty Makefile Project“ ausgewählt werden.
6. Unter „Project > Properties“ muss unter „C/C++ Build > Settings“ noch „GNU Elf Parser“

für den „Binary Parser“ ausgewählt werden.

7. Jetzt kann bereits ein Testcode z. B. von Olimex in Eclipse über „File > Import“ durch Auswählen des Quellenverzeichnisses in Eclipse importiert werden.
8. Über „Project > Clean“ bestehende kompilierte Dateien löschen und unter „Project > Build Project“ dann alles neu kompilieren. Nachdem alles erfolgreich kompiliert wurde, wird eine ELF-Datei (Executable and Linkable Format) erstellt.
9. Um nun den Debugger richtig zu konfigurieren, sind einige Schritte notwendig, damit Eclipse mit dem OpenOCD-GDB-Server richtig kommuniziert. Als erster Schritt muss in Eclipse unter „Help > Install New Software“ das Paket „Zylin Embedded CDT“ nachinstalliert werden. Dieses erlaubt, die vom Kompiler erstellte ELF-Datei über die OpenOCD Kommandos dem Debugger-Server zu übergeben.
10. Dabei muss unter „Run > Debug Configurations“ und unter der linken Baumansicht „Zylin Embedded debug“ ausgewählt werden. Dann sollte eine neue Konfiguration durch „New Launch Configuration“ erstellt werden. Unter dem „Main“ Tab muss der Pfad zur ELF-Datei und dem „Debugger“ Tab der Pfad zum Debugger „arm-none-eabi-gdb.exe“ angegeben werden. Die OpenOCD-Kommandos können unter dem Tab „Commands“ eingestellt werden.
11. Nach dem Starten des OpenOCD-GDB Servers kann nun die Debugkonfiguration, über „Run > Debug As“ aufgerufen werden. Die ELF-Datei, welche nun auf dem Mikrocontroller ausgeführt wird, kann durch Eclipse jetzt problemlos debuggt werden [OPENOCD08].

Damit ist die Entwicklungsumgebung zum Programmieren und Debuggen des Mikrocontrollers konfiguriert. Zum Entwickeln der Serversoftware ist es notwendig, eine Webentwicklungsumgebung aufzustellen. Hierfür wurde NetBeans IDE verwendet. Hier gibt es Erweiterungen für die Programmierung von PHP-CSS Webseiten. Zum Einrichten der Netbeans-Entwicklungsumgebung sind folgende Schritte notwendig:

1. Herunterladen von NetBeans IDE von http://netbeans.org und dessen Installierung. Netbeans wird für Windows, Linux, OS X und Solaris zur Verfügung gestellt.
2. Nach der Installation NetBeans starten und unter Tools > Plugins folgende Zusatzpakete nachinstallieren: PHP, PHP Smarty Framework, LessCSS Module.
3. Dann sollte noch eine Webserver Software auf dem Entwicklungsrechner installiert werden, damit die geänderten Dateien nicht immer auf einem Webserver hochgeladen werden müssen, um die Funktionalität zu testen. Unter http://apache.org das Installationspaket herunterladen und installieren. Es sollte bereits im Browser unter http://localhost/ eine Seite von Apache angezeigt werden.
4. Nun kann bereits ein neues Projekt angelegt werden. Unter File > New eine neue PHP > PHP Applikation auswählen.
5. Unter der Run Configuration Run As > Local Web Site auswählen und kontrollieren, ob die Project URL mit der Apache-Konfiguration übereinstimmt.
6. Unter den PHP Frameworks noch Smarty Framework auswählen und die Entwicklungsumgebung ist bereit zum Programmieren von PHP-CSS Anwendungen.

4.3 Realisierung Software

Die Realisierung der Software unterteilt sich in die Software, welche auf dem Mikrocontroller ausgeführt wird und in die Software auf dem Server. Dabei wird versucht, die meisten Vorgänge auf dem Server durchzuführen, um so viel Energie wie möglich an den batteriebetriebenen Teilen des Systems einzusparen. Um die Vorgänge mit möglichst wenig Rechenleistungsoverhead zu belasten und um nur die notwendigsten Abläufe durchführen zu müssen, muss die Programmierung des NFC-Schlosses eine harwarenahe sein. Die Serveranwendung wird damit als Web-Anwendung realisiert, sodass das Interagieren mit den Benutzern und dem Administrator weltweit vereinfacht zugänglich gemacht werden kann.

4.3.1 NFC Schloss

Die Programmierung des Mikrocontrollers sollte möglichst hardwareunabhängig sein und möglichst leicht zum Portieren auf einen anderen Mikrocontroller sein. Dafür eignet sich ein HAL (Hardware Abstraction Layer), an dem die wichtigsten Methoden zur Verfügung gestellt werden und durch eine Konfigurationsdatei alle Parameter für die Peripherie und Schnittstellen eingestellt werden können. Für den STM32 wird von STMicroelectronics die STM32 Standard Peripheral Library zur Verfügung gestellt. Diese besteht aus dem Treiber für die jeweilige Peripherie und die jeweiligen Funktionen für die volle Peripheriefunktionalität. So können C-Programme leichter erstellt werden, ohne die Spezifikation der Peripherie genau studieren zu müssen. Dies ermöglicht es, sich bei der Entwicklung mehr auf die Ausprogrammierung der Funktionalität und weniger auf das Studieren des genauen Aufbaus des Mikrocontrollers zu konzentrieren. Damit spart man bei der Entwicklung Zeit und die Applikationsfunktionalität kann schneller implementiert werden. Sollte später in der Produktion ein anderer Mikrocontroller verwendet werden, so wird auch das Portieren der Software erleichtert. Es muss dabei lediglich die Konfiguration der Peripherie angepasst werden. Der vom Entwickler erstellte Quellcode basierend auf der STM32 Standard Peripheral Library kann unverändert übernommen werden. Die STM32 Standard Peripheral Library hält sich an den CMSIS-Standard (Cortex Mikrocontroller Software Interface Standard). Damit wird auch das Portieren auf Mikrocontroller sichergestellt, die nicht von STMicroelectrolics stammen.

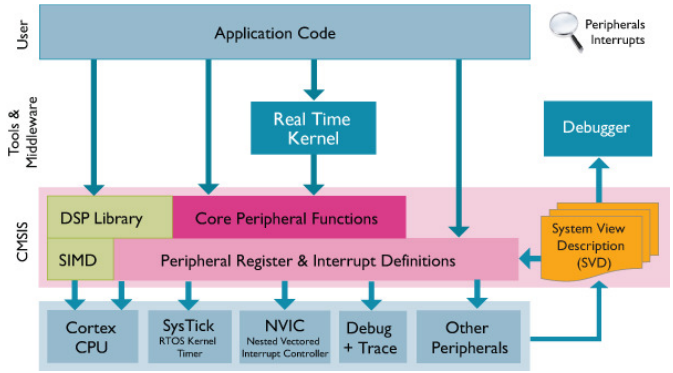


***Abbildung 26**: CMSIS (Cortex Mikrocontroller Software Interface Standard). Hardware Abstraction Layer für Cortex Mikrocontroller. Dem Programmierer werden fertige Funktionen zum Zugreifen auf die Peripherie angeboten, ohne die genaue Spezifikation lernen zu müssen. Der „Application Code" kann somit leicht auf andere Cortex Mikrocontroller portiert werden [CMSISHAL].*

Für die Realisierung der Software wurde also eine CMSIS-CM3 HAL (Hardware Abstraction Layer) verwendet. Im Allgemeinen wurden die Abläufe wie unter Punkt 3.3.1 implementiert [CMSISHAL].

Im Folgenden wird über die Initialisierung des PN532-Bausteins, sowie über das Auslesen der Target-ID berichtet. Durch die Pinkonfiguration, wie unter 4.1 beschrieben, wird auch der Standardkommunikationsmodus ausgewählt. Eingestellt wurde die USART-Schnittstelle (Universal Asynchronous Receiver/Transmitter). Die Datenpakete, welche der NFC-Chip erwartet, sind folgendermaßen aufgebaut:

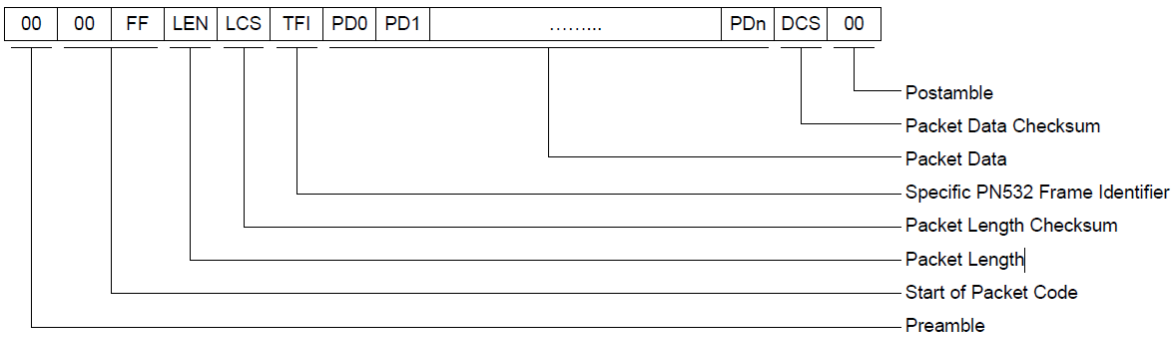


***Abbildung 27**: Aufbau eines Datenframes, wie zwischen Mikrocontroller und PN532 kommuniziert wird. Es enthält Information über den Start, die Länge der Daten, Checksummen, Daten und den Postamble.*

- PREAMBLE: 1 Byte 0x00
- START CODE: 2 Bytes (0x00 and 0xFF)
- LEN: 1 Byte besagt, wie viele Datenbytes im Datenpaket enthalten sind. Dabei zählen TFI und PD0 bis PDn. Damit sind wir mit dem einfachen Datenpaket auf 255 Datenbytes begrenzt.
- LCS: 1 Byte Checksumme für die Länge des Pakets. LCS muss dabei folgende Bedingung erfüllen: LEN + LCS = 0x00,
- TFI: spezifisch für den PN532 Chip. TFI kann zwei Werte annehmen:
    - 0xD4, falls die Daten vom Mikrocontroller zum PN532-Chip gesendet werden und
    - 0xD5, falls die Daten vom PN532-Chip zum Mikrocontroller geschickt werden.
- DATA [LEN-1]: Nutzdatenpaket bzw. Kommandos
- DCS 1: Checksumme über die Nutzdaten. DCS wird folgendermaßen zusammengestellt: TFI + PD0 + PD1 + … + PDn + DCS = 0x00
- POSTAMBLE 1 Byte 0x00

Wird ein Paket vom PN532 erfolgreich erhalten, antwortet dieser mit einem Acknowledgeframe, was folgendermaßen aussieht: 0x00,0x00,0xFF,0x00,0xFF,0x00. Wird ein Paket nicht richtig empfangen oder stimmen die Checksummen nicht überein, so wird mit einem NACK-Frame geantwortet: 0x00,0x00,0xFF,0xFF,0x00,0x00.

Nachdem die Spannungsversorgung am PN532-Breakoutboard angeschlossen wird, ist der Chip standardmäßig im „Power Down Mode“. Um den Chip aufzuwecken, kann man einen Befehl mit einem langen PREAMBLE schicken. Laut Spezifikation ist die Bedingung zum Aufwachen nach der 5ten empfangenen steigenden Flanke. Empfohlen wird das Schicken der Bytefolge 0x55,0x55, darauf folgend mindestens 14 Mal 0x00 und anschließend das Übermitteln eines Datenframes mit dem gewünschten Befehl. Nach dem Aufwecken sollte als Erstes der Modus ausgewählt werden, mit dem das Verhalten des PN532-Chips geregelt wird. Die Initialisierung erfolgt, indem man den „Normal Mode“ wählt. Es wird der Befehl PD0 bzw. PD1 0x14,0x01 geschickt. Der gesamte Frame sieht also folgendermaßen aus: 0x00, 0x00, 0xFF, 0x03, 0xFD, 0xD4, 0x14, 0x01, 0x17, 0x00. Dadurch wird der Chip in den Lesegerätmodus gestellt. Mit dem Befehl 0x14,0x02,0x00 würde der PN532 in den „Virtual Card Mode“ umgestellt. Damit könnte sich dieser identisch wie ein Target verhalten.

Um den PN532 wieder in den „Power Down Modus“ zu versetzen, wird das Kommando 0x16, 0x10 gegeben. Durch den Parameter „WakeUpEnable“ 0x10 wird das wieder Aufwachen durch die HSU (High Speed Usart) ermöglicht.

Nach der Initialisierung kann beliebig mit dem PN532 kommuniziert werden [NXPPN532UM].

4.3.2 Serversoftware

Die serverseitige Software sollte so gestaltet werden, dass es möglichst einfach ist, das System zu erweitern, Neuerungen zu implementieren und die Sicherheit zu garantieren. Zusätzlich sollte es für den Benutzer und Administrator möglichst einfach sein, auf das System zuzugreifen. Somit ist es sinnvoll eine Web-Anwendung zu entwickeln, da diese von jedem Computer mit Browser aufrufbar ist.

Um die Anzeige der Anwendung von der Datenbank, sowie Programmlogik zu trennen, wurde die Anwendung nach dem Architekturmuster MVC (Model View Controller) geplant.

Die Benutzeranzeige basiert auf dem „Smarty PHP Template Engine“. Damit können HTML-Vorlagen erstellt werden, die allein für die Anzeige verantwortlich sind und im Allgemeinen keine Logik enthalten. Sich ändernde Variablen und Werte in der Vorlage können an Platzhalter vom PHP-Programm übergeben werden. Zum Programmieren wurde die Skriptsprache PHP5 ausgewählt, da diese das Programmieren mit Klassen ermöglicht und die für das Webprogramm benötigten Funktionalitäten bietet. Zusätzlich ist PHP sehr gut geeignet mit Datenbanken zu arbeiten. Unter den verfügbaren Datenbanken wurde das Open-Source-Datenbanksystem MySQL ausgewählt. Durch verbinden der drei Komponenten PHP, MySQL und Smarty hat man die Grundlage, um komplexe persistente Systeme zu entwerfen.

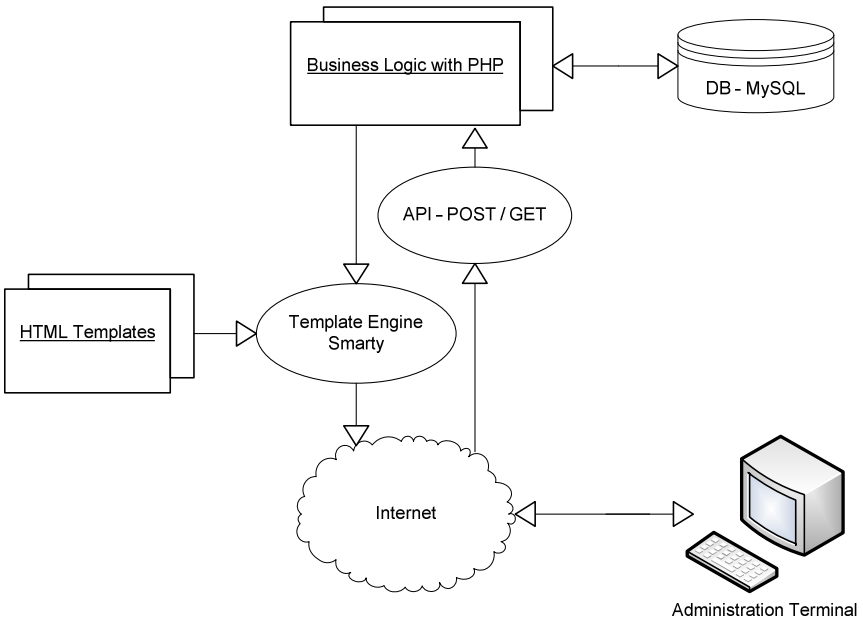


***Abbildung 28**: Aufbau der Serversoftware. Die Programmlogik wird vom User Interface so weit wie möglich abgekoppelt.*

Da es sich bei einer Web-Anwendung um ein verteiltes System handelt, muss man sicherstellen, dass die Transaktionen in der Datenbank immer vollständig durchgeführt werden, bevor eine weitere Instanz des Programms versucht Änderungen am gleichen Datensatz durchzuführen. Um dies zu garantieren, ist eine genaue Definition des Interfaces zwischen Server und Client in den Browsern notwendig. Außerdem muss das Arbeiten mit der Datenbank durch Transaktionen erfolgen.

Die Kommunikation von Server zu Client erfolgt allein über das Anzeigen einer dynamischen HTML-Seite, die im Fall auch Java Script enthalten kann, um die Logik des UI (User Interfaces) zu implementieren. Die Kommunikation von Client zum Server kann hingegen über verschiedene Methoden passieren. Eine Möglichkeit besteht darin, durch ein einfaches Aufrufen einer URL (Uniform Resource Locator) dem Server eine bestimmte Aufgabe mitzuteilen, die dann wiederum mit einer dynamischen HTML-Seite beantwortet wird. Eine weitere Möglichkeit, welche vor allem dann zum Tragen kommt, wenn der Client mehr Daten an den Server mitteilen will, ist es den Aufruf der URL mit einem GET oder POST Request-Methode zu erweitern. Durch die GET-Methode werden die gewünschten Daten durch Erweitern der URL-Anfrage mit Parametern versendet. Da aber in manchen Browsern, wie Microsoft Internet Explorer, die maximale URL-Länge auf 2083 Zeichen limitiert ist, ist man mit den übertragbaren Datenmengen limitiert. Geeigneter ist die POST-Methode, welche vor allem für das verschicken von Formularen geeignet ist. Eine Limitierung im Sinne von Datenmenge gibt es hier keine.

Im folgenden einige Ansichten der Administrationssoftware:

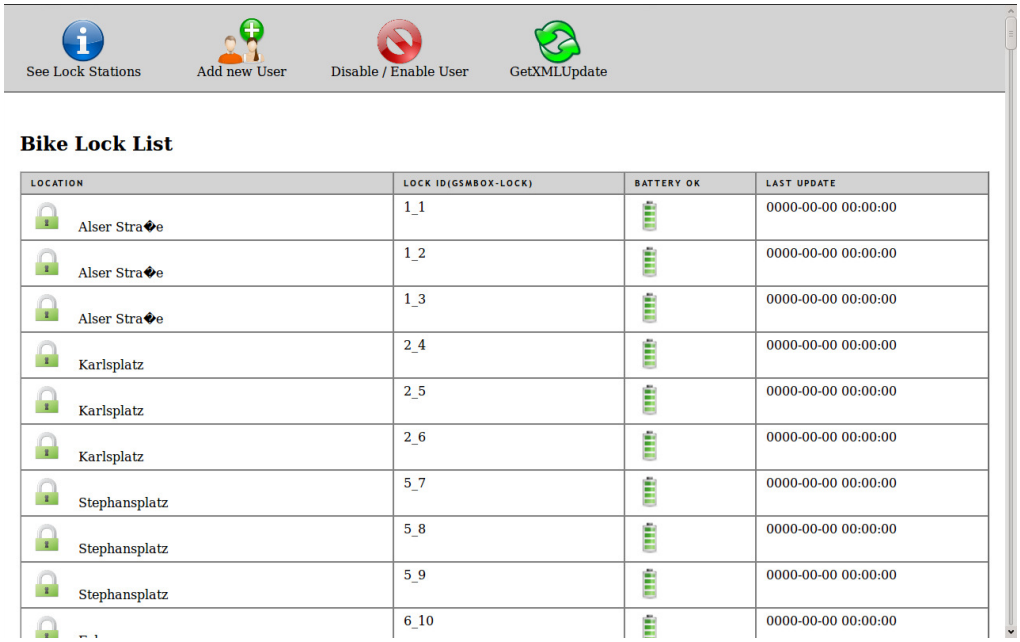


Abbildung 29: *Ansicht der Schlossstationen und der einzelnen Schlösser. Sobald von den Schlössern Information über den Batteriestatus sowie Schlossfehlfunktionen erhalten wird, können die Icons automatisch angepasst werden, damit sofort auffällt, wenn an irgendeinem Schloss Probleme auftreten. Diese Seite ist vor allem für die Wartung und für die Betriebsüberwachung.*

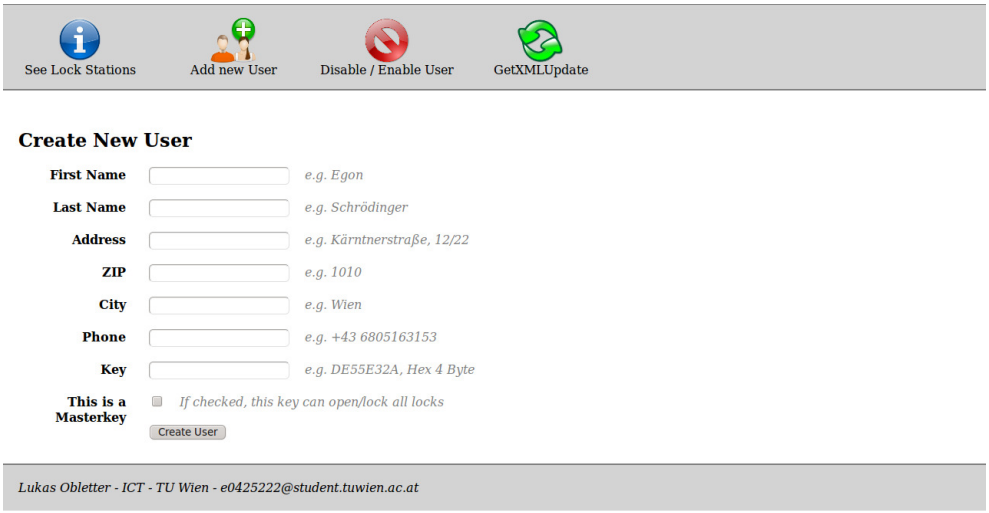
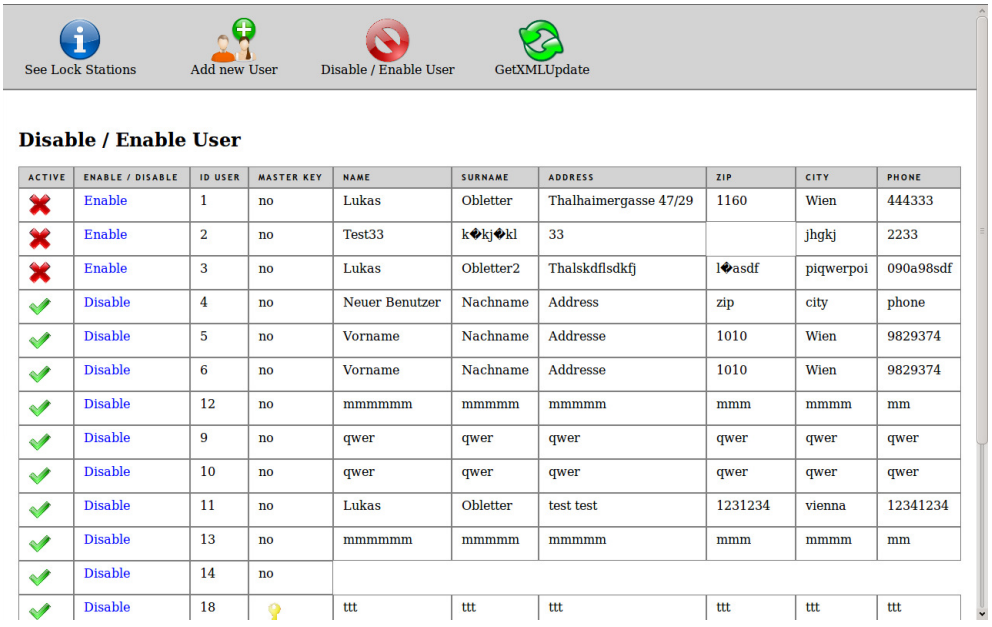


***Abbildung 30**: Durch diese Seite können an einer Servicestelle neue Benutzer eingetragen werden oder neue Masterschlüssel erstellt werden. Die Masterschlüssel können alle Schlösser öffnen und schließen. Der Schlossbenutzer sieht auf der Werbeseite für das Schlosssystem nur das Formular integriert. Damit kann ein Benutzer online einfach und schnell ein Account für das Schlosssystem erstellen.*



ACTIVE	ENABLE / DISABLE	ID USER	MASTER KEY	NAME	SURNAME	ADDRESS	ZIP	CITY	PHONE
✗	Enable	1	no	Lukas	Obletter	Thalhaimergasse 47/29	1160	Wien	444333
✗	Enable	2	no	Test33	k□kj□kl	33		jhgkj	2233
✗	Enable	3	no	Lukas	Obletter2	Thalskdflsdkfj	l□asdf	piqwerpoi	090a98sdf
✓	Disable	4	no	Neuer Benutzer	Nachname	Address	zip	city	phone
✓	Disable	5	no	Vorname	Nachname	Addresse	1010	Wien	9829374
✓	Disable	6	no	Vorname	Nachname	Addresse	1010	Wien	9829374
✓	Disable	12	no	mmmmmm	mmmmm	mmmmm	mmm	mmmm	mm
✓	Disable	9	no	qwer	qwer	qwer	qwer	qwer	qwer
✓	Disable	10	no	qwer	qwer	qwer	qwer	qwer	qwer
✓	Disable	11	no	Lukas	Obletter	test test	1231234	vienna	12341234
✓	Disable	13	no	mmmmmm	mmmmm	mmmmm	mmm	mmmm	mm
✓	Disable	14	no						
✓	Disable	18		ttt	ttt	ttt	ttt	ttt	ttt

***Abbildung 31**: Sperren oder Freischalten von bestimmten Benutzern im Fall von Schlüsselverlust oder Missbrauch. Außerdem werden die persönlichen Daten der Benutzer angezeigt, damit im Fall eine Kontaktaufnahme möglich ist.*

Die XML-Datei, wie unter Punkt 3.4 beschrieben ist, welche den GSM-Boxen zum herunterladen zur Verfügung gestellt wird, wird dynamisch zum Zeitpunkt des Aufrufs erstellt, da für jede GSM-Box eine andere Update-Datei erwartet werden kann. Diese kann dann über eine authentifizierte http-Seite rückgemeldet werden oder über eine passwortgeschützte FTP-Verbindung heruntergeladen werden.

5. Energieoptimierugstechniken

Der Energieverbrauch ohne zusätzliche Maßnahmen ist für eine batteriebetriebene Anwendung untragbar. Durch die verschiedenen Optimierungen, welche in den nächsten Unterpunkten beschrieben werden, wird vor allem versucht, den konstanten Energieverbrauch auf ein Minimum zu bringen. Dies sollte soweit gelingen, dass mit einem relativ kleinen Batteriepack ein Betrieb über mehrere Jahre möglich ist. Nur mit dieser Voraussetzung wäre ein solches System erst einsatzfähig und seine Betriebskosten tragbar.

5.1 Optimierung durch Näherungssensor

Das natürliche Verhalten der Benutzer, die beim Verwenden eines NFC Lesegerätes in jedem Fall mit einer Hand in die Nähe der Antenne des Lesegerätes kommen, lässt sich vorteilhaft für die Energieeffizienz nutzen.

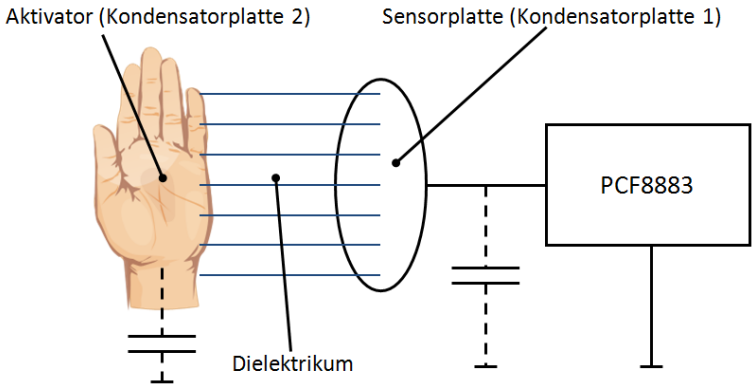


***Abbildung 32**: Allgemeiner Laboraufbau des Näherungssensors.*

Ohne Näherungssensor muss mit einem aktiven Magnetfeld ständig nach passiven Transpondern gesucht werden. Dieses Magnetfeld muss so stark sein, dass die Transponder mit genügend Energie versorgt werden, um auf die Anfrage des Lesegeräts zu antworten.

Energiesparender ist es durch kapazitive Näherungssensorik das Annähern einer Hand zu erkennen, und sodann weitere Komponenten einzuschalten. Während des „Sleep-Modus“ ist nur der Näherungssensor aktiv. Dieser überprüft ständig auf eine sich ändernde Kapazität an der Sensorplatte. Wenn man annimmt, dass die Sensorplatte die gleiche Fläche A wie eine Hand hat, so hängt die Kapazität mit dem Abstand d folgendermaßen zusammen:

$$C_H = \varepsilon_0 \cdot \varepsilon_R \cdot \frac{A}{d}. \qquad 5.1$$

ε_0 ist dabei die elektrische Feldkonstante und ε_R die relative Permittivität des Dielektrikums. Bei Luft ist diese gleich 1. Wie in Abbildung 33 dargestellt, steigt die Kapazität mit dem Annähern der Hand stark an.

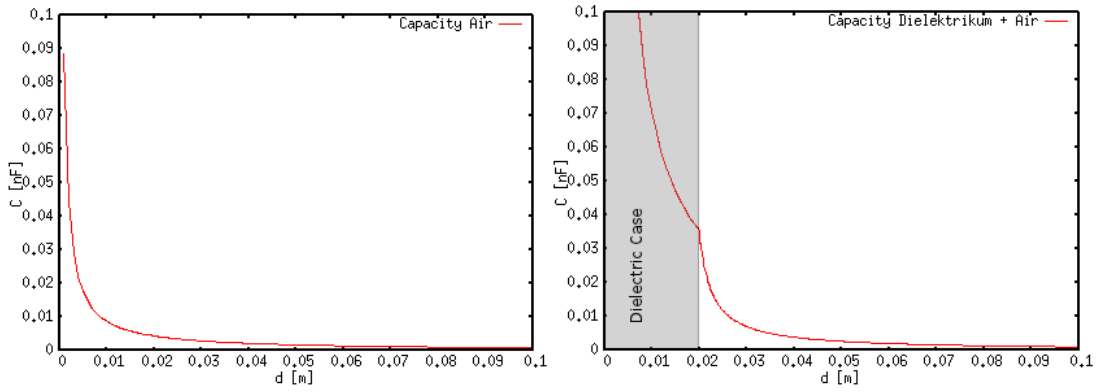


***Abbildung 33**: Vergleich der gemessenen Kapazität an der Sensorplatte mit und ohne dielektrischem Gehäuse als Funktion des Abstandes d der Hand. Die Sensitivität des Sensors wird durch das Gehäuse geschwächt.*

Da die Sensorplatte aber geschützt in einem Gehäuse angebracht werden soll, werden die Feldlinien materialabhängig nach außen geschwächt.

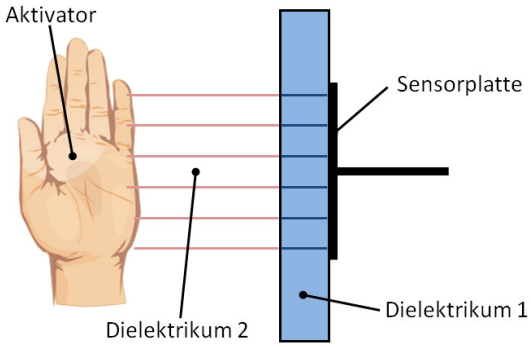


***Abbildung 34**: Falls die Sensorplatte innen in einem Gehäuse angebracht wird, so muss das Dielektrikum des Materials berücksichtigt werden. Dies kann durch eine Ersatzschaltung mit zwei Kondensatoren in Serie dargestellt werden.*

Dabei ist es wichtig, dass das Gehäuse aus einem dielektrischen Material besteht, damit durch bewegliche Ladungsträger die Durchlässigkeit des Feldes nicht verhindert wird. Die Gesamtkapazität, welche man an der Sensorplatte misst, ist

$$C_{GES} = \frac{1}{\frac{1}{C_H} + \frac{1}{C_D}}, \qquad 5.2$$

wobei C_D konstant ist, da sich die Dicke des Materials nicht ändert. Setzt man nun für C_D und C_H Formel 5.1 ein, so erhalten wir den Zusammenhang:

$$C_{GES} = \frac{\varepsilon_0 \cdot \varepsilon_R \cdot A}{\varepsilon_R \cdot d_H + d_D}. \qquad 5.3$$

Da die Sensibilität des Sensors durch die Änderung der Kapazität im Verhältnis zur Änderung des Abstandes der Hand zur Sensorplatte ausgedrückt werden kann, bilden wir die Ableitung

$$\frac{\partial C_H}{\partial d_H} = -\frac{\varepsilon_0 \cdot \varepsilon_R \cdot A}{{d_H}^2} \qquad 5.4$$

und

$$\frac{\partial C_{GES}}{\partial d_H} = -\frac{\varepsilon_0 \cdot {\varepsilon_R}^2 \cdot A}{(\varepsilon_R \cdot d_H + d_D)^2}. \qquad 5.5$$

Dabei sehen wir, dass die Steigung $\frac{\partial C_H}{\partial d_H} < \frac{\partial C_{GES}}{\partial d_H}$ von $0 < d_H < \infty$ gilt. Indem die Sensorplatte hinter einem dielektrischen Gehäuse angebracht wird, nimmt also die Sensibilität des Sensors ab.

Durch das Annähern der Hand ändert sich im Verhältnis die gemessene Kapazität an der Sensorplatte langsamer.

Eine weitere Herausforderung besteht noch darin, die Antenne des NFC Lesegerätes, die kapazitive Sensorplatte und das Gehäuse geometrisch so zu positionieren und zu gestalten, dass die Lesefähigkeit des NFC Lesegerätes nicht beeinflusst wird und die Sensibilität des Näherungssensors genügend groß ist. Damit muss beim Näherungssensor vor allem darauf geachtet werden, dass die Feldlinien immer den kürzesten Weg zur Masse suchen. Dafür muss die Sensorplatte bestmöglich vom Schlossgehäuse, welches aus leitfähigem Material besteht, isoliert sein. Die Sensorplatte muss in der Nähe der Antenne angebracht werden, um beim Annähern des Targets an die Antenne auch sicher zu schalten. Die Antenne darf dabei nicht beeinflusst werden. So kann die Sensorplatte nicht mitten in der Antenne platziert werden. Wäre das der Fall würde das gesamte magnetische Feld absorbiert werden und nicht genügend Leistung für die Targets transportiert werden. Eine sinnvolle Lösung hierfür ist eine ringförmige Platzierung der Sensorplatte um die Antenne, wie in Abbildung 35 dargestellt ist.

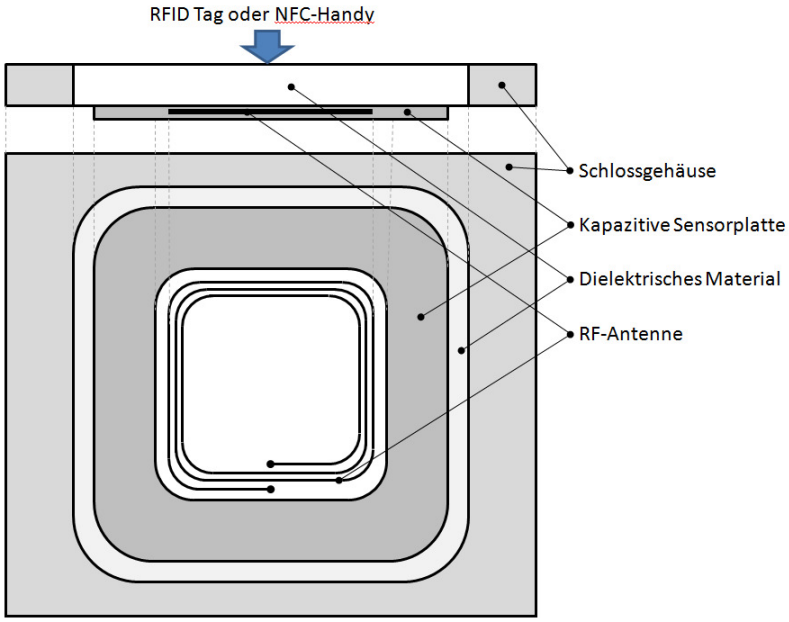


***Abbildung 35**: Anordnung der NFC-Antenne und der kapazitiven Sensorplatte, damit die Kommunikation nicht gestört wird und die Sensorplatte sicher schaltet, wenn ein Target in der Nähe der NFC-Antenne gehalten wird.*

Damit Schmutz, Feuchtigkeit, Alterung oder leichte Beschädigung der Sensorplatte keinen Einfluss auf die Funktionalität des Näherungssensors haben, sollte dieser vor allem überprüfen, in welcher Richtung sich die Kapazität ändert und nicht den absoluten Wert der Kapazitätsänderung betrachten.

Durch einen Regelkreis können dann sehr langsame Änderungen der Kapazität ausgeglichen werden. Mit einer UP/DOWN Counter Logik kann mitgezählt werden, wie lange sich die Kapazität in einer Richtung ändern muss, bis geschaltet wird. Beim PCF8883 von NXP wird immer bis 63 gezählt und dann geschalten. Dies ergibt einen digitalen Filter, der kurze Störungen und schnelle Kapazitätsänderungen unterdrückt. Eine mögliche Schaltung ist in Abbildung 36 dargestellt [NXPPCF].

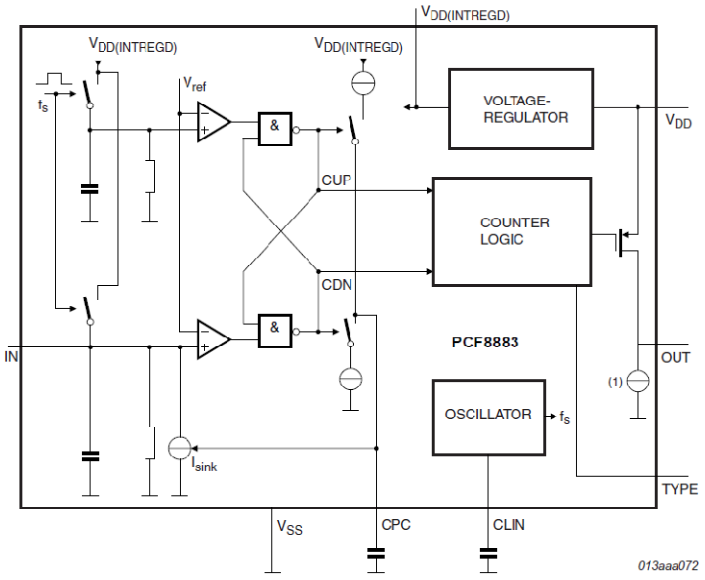


***Abbildung 36**: Schematischer innerer Aufbau des kapazitiven Näherungsschalters.*

5.2 Optimierung im Funkteil

In diesem Kapitel wird analysiert, wie weit die Feldstärke reduziert werden kann und welche Antennengröße optimal ist.

Zur Vereinfachung der Analyse werden kreisförmige Antennen angenommen. Laut NFC Standardisierung muss das Feld die magnetische Feldstärke $H_{threshold} = 0{,}1875 \frac{A}{m}$ am Target überschreiten, damit die Versorgung sichergestellt wird.

Eine von Strom durchflossene Spule erzeugt eine magnetische Feldstärke H, die den Verlauf

$$H(x) = \frac{I \cdot N \cdot R^2}{2 \cdot \sqrt{(R^2 + x^2)^3}} \qquad 5.6$$

mit dem Abstand x hat. R stellt den Antennendurchmesser dar, N die Windungszahl und I den in der Spule des Lesegeräts fließenden Strom.

Untersucht man die Feldstärke bei einem fixen Abstand x und variiert den Antennendurchmesser R, so erreicht die Feldstärke an einem Punkt ein Maximum. Dieser wird durch die Ableitung der Formel 5.7 nach R berechnet:

$$H'(R) = \frac{d}{dR} H(R) = \frac{2 \cdot I \cdot N \cdot R}{\sqrt{\left(R^2 + x^2\right)^3}} - \frac{3 \cdot I \cdot N \cdot R^3}{\left(R^2 + x^2\right) \cdot \sqrt{\left(R^2 + x^2\right)^3}} . \qquad 5.8$$

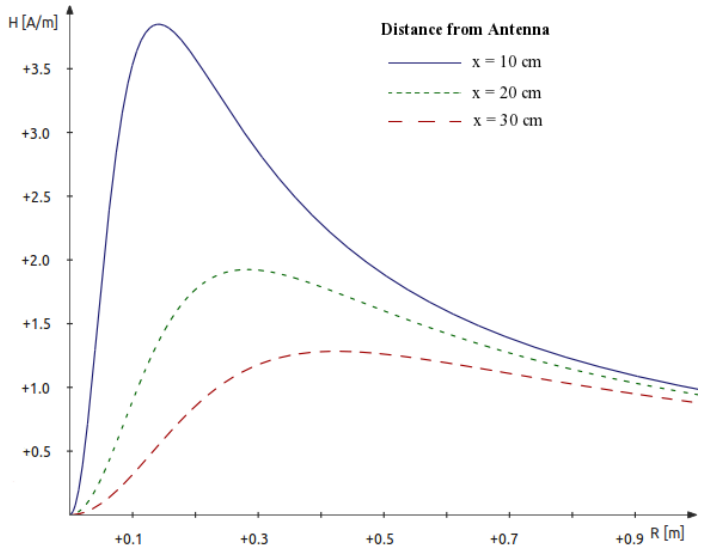


***Abbildung 37**: Magnetische Feldstärke in Abhängigkeit des Antennenradius R in einem fixen Abstand zur Antenne x = 10 cm, x = 20 cm und x = 30 cm. Die Feldstärke hat ein Maximum, wo der Radius der Antenne √2-Mal x ist.*

Die Nullstellen findet man bei $R_{1/2} = \pm x \cdot \sqrt{2}$. Damit muss idealerweise der Radius der Antenne des Lesegerätes das $\sqrt{2}$ fache des gewünschten Leseabstandes sein. Um Schwierigkeiten beim Lesen der Targets zu vermeiden, sollte eine Lesereichweite von einigen Zentimetern gewährleistet sein.

Dadurch kann der Strom I in der Lesegerätantenne bestimmt werden, welcher eine normgerechte Feldstärke am Transponder im maximalen Abstand erzeugt:

$$I(R) = \frac{H_{threshold} \cdot R \cdot \sqrt{\frac{27}{2}}}{N}. \qquad 5.9$$

Da bei der Standardfrequenz $f = 13{,}56\,MHz$ eine große Stromverdrängung aus dem Leiter bemerkbar ist, ist es wichtig, die Antennenwindung mit einer möglichst großen Oberfläche zu bauen und den Widerstand durch nicht zu viele Windungen klein zu halten. Üblicherweise werden in RF-ID Systemen 3 bis 10 Windungen verwendet.

Möchte man eine Ansprechreichweite von drei Zentimetern und nimmt man 5 Spulenwindungen an, so wird folgender Strom durch die Spule benötigt:

$$I(R) = \frac{0{,}1875\frac{A}{m} \cdot 0{,}03m \cdot \sqrt{\frac{27}{2}}}{5} = 413{,}351\mu A$$

Die Spannung, welche im Lesegerät an der Antenne angelegt werden muss, beträgt:

$$u_L = j\omega M \cdot i_T - j\omega L_L \cdot i_L - i_L \cdot R_L. \qquad 5.10$$

Der Kopplungsfaktor

$$k = \frac{M}{\sqrt{L_T \cdot L_L}} \qquad 5.11$$

ist dabei < 1 % und der Strom im Transponder i_T ist einige Größenordnungen kleiner als jener in der Spule des Lesegerätes i_L und somit vernachlässigbar klein. Die Gleichung vereinfacht sich somit zu:

$$u_L = j\omega L_L \cdot i_L - i_L \cdot R_L. \qquad 5.12$$

Die Induktivität der Antenne L_L lässt sich durch

$$L_L = N^2 \cdot \mu_0 \cdot R \cdot \ln\left(\frac{2R}{d}\right) \qquad 5.13$$

berechnen, wobei d der Durchmesser des verwendeten Leiters ist. Durch die hohe Frequenz in der Antennenspule macht sich der Widerstand R_L bemerkbar. Mit der äquivalenten Leitschichtdicke

$$\delta = \sqrt{\frac{2\rho}{\omega \cdot \mu}}, \tag{5.14}$$

welche die Dicke eines fiktiven Ersatzleiters darstellt, kann

$$R_L = \frac{\rho \cdot l}{A} = \frac{\rho \cdot l}{\left(\frac{\delta}{2}\right)^2 \pi} = \frac{2 \cdot l \cdot \omega \cdot \mu}{\pi} \tag{5.15}$$

berechnet werden und dadurch die ohmschen Verluste

$$P = R_L \cdot I^2(R) = \frac{l \cdot \omega \cdot \mu \cdot H_{threshold}^2 \cdot R^2 \cdot 27}{N^2 \cdot \pi}. \tag{5.16}$$

[FINKENZELLER2010]

5.3 Optimierung durch „Sleep-Modus“

Eine weitere Möglichkeit Energie zu sparen ist es, alle Bauteile möglichst im „Sleep-Modus“ zu halten. Durch regelmäßiges Aufwachen der Elektronik wird jedes Mal nach einem Target gesucht. Damit müssen mindestens 35 ms abgewartet werden, bis man sicherstellen kann, ob sich ein Target im RF-Feld befindet.

Sobald ein Benutzer ein Target in Lesereichweite hält, sollte dieser aber so schnell wie möglich erkannt werden. Die Zeit im „Sleep-Modus“ sollte nicht zu lange dauern. Der Benutzer darf nicht den Eindruck bekommen, dass das Schloss defekt ist. Daher sollte nach maximal einer Sekunde der Mikrocontroller durch einen Interrupt aufgeweckt werden, um wiederum nach einem Target zu suchen.

5.4 Vergleich der Optimierungstechniken

Der gesamte Energieverbrauch setzt sich aus einem konstanten Energiefluss zur Bereitstellung des Service plus einer bestimmten Energiemenge zusammen, die nur bei einer Schlossbenutzung verbraucht wird.

Damit ist die durchschnittlich verbrauchte Leistung

$$P_{TOT} = P_K + P_B(f) \quad 5.17$$

wobei P_K konstant ist und $P_B(f)$ davon abhängig ist, mit welcher Frequenz ein Schloss benutzt wird.

Ohne Energieoptimierung setzt sich P_K durch den Stromverbrauch aus den folgenden Bauteilen zusammen: NFC Chip PN532, Cortex Mikrocontroller STM32F103 und Speicher. Diese sind ohne Maßnahmen, wie unter Punkt 5.1 bis 5.3 beschrieben, 24 Stunden pro Tag im normalen Modus und haben somit einen Verbrauch von

$$P_K = P_{NFC} + P_{STM} + P_{MEM} . \quad 5.18$$

Bei jeder Schlossbenutzung muss der Riegel betätigt werden, was zu einem Verbrauch von

$$P_B(f) = E_{Riegelbetätigung} \cdot f \quad 5.19$$

führt. Hierbei ist $E_{Riegelbetätigung}$ jene Energie, die bei einer einzigen Betätigung des Riegels verbraucht wird.

Typische Werte der einzelnen Bauteile werden in Tabelle 2 aufgelistet.

Bauteil	***[N]ormal Operation***	***[L]ow-Power-Mode***
PN532 [P_{NFC}]	*330 mW*	*6,6 µW*
STM32F103 [P_{STM}]	*132 mW*	*33 µW*
ST M58BW016 [P_{MEM}]	*100 mW*	*198 µW*
Riegel [$E_{Riegelbetätigung}$]	*250 mJ*	*0 mJ*
Näherungssensor [$P_{VICINITY}$]	*9,9 µW*	*9,9 µW*

***Tabelle 2**: Stromverbrauch der einzelnen Bauteile im normalen Modus und im Low-Power-Modus.*

Um Energie zu sparen, besteht die Möglichkeit, die Bausteine so viel wie möglich im „Sleep-Modus“ zu halten, wie dies unter Punkt 5.3 beschrieben ist. Es braucht $t_{recognize} = 35\,ms$, um ein Target im Feld zu erkennen. Die gesamte Authentifizierungsprozedur kann in $t_{authenticate} = 125\,ms$ durchgeführt werden. Das Verstellen des Riegels für $t_{bolt} = 300\,ms$ bei einer Spannung von 5V und

einem Strom von 165 mA ist also mit dem Aufwenden der Energie $E_{Riegelbetätigung} = 250mJ$ sichergestellt. Damit berechnen wir

$$P_K = \frac{\left(P_{NFC,N} + P_{STM,N} + P_{MEM,N}\right) \cdot t_{recognize} + \left(P_{NFC,L} + P_{STM,L} + P_{MEM,L}\right) \cdot t_{sleep}}{t_{recognize} + t_{sleep}} \quad 5.20$$

und

$$P_B(f) = \left[\left(t_{authenticate} - t_{recognize} + t_{bolt}\right) \cdot \left(P_{NFC,N} + P_{STM,N} + P_{MEM,N}\right) + E_{Riegelbetätigung}\right] \cdot f \,. \quad 5.21$$

Hier entstehen vor allem durch den PN532-NFC-Chip immer noch große Verluste.

Durch die folgende Energieanalyse wird gezeigt, wie viel Energie durch das Einsetzten eines kapazitiven Näherungssensors eingespart werden kann. Wir gehen vom Idealfall aus, wobei der Funkteil des Schlosses erst dann eingeschaltet wird, wenn sich bereits ein Target im Lesebereich befindet. Dann wird es auch nur so lang eingeschaltet, bis ein einziger Lesezyklus aus dem Target abgeschlossen ist. Ist das Auslesen des Targets erfolgreich und der Zugriff des Schlüssels gewährt, so kann das Schloss geöffnet werden und die Elektronik kann wieder in den „Sleep-Modus“ gehen.

Die konstant verbrauchte Leistung P_K entspricht damit dem Verbrauch des kapazitiven Näherungssensors PCF8883 und die „Low-Power-Mode“-Leistung des Mikrocontrollers, um für Updates aufwachen zu können:

$$P_K = P_{PCF} + P_{STM,L} \,. \quad 5.22$$

Die Leistung P_B stellt sich aus allen anderen Komponenten zusammen:

$$P_B(f) = \left[\left(t_{authenticate} + t_{bolt} + t_{memlookup}\right) \cdot \left(P_{NFC,N} + P_{STM,N} + P_{MEM,N}\right) + E_{Riegelbetätigung}\right] \cdot f \,. \quad 5.23$$

Beim Vergleich der Optimierungstechniken sieht man, dass durch den Näherungssensor die konstant verbrauchte Leistung P_K zum Bereitstellen des Services am geringsten ist.

Bei der Benutzung des Schlosses kann noch weiter optimiert werden, in dem bestimmte Bauteile von der Spannungsversorgung komplett getrennt werden, sobald diese nicht mehr gebraucht werden. Der einzige Bauteil, der immer eingeschaltet sein muss, um die ganzen Vorgänge zu koordinieren, ist der Mikrocontroller. Der Näherungssensor führt das Aufwachen des Mikrocontrollers herbei, welcher seinerseits den PN532-Chips so lange ($t_{authenticate}$) versorgt, bis die Authentifizierung mit dem Target durchgeführt wurde. Dann kann der PN532-Chip wieder abgeschaltet werden und dafür der Flash-Speicherbaustein versorgt werden. Nachdem sichergestellt wurde, dass die ID des Targets im Speicher nicht enthalten ist ($t_{memlookup}$), wird der Speicherbaustein ebenfalls abgeschaltet und der

Riegel kann betätigt werden. Nachdem (t_{bolt}) der Riegel betätigt wurde, kann der Mikrocontroller wieder in den „Low-Power-Mode“ gehen. Bildlich wird der Vorgang in Abbildung 38 dargestellt.

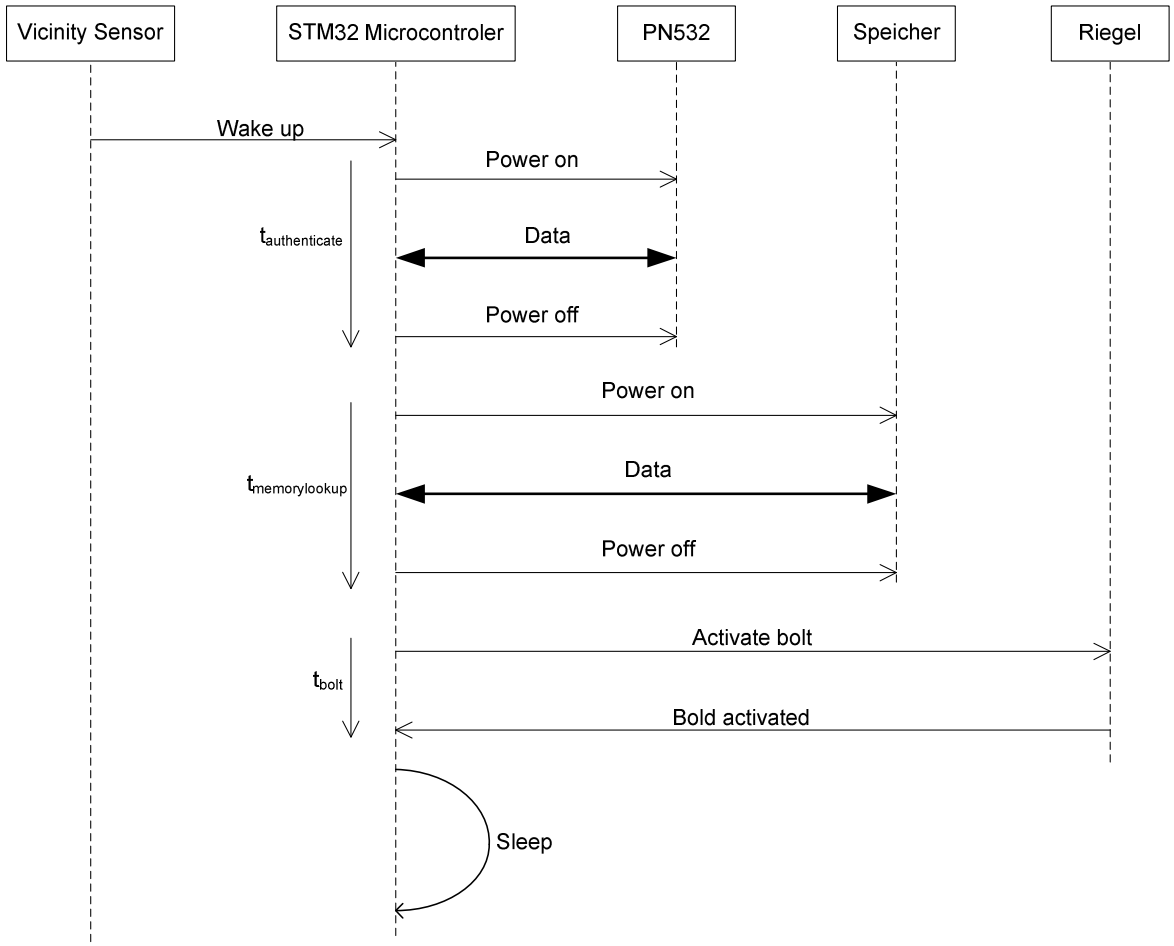


***Abbildung 38**:Bauteilaktivierung während einer Schlossbenutzung. Es wird versucht, jene Bauteile, welche für die aktuelle Aktion nicht benötigt werden, immer im Low-Power-Mode zu halten. Die Startupzeiten der Bauteile sind vernachlässigbar klein.*

Die Mindestzeiten vom Anlegen der Versorgungsspannung bis zum vollen Betrieb des Bauteils sind um Größenordnungen kleiner, als die Dauer der einzelnen Aufgaben, die durchzuführen sind. Damit können die „Startup“-Zeiten in der Energieberechnung vernachlässigt werden.

Die Maximaldauer der einzelnen Aufgaben ist in **Tabelle 3** zusammengefasst. In den angegebenen Zeiten ist der jeweilige Vorgang garantiert abgeschlossen.

Abkürzung	Beschreibung	Zeit
$t_{recognize}$	Zeit, bis ein Target im Feld vom Mikrocontroller erkannt wird und die ID ausgelesen wurde.	< 35 ms
$t_{authenticate}$	Zeit, bis ein Target erkannt und authentifiziert wurde.	< 125 ms
t_{bolt}	Zeit, bis der Schlossriegel erfolgreich betätigt wurde.	< 300 ms
$t_{memlookup}$	Zeit, bis sichergestellt werden kann, dass die Target-ID im Speicher nicht enthalten ist.	< 50 ms

***Tabelle 3**: Maximalzeiten zum durchführen einer bestimmten Aufgabe im NFC-Schloss*

Insgesamt haben wir also für die konstant verbrauchte Leistung

$$P_K = P_{PCF} + P_{STM,L} \tag{5.24}$$

und die optimierte Leistung P_B :

$$\mathrm{P_B}(f) = \left[\left(t_{authenticate} + t_{bolt} + t_{memlookup}\right) \cdot P_{STM,N} + t_{authenticate} \cdot P_{NFC,N} + t_{memlookup} \cdot P_{MEM,N} + E_{Riegelbetätigung}\right] \cdot f \tag{5.25}$$

6. Ergebnisse und Einschätzung der Batterielebensdauer

Stellen wir nun die verschiedenen Optimierungstechniken hinsichtlich der gesamten verbrauchten Leistung

$$P_{TOT} = P_K + P_B(f) \tag{6.1}$$

gegenüber, so sehen wir bei der letzten mit koordinierter Bauteilenergieversorgung in der Abbildung 39d die optimalste Lösung. In erster Linie werden hier nur die Schlossbenutzungen analysiert.

Alle Bauteile sind für 3,3 V ausgelegt. Der Näherungssensor PCF8883 kann allerdings mit einer Betriebsspannung von 3 V bis 9 V betrieben werden. Damit kann der Näherungssensor direkt von der Batterie versorgt werden, während allen anderen Bauteilen ein 3,3 V-Linearregler vorgeschaltet werden muss.

Verwendet man zum Betreiben des Schlosses eine 3,6 V- LiSoCl2 Primärzelle, so hat diese typischerweise 6000 mAh. Das ergibt eine gespeicherte Energie $E_{battery}$ von 77760 Joule. Damit kann man die Batterielebensdauer mit

$$t_{battery} = \frac{E_{battery}}{P_{TOT}} = \frac{E_{battery}}{P_K + P_B(f)} \tag{6.2}$$

einschätzen.

Dies ergibt bei der optimalsten Lösung mit Näherungssensor und Energieversorgungskoordinierung der Bausteine (wie unter Abbildung 40 dargestellt):

- Bei 0 Benützungen pro Tag: 20979 Tage (57,4 Jahre)
- Bei 100 Benützungen pro Tag: 1963 Tage (5,4 Jahre)
- Bei 200 Benützungen pro Tag: 1029 Tage (2,8 Jahre)

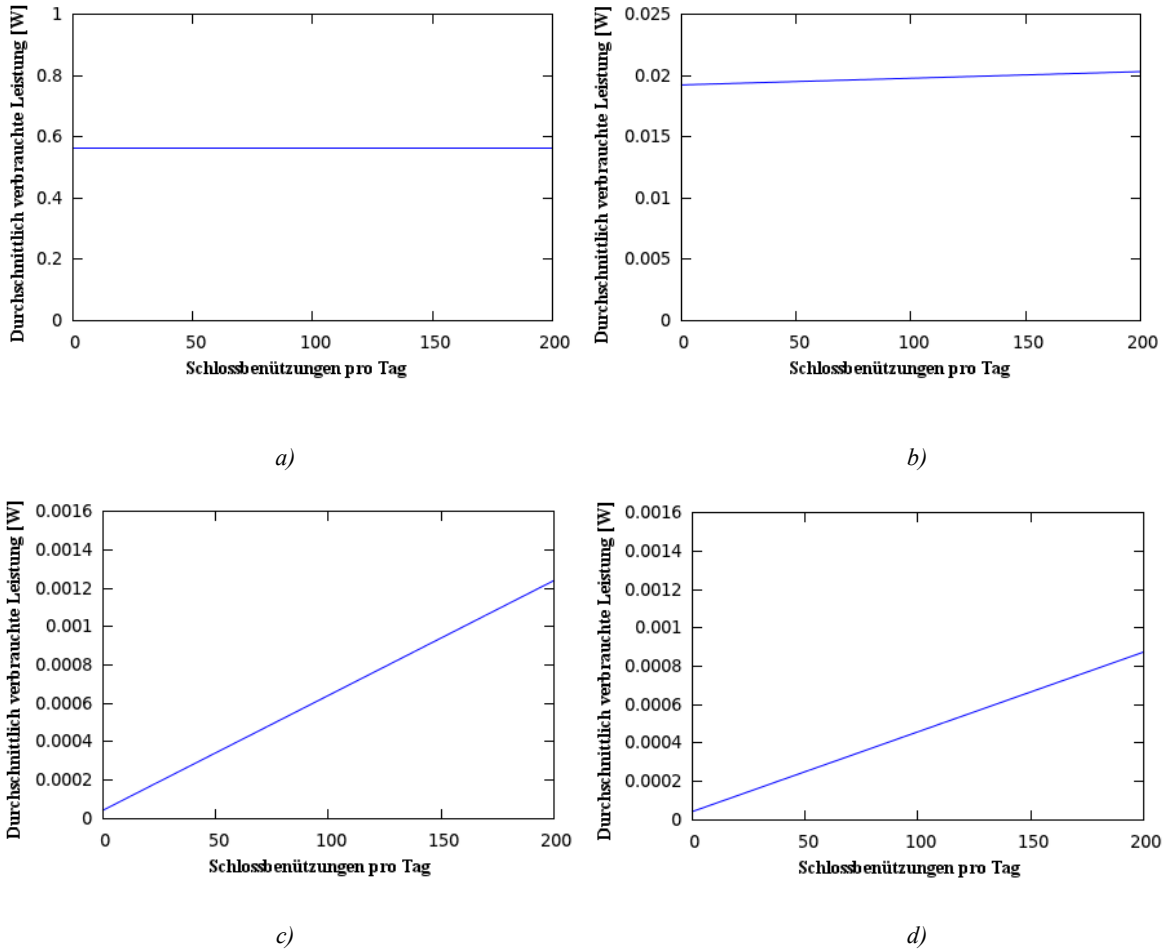


***Abbildung 39**: Vergleich der Energiesparmaßnahmen. Durchschnittlich verbrauchte Leistung eines NFC-Schlosses in Abhängigkeit der Schlossbenutzungen pro Tag.*
a) Ohne Energiesparmaßnahmen. Alle Bauteile sind immer im normalen Modus.
b) Optimierung durch „Sleep-Modus". Durch regelmäßiges Aufwachen wird nach Targets gesucht. Den Großteil der Zeit befindet sich das Schloss im „Sleep-Modus", wobei eine Verzögerung bei der Benutzung des Schlosses zu erwarten ist.
c) Optimierung durch Näherungssensor. Die Bauteile werden erst dann aktiviert, wenn der Näherungssensor ein annäherndes Objekt entdeckt. Damit wird die konstant verbrauchte Leistung auf ein Minimum reduziert.
d) Optimierung durch Näherungssensor und Koordinierung der Energiezufuhr der Bauteile. Es wird die Versorgung immer nur für jene Bauteile aktiviert, welche gerade gebraucht werden.

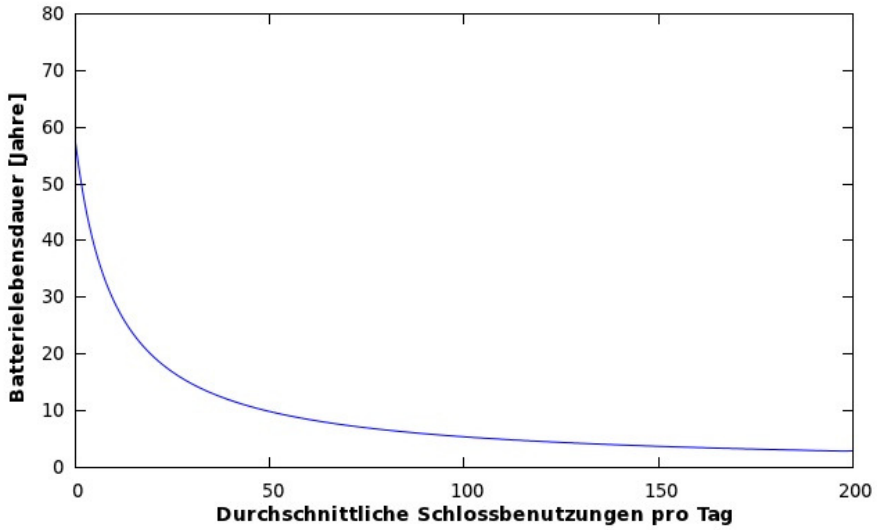


***Abbildung 40**: Geschätzte Batterielebensdauer für eine 3,6 V - LiSoCl2 Primärzelle mit 6 Ah in Abhängigkeit der durchschnittlichen NFC-Schlossbenutzungen pro Tag. Es wird eine ideale Batterie angenommen.*

6.1.1 Berücksichtigen der Selbstentladung

Bei einer Betriebsdauer über mehrere Jahre muss die Selbstentladung der Batterie berücksichtigt werden. Die Selbstentladung ist stark vom Batterietyp und der Umgebungstemperatur abhängig. Als Richtwert werden 2% Selbstentladung pro Jahr angenommen.

Daraus wird die Batterielebensdauerberechnung zu

$$\mathrm{t}_{\mathrm{battery}}(f) = \frac{E_{battery}}{\mathrm{P_K} + \mathrm{P_B}(f) + \frac{0.02 \cdot E_{battery}}{3600 \cdot 24 \cdot 365}} \qquad 6.3$$

angepasst.

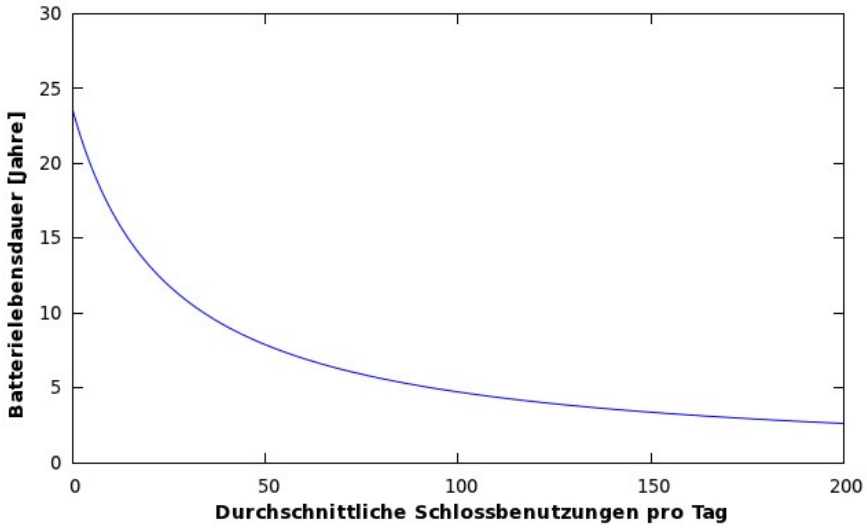


***Abbildung 41**: Geschätzte Batterielebensdauer für eine 3,6 V - LiSoCl2 Primärzelle mit 6 Ah in Abhängigkeit der durchschnittlichen NFC-Schlossbenutzungen pro Tag. Es wird für die Batterieselbstentladung ein Richtwert von 2%/Jahr angenommen.*

6.1.2 Berücksichtigen der Updates

Viel Energie wird auch benötigt, um die Updates durchzuführen. Dabei muss der Funkteil eingeschaltet, alle Daten übermittelt und schlussendlich die Schlüsselliste im Speicher aktualisiert werden. Der Stromverbrauch ist davon abhängig, wie viel Daten zwischen Schloss und GSM-Box übermittelt werden müssen. Je öfter ein Schloss benützt wird, desto mehr Log Daten werden gesammelt und sind somit zu übermitteln. Je mehr Schlüssel bereits im Schlossspeicher enthalten sind, desto mehr Rechenleistung ist aufzuwenden, um neue Schlüssel im Speicher hinzuzufügen oder zu entfernen. Diese müssen wie unter 3.4 sortiert werden.

Die durchschnittlich verbrauchte Leistung P_u für die Updatefunktionalität ist

$$P_u(f, f_u, M, N) = P_{FUNK,L} + f_u \cdot \left[E_{funkinit} + E_{datasort}(N, M)\right] + \frac{E_{datasend}(f) + E_{datareceive}(M)}{24 \cdot 60 \cdot 60s}. \qquad 6.4$$

Dabei ist f die Schlossbenutzungsfrequenz, f_u die Updatefrequenz, M die Anzahl der Schlüssel zum Updaten und N die Anzahl der Schlüssel, welche schon im Speicher des Schlosses gespeichert sind.

Abkürzung	Beschreibung	Richtwert
M	Durchschnittliche Anzahl der Schlüssel zum Updaten pro Tag.	10
N	Im Schlossspeicher gespeicherte Schlüssel.	1000
$E_{funkinit}$	Energie zum Initiieren einer Funkverbindung zwischen NFC-Schloss und GSM-Box. Benötigt werden 2 mA für 2 ms bei 3 V. Angenommen wird, dass GSM-Box und Schloss zeitlich synchronisiert sind und zugleich aufwachen.	12 µJ
$E_{TX,Byte}$	Energie zum Senden eines Bytes. Durchschnittlich verbraucht der Funkteil beim Senden 14 mA bei 3 V. Bei einer Datenrate von 100 kbps dauert das Senden eines Bytes 80 µs.	3,36 µJ
$E_{RX,Byte}$	Energie zum Empfangen eines Bytes. Beim Empfang verbraucht der Funkteil durchschnittlich 12 mA bei 3 V. Die Datenrate ist auch beim Empfang 100 kbps.	2,88 µJ
$t_{datasort}(N, M)$	Zeit, welche zum Sortieren der Schlüssel im Speicher benötigt wird. Diese Zeit ist stark davon abhängig, wie viele Schlüssel bereits im Speicher enthalten sind. Nimmt man den Richtwert für N mit 1000 Schlüsseln an, so können die Datensätze innerhalb 300 ms sortiert werden.	300 ms
$P_{FUNK,L}$	Leistung, welche der Funkteil im „Sleep-Modus“ durchschnittlich verbraucht: 100 nA bei 3V	300 nW

Wie unter Kapitel 3.4 beschrieben, werden die Daten vom Schloss zur GSM-Box in Form einer XML-Datei übertragen. Vor dem Übertragen ist es hier sinnvoll, die Beginn- und Schlusszeichen (zwischen spitzen Klammern < > bzw. </ >) möglichst kurz zu halten, um die Datenmenge zu minimieren. Ein Buchstabe ist für die Menge an benötigten Befehlen genug.

Folgende Daten müssen übertragen werden:

- Konstante Datenmenge: idLock (8 Byte), installedUpdateId (8 Byte), batteryOk (1 Byte), lockOk (1 Byte) plus die Einkapslung der einzelnen Elemente (5 Mal je 3 und 4 Byte = 35 Byte). Insgesamt ergibt dies 53 Byte.
- Schlossbenutzungsfrequenzabhängige Datenmenge: für jede Schlossbenutzung der benützte Schlüssel (8 Byte) und die Uhrzeit (10 Byte). Die Einkapslung der Daten verbraucht hier wiederum je 3 Mal 3 und 4 Byte (21 Byte). Insgesamt ergibt dies 39 Byte pro Schlossbenutzung.

Zum Empfangen der Daten über Funk gilt die gleiche Überlegung. Hier kommen pro Schlüssel, der zu übertragen ist Daten im Ausmaß von 8 Byte Schlüssel-ID und 7 Byte Einkapslung zusammen.

Daraus erhalten wir die Teilformeln:

$$E_{datasend}(f) = [53Byte + f \cdot 39Byte] \cdot E_{TX,Byte}$$

und

$$E_{datareceive}(M) = 15Byte \cdot M \cdot E_{RX,Byte}.$$

$E_{TX,Byte}$ ist jene Energie, welche zum Übertragen eines Bytes vom Schloss zur GSM-Box notwendig ist und $E_{RX,Byte}$ jene Energie, welche zum Empfangen der Schlüsseldaten von der GSM-Box aufzuwenden ist.

Zum Sortieren der Schlüssel im Flash-Speicher sind N Lesezyklen und $N+M$ Schreibzyklen notwendig. Zusätzlich kommen noch die $M \cdot \log_2(N)$ Speicherlesezyklen zum Identifizieren des Indexes im Speicher, wo die neuen Schlüssel einzufügen bzw. zu löschen sind, hinzu. Für die gesamte Speicherreorganisation wird dabei die Zeit $t_{datasort}(N,M)$ benötigt. Daraus folgt

$$E_{datasort}(N,M) = t_{datasort}(N,M) \cdot [P_{STM,N} + P_{MEM,N}]. \qquad 6.5$$

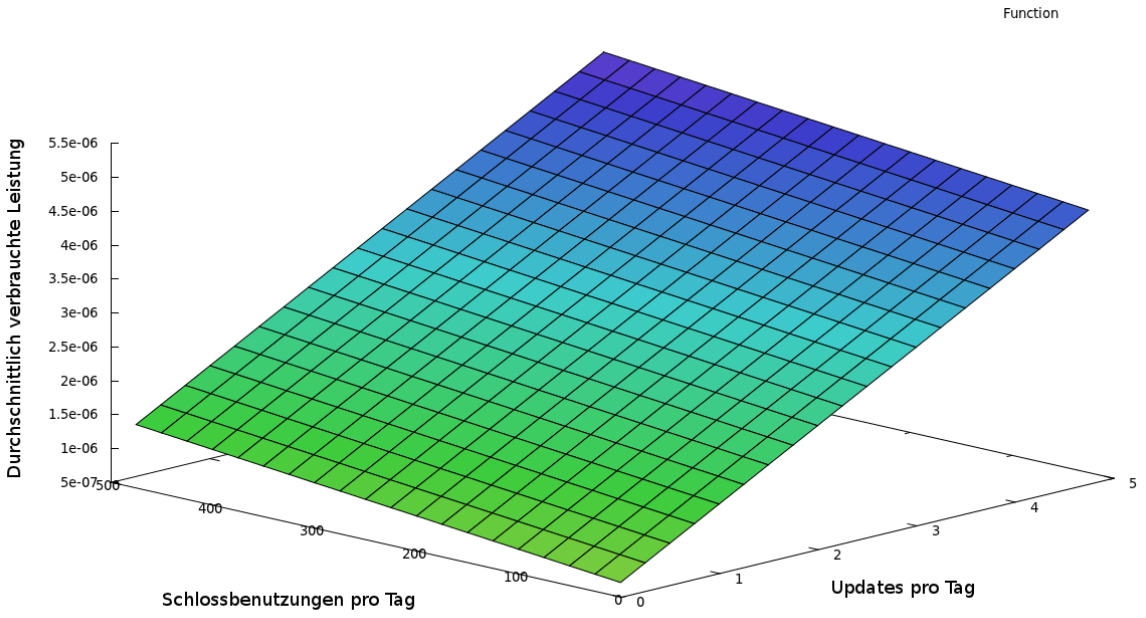


***Abbildung 42**: Durchschnittlich verbrauchte Leistung P_u zum Übertragen der Logs und zum Updaten der Schlüsseldatenbank im NFC-Schloss.*

Wird nun der Verbrauch der Leistung P_u in die Batterielebensdauer einbezogen, so erhält man:

$$\mathrm{t}_{\mathrm{battery}}(f, f_u) = \frac{E_{battery}}{\mathrm{P_K} + \mathrm{P_B}(f) + \frac{0.02 \cdot E_{battery}}{3600 \cdot 24 \cdot 365} + P_u(f, f_u, M, N)}. \qquad 6.6$$

Die Lebensdauer der Batterie wird in Abhängigkeit der Schlossbenutzungs- und Updatefrequenz mit den Richtwerten in Abbildung 43 dargestellt.

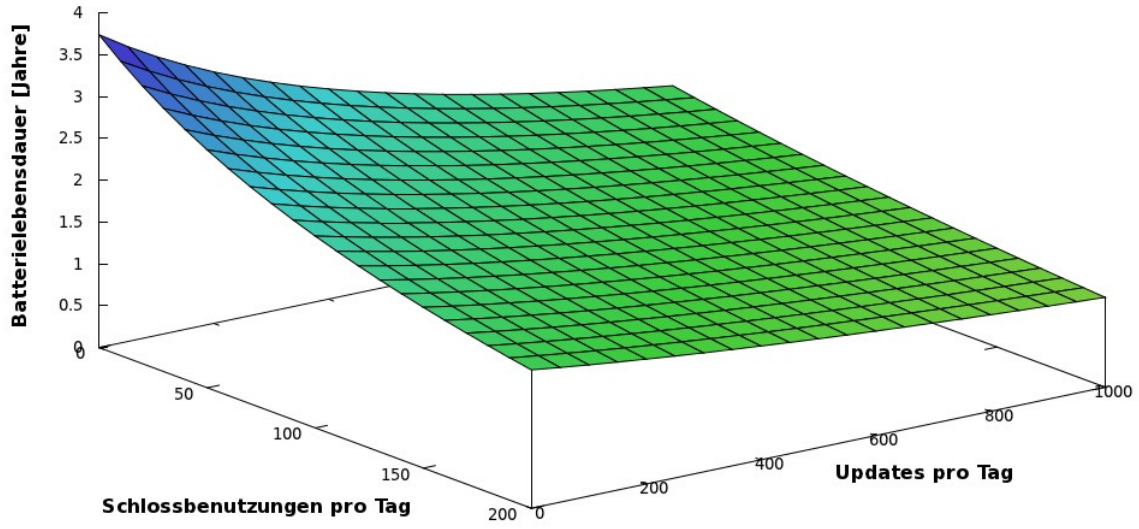


***Abbildung 43**: Geschätzte Batterielebensdauer einer 3,6 V - LiSoCl2 Primärzelle mit 6 Ah in Abhängigkeit der Schlossbenutzungen und der Updatefrequenz.*

Damit ist die Machbarkeit eines solchen Schlosssystems gezeigt. Die Voraussetzung, dass die Schlösser wartungsfrei über mehrere Jahre funktionieren sollen, ist damit gegeben. Somit ist auch für entlegene Orte ein gesundes NFC-Ökosystem garantiert.

7. Ausblick und weiterführende Arbeit

Nachdem die Machbarkeit eines NFC-Schlosssystems gezeigt wurde und als Gesamtkonzept ausgearbeitet wurde, bedarf es zur kompletten Fertigstellung noch einiges an Arbeit. Vor allem im Bereich der Implementierung.

Das NFC-Schloss muss noch an den bestehenden Funkteil angeschlossen werden und das Herunterladen der Updates über diesen ausprogrammiert werden. Als Nächstes sollte ein Prototyp des Schlosses entworfen werden. Mit den Bauteilen und der Beschaltung sollte, wie unter Punkt 4 beschrieben ist, ein Board-Layout entworfen werden. Außerdem sollte der mechanische Aufbau des Schlosses robust entworfen werden und so aufgebaut sein, dass die NFC-Feldlinien nicht durch nicht-dielektrische Materialien zu stark abgeschwächt werden.

Die GSM-Box wurde in dieser Arbeit nicht näher betrachtet. Diese sollte mit dem bestehenden energiesparenden Funkteil und dem GSM-Modul bestückt werden. Auch hier sollten ein Prototyp und die Software für das Koordinieren der Updates der verschiedenen Schlösser erstellt werden.

Bei der Serversoftware bedarf es eigentlich nur mehr einiger Anpassungen an der Benutzeroberfläche, ansonsten ist diese für das Verwalten des Schlosssystems und das Bereitstellen der Updates bereit.

Damit wäre das NFC-Schlosssystem bereit, um getestet und eingesetzt zu werden.

Literatur

[LANGER2010] Josef Langer und Michael Roland: *Anwendungen und Technik von Near Field Communication (NFC),* Springer Verlang Berlin Heidelberg, ISBN 978-3-642-05496-9, 2010

[ECMA352] Standard ECMA International: *Near Field Communication Interface and Protocol – 2 (NFCIP-2)*, 2nd Edition, Juni 2010, publiziert unter: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-352.pdf

[ECMA340] Standard ECMA International: *Near Field Communication Interface and Protocol (NFCIP-1)*, 2nd Edition, Dezember 2004, publiziert unter: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-340.pdf

[NFCFORUM-TS-NDEF] nfc-forum.org: *NFC Data Exchange Format (NDEF) Technical Specification,* 2011, publiziert unter: http://www.nfc-forum.org/specs/spec_license

[IEEEMAN2008] G. Madlmayr, J. Langer, J. Scharinger, NFC Res. Lab. Hagenberg, Univ. of Appl. Sci. of Upper Austria, Hagenberg im Muhlkreis: *Managing an NFC Ecosystem*, IEEE, ISBN: 978-0-7695-3260-8, Juli 2008

[LAMARCA2007] Anthony LaMarca, Marc Langheinrich und Khai N. Truong: *Pervasive Computing*, Springer Verlang Berlin Heidelberg, ISSN 0302-9743, 2007

[KERN2007] Christian Kern: *Anwendungen von RFID-Systemen*, *),* Springer Verlang Berlin Heidelberg, ISBN-10 3-540-44477-7, 2007

[FINKENZELLER2010] Klaus Finkenzeller: *RFID Handbook*, Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication, Third Edition, John Wiley & Sons, Ltd. ISBN 978-0-470-69506-7, 2010

[FINKENZELLER2002] Klaus Finkenzeller: *RFID Handbuch*, Grundlagen und praktische Anwendungen induktiver Funkanlagen, Transponder und kontaktloser Chipkarten, Carl Hasner Verlag München Wien, ISBN 3-446-22071-2, 2002

[NFCFORUM-TS-LLCP] nfc-forum.org: *Logical Link Control Protocol Technical Specification 1.1*, 2011, publiziert unter: http://www.nfc-forum.org/specs/spec_license

[FRA07] Christian Franzen: *Die Geschichte der RFID-Technologie*, Juni 2007, publiziert unter: http://home.arcor.de/ivenae/files/geschichte_der_rfid_technologie.pdf

[LANDT05] Jeremy Landt: *The history of RFID*, IEEE 0278-6648, November 2005, publiziert unter: http://autoid.mit.edu/pickup/RFID_Papers/008.pdf

[SUTH05] Max Sutherland: *Wake Up Call! The Future of RFID is Dawning*, January 2005, publiziert unter: http://www.sutherlandsurvey.com/Columns_Papers/Wake%20up%20call!%20The%20future%20of%20RFID%20is%20dawning.pdf

[JANNASCH04] Uta Jannasch und Dr. Sarah Spielkermann: *RFID Technologie im Einzelhandel der Zukunft: Datenentstehung, Marketing Potentiale und Auswirkungen auf die Privatheit des Kunden*, April 2004, publiziert unter: http://interval.hu-berlin.de/downloads/rfid/kernprobleme_Security/WorkingPaper%20Jannasch.doc

[ONDRUS06] Jan Ondrus und Yves Pigneur: *An Assessment of NFC for Future Mobile Payment Systems*, Department of Information Systems, University of Lausanne, publiziert unter: http://www.janondrus.com/wp-content/uploads/2008/05/icmb07.pdf

[SUHR04] Jan Suhr, Malwina Prokopczyk und Frank Reimann: *Datenschutz und RFID-Technik*, Technische Universität Berlin, Februar 2004, publiziert unter: http://ig.cs.tu-berlin.de/lehre/w2003/ir1/uebref/SuhrEtAl-Gutachten-G4-022004.pdf

[ANDR07] Marco Andres: *Datenschutz bei RFID-Anwendungen*, FTK Forschungsinstitut für Telekommunikation Dortmund, September 2007, publiziert unter: http://www.bitkom.org/files/documents/DS_bei_RFID_Support_Center_NRW_2007.pdf

[ENGELH06] Corinna Engelhardt-Nowitzki und Elisabeth Lackner: *Chargenverfolgung, Möglichkeiten, Grenzen und Anwendungsgebiete*, Deutscher Universitäts-Verlag, ISBN 978-3-8350-0639-3, 2006

[SECURITY] Ernst Haselsteiner und Klemens Breitfuß: *Security in Near Field Communication (NFC), Strenghts and Weaknesses*, Philips Semiconductors, publiziert unter: http://events.iaik.tugraz.at/RFIDSec06/Program/papers/002%20-%20Security%20in%20NFC.pdf

[SCHOBLICK05] Robert und Gabriele Schoblick: *Radio Frequency Identification, Grundlagen – Eingeführte Systeme – Einsatzbereiche – Datenschutz – Praktische Anwendungsbeispiele*, Franzis Verlag Poing, ISBN 3-7723-5920-5, 2005

[DOBKIN08] Daniel M. Dobkin: *The RF in RFID, Passive UHF RFID in Practice*, Elsevier Inc. Verlag, ISBN 978-0-7506-8209-1, 2008

[WANG09] Kai Wang: *Security Issues in RFID*, Research Institute of Information Technology, Tsinghua University, Beijing, China, 2009

[HUNT07] V. Daniel Hunt, Albert Puglia und Mike Puglia: RFID, a guide to radio frequency identification, Technology Research Corporation, Verlag John Wiley & Sons, Inc., ISBN 978-0-470-10764-5, 2007

[OLIMEXSTM08] Olimex Ltd: *STM-P103 development board, User Manual*, Rev.A, April 2008, publiziert unter: http://olimex.com/dev/pdf/ARM/ST/STM32-P103.pdf

[NXPPN532] Philips Semiconductors: *PN532/C1, NFC controller data sheet*, Rev. 1.2, März 2011, publiziert unter: http://www.nxp.com/documents/short_data_sheet/120112.pdf

[NXPPCF] NXP Semiconductors: *AN10832, PCF8883 - capacitive proximity switch with auto-calibration*, Rev. 03, September 2010, publiziert unter: http://ics.nxp.com/support/documents/interface/pdf/an10832.pdf

[OPENOCD08] Oliver Spencer: *Open On-Chip Debugger (openocd)*, Edition 1.0, April 2008, publiziert unter: http://www.zylin.com/zy1000/openocd.pdf

[ECLIPSEOPENOCD] Olimex Ltd, *Using OpenOCD server and Eclipse CDT IDE*, Dezember 2010, publiziert unter: http://olimex.com/dev/soft/arm/JTAG/Manual_ECLIPSE_from%20sample%20project.pdf

[YAGARTO] Michael Fischer: *GDB-Server, Yagarto GNU ARM toolchain, Eclipse Integrated Development Environment*, 2010, publiziert unter: http://www.yagarto.de/howto/yagarto2/index.html

[NXPPN532UM]NXP Semiconductors: *UM0701-02, PN532 User Manual*, Rev. 02, 2007, publiziert unter: http://www.nxp.com/documents/user_manual/141520.pdf

[CMSISHAL] ARM Ltd.: CMSIS - *Cortex Microcontroller Software Interface Standard*, 2011, publiziert unter: http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php