Hieronymus, Martin; Splettstößer, Anika; Schlumberger, Emil

Working Paper

Near field communication mit Arduino

Arbeitspapiere der Nordakademie, No. 2012-09

Provided in Cooperation with:
Nordakademie - Hochschule der Wirtschaft, Elmshorn

Suggested Citation: Hieronymus, Martin; Splettstößer, Anika; Schlumberger, Emil (2012) : Near field communication mit Arduino, Arbeitspapiere der Nordakademie, No. 2012-09, Nordakademie - Hochschule der Wirtschaft, Elmshorn

This Version is available at:
https://hdl.handle.net/10419/68242

ARBEITSPAPIERE DER NORDAKADEMIE

ISSN 1860-0360

Nr. 2012-09

Near Field Communication mit Arduino

Martin Hieronymus, Anika Splettstößer, Emil Schlumberger

Dezember 2012

Eine elektronische Version dieses Arbeitspapiers ist verfügbar unter:
http://www.nordakademie.de/arbeitspapier.html

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT
THEORIE PRAXIS

Köllner Chaussee 11
25337 Elmshorn
http://www.nordakademie.de

Nordakademie Elmshorn

Hochschule der Wirtschaft

Near Field Communication mit Arduino

Martin Hieronymus

Anika Splettstößer

Emil Schlumberger

07. November 2012

Inhaltsverzeichnis

Inhaltsverzeichnis I

Abkürzungsverzeichnis II

Abbildungsverzeichnis III

Tabellenvezeichnis III

I. Einleitung 1

A) Theoretischer Teil 2

1. NFC 2

2. Technische Grundlagen (Protokolle) 4

3. Stand der Technik 7

3.1 Hardware 7

3.2 Software 8

4. Anwendungsmöglichkeiten 10

5. Sicherheit 11

6. Verbreitung 12

B) Praktischer Teil 14

7. Projektverlauf 14

7.1 Kompatibilität der Libraries mit den NFC-Shields 14

7.2 Apps zum Auslesen und Beschreiben von Tags 15

7.3 Auslesen eines vom Smartphone beschriebenen Tags mit dem NFC-Shield 16

7.4 Card-Emulation Mode des NFC-Shields 16

7.5 Card-Emulation Mode des Smartphones 16

7.6 Peer-to-Peer Kommunikation 16

8. Schwierigkeiten und Lösungsansätze 17

9. Ergebnis 18

10. Programmierung mit Arduino 18

II. Fazit 20

Anhang III

Quellenverzeichnis VI

Abkürzungsverzeichnis

NFC	Near Field Communication
NDEF	NFC Data Exchange Format
NFCIP	Near Field Communication Interface and Protocol
JIS	Japanese Industrial Standard
MB	Megabyte
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
ID	Identifier
LSB	Least Significant Byte
MISO	Master in, Slave out
MOSI	Master out, Slave in
MSB	Most Significant Byte
UID	Unique Identifier
RFID	Radio-Frequency Identification
RTD	Record Type Definition
SCK	Serial Clock
SPI	Serial Peripheral Interface
SS	Slave Select
WLAN	Wireless Local Area Network

Abbildungsverzeichnis

Abbildung 1: Betriebsarten NFC fähiger Geräte 2
Abbildung 2: Vergleich verschiedener Technologien zur Datenübertragung 3
Abbildung 3: Vergleich verschiedener Technologien zur Datenübertragung 4
Abbildung 4: Speicherarchitektur Tag 5
Abbildung 5: Aufbau Block 3 (Sector Trailer) 5
Abbildung 6: Beispiel NDEF Nachricht 5
Abbildung 7: Erläuterung TLV 6
Abbildung 8: Uri Identifier 6
Abbildung 9: Entschlüsselte NDEF Nachricht 6
Abbildung 10: Arduino Duemilanove mit Atmega 328 III
Abbildung 11: Adafruit PN532 NFC/RFID Controller Shield III
Abbildung 12: Seeed Studio NFC Shield V1.0 IV
Abbildung 13: Tag (MIFARE Classic 1k) IV
Abbildung 14: Ausschnitt aus „nfc_beschrieben_lesen_1.pdf“ V
Abbildung 15: Ausschnitt aus "nfc_beschrieben_lesen_8.pdf" V

Tabellenverzeichnis

Tabelle 1: NFC-Tag Definitionen und Unterschiede 7
Tabelle 2: Übersicht verwendete NFC Apps für Android Smartphones 8
Tabelle 3: Beteiligte bei der bargeldlosen Bezahlung mit Beispielen 12
Tabelle 4: Kompatibilität Arduino, NFC-Shields, Libraries 15
Tabelle 5: Verwendete Key A 17

I. Einleitung

Das vorliegende Arbeitspapier beschäftigt sich mit dem Themenkomplex Near Field Communication (NFC). Diese Technologie ist viel diskutiert. Ihr wird seit Jahren vorausgesagt, dass sie die Anwendungsmöglichkeiten von Smartphones erweitern wird.[1] Zahlreicher Interessengruppen suchen Wege und Möglichkeiten der kommerziellen Nutzung dieser Technologie.[2] Aus Interesse an diesem vielfältigen Themenspektrum und dem noch offenen Entwicklungsbedarf in einigen Aspekten wurde dieses Arbeitspapier erstellt. In diesem sollen die Möglichkeiten der NFC-Technologie erörtert und geklärt werden. Auf Grund der Interaktionen von NFC mit anderen Geräten und dem Austausch von Informationen, lag der Schwerpunkt dieser Arbeit in der Ermittlung und dem Test von Schnittstellen. Eine weitere Zielsetzung ist es, verschiedene Schlüsselfunktionen der Technologie, wie Kommunikation zwischen NFC tauglichen Geräten, auf ihre Umsetzbarkeit zu testen.

Vor dem Hintergrund wird im ersten Teil dieses Papiers die Theorie zu „Near Field Communiction“ in der Theorie betrachtet. Hierbei werden die allgemeine Funktionsweise, die technische Funktionsweise und verschiedene Protokolle dargestellt. Des Weiteren wird auf den aktuellen Stand der Technik eingegangen. Um die Praxistauglichkeit von NFC einschätzen zu können, werden im Anschluss die Verbreitung und die Sicherheit der Technologie analysiert.

Der anschließende, praktische Teil stellt den Projektverlauf dar. Um in die NFC Thematik einzusteigen, wurden ein NFC fähiges Mobiltelefon, eine Arduino-Mikrocontrollerboard mit NFC-Shield und ein NFC-Tag verwendet. Im Projektverlauf wird beschrieben, wie diese miteinander agieren können und welche Funktionen aus dem theoretischen Teil bei diesen Geräten bereits implementiert wurden. Hierbei wird auf die, beim Testen der Funktion, gestoßen Schwierigkeiten eingegangen. Auf Grundlage des zuvor dargestellten Wissens werden Lösungen der zu bewältigenden Schwierigkeiten skizziert. Zum Abschluss geht diese Arbeit auf die Ergebnisse der Programmierung mit dem Arduino ein.

[1] (DPA, 2012); (Gernert, 2012)
[2] (Jovanovic & Muñoz Organero, 2011)

A) Theoretischer Teil

1. NFC

Near Field Communication (NFC) basiert auf RFID und ermöglicht die Kommunikation von Geräten auf kurze Distanzen von 5 bis 10 cm durch Radiosignale. Diese werden mit einer Frequenz von 13.56 MHz ausgestoßen, was eine Datenübertragung bis zu einer Geschwindigkeit von 424 kb/s ermöglicht.[3]

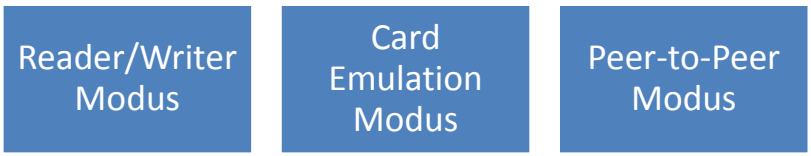

Abbildung 1: Betriebsarten NFC fähiger Geräte

Bei NFC fähigen Geräten wird zwischen verschiedenen Betriebsarten unterschieden. Zum einen können, auf passiven Tags gespeicherte, Informationen ausgelesen werden.[4] Dieser Prozess wird gestartet, wenn sich der Tag in der Reichweite eines aktiven Geräts befindet. Dieses erzeugt ein Feld und betreibt damit den Tag. Somit benötigen die passiven Tags keine eigene Energieversorgung.[5] Des Weiteren sind die Tags mit Stückpreisen von wenigen Euros günstig und können mehr Informationen enthalten als herkömmliche Barcodes, die ebenfalls passiv ausgelesen werden. [6]

Wenn Informationen von einem Tag nur ausgelesen werden, befindet das Gerät sich im Reader Modus. Im Writer Modus können die Tags neu beschrieben werden.[7] Ob dies möglich ist, hängt von den Tags ab. Unter Umständen sind diese nicht überschreibbar, beziehungsweise beschreibbar. [8]

Die Geräte können selber als Tag fungieren und somit ausgelesen werden. Dies wird als Card Emulation Modus bezeichnet. Wenn zwei Geräte abwechselnd in den Card Emulation Modus gehen, um miteinander zu kommunizieren, wird dies als Peer-to-Peer Kommunikation bezeichnet.[9]

Eine Besonderheit von NFC fähigen Geräten ist, dass ohne explicit spezifizierten Standard eine Interaktion mit RFID Tags/ Geräten möglich ist. Dies liegt daran, dass NFC auf den Normen und der Technologie von RFID aufbaut, sich aber durch eine kürzere Reichweite und eine schnellere Datenübertragung auszeichnet.[10] NFC basiert auf den folgenden Normen:

- ISO 18092
- ISO 14443
- JIS X 6319-4

[3] (Steffen, Preißinger, Jörg, Schöllermann, Müller, & Schnabel, 2010), S.1; (Langer & Roland, 2010), S.87
[4] (Nokia, 2011), S.6
[5] (Langer & Roland, 2010); S.92f
[6] (Gallo, 2011); S.8
[7] (Nokia, 2011), S.7
[8] (Gallo, 2011), S.8ff
[9] (Langer & Roland, 2010), S.91
[10] (Langer & Roland, 2010), S.89ff

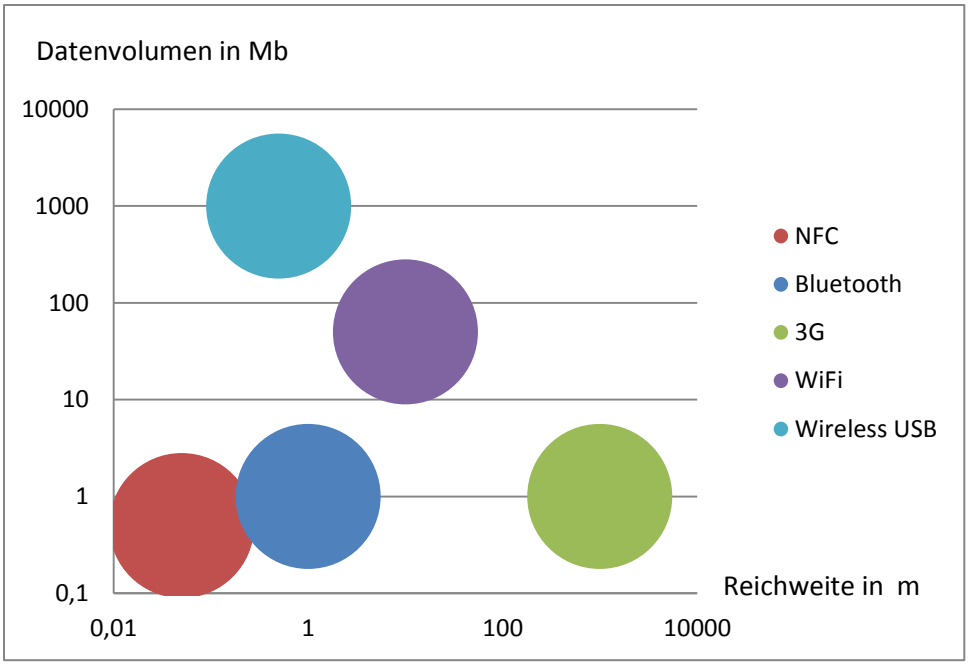

Abbildung 2: Vergleich verschiedener Technologien zur Datenübertragung[11]

Im Vergleich mit anderen Technologien ist erkennbar, dass NFC vor allem im Nahbereich und mit geringer Datenlast operiert. Obwohl NFC nur in einem kleinen Bereich einsetzbar ist, hat es einige Vorteile gegenüber den anderen Technologien. Zum einen wird die Verbindung zwischen zwei Geräten sehr schnell aufgebaut; für beispielsweise schnelle bargeldlose Bezahlung. Außerdem wird diese Eigenschaft dazu genutzt, den Authentifizierungsvorgang für Bluetooth zu beschleunigen, in dem dies über NFC geschieht und die Verbindung dann weitergegeben wird. [12] Des Weiteren ist die geringe Reichweite eine zusätzliche Sicherheit gegen Abhören der Verbindung.[13] Größere Datenvolumina werden für die bisherigen NFC Anwendungen nicht benötigt und würden die Kosten für die Tags erhöhen.

[11] Vgl. (NFC Forum, 2011), S.3
[12] (NFC Forum, 2011), S.1; (Jovanovic & Muñoz Organero, 2011), S.3
[13] (Jovanovic & Muñoz Organero, 2011), S.3

2. Technische Grundlagen (Protokolle)

Die zugrunde liegenden Standards werden vom NFC Forum entwickelt und festgelegt. Dieses ist ein Zusammenschluss von Unternehmen. Zwei von diesen sind NXP Semiconductors und Sony, welche die Technologie 2002 entwickelt haben. Hierbei werden die Übertragungsprotokolle von MIFARE und FeliCa kombiniert und erweitert. [14]

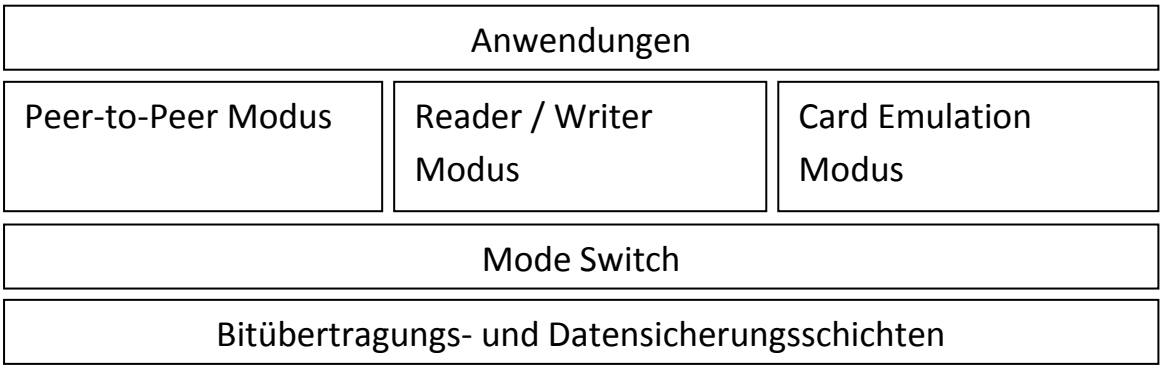

Abbildung 3: Vergleich verschiedener Technologien zur Datenübertragung[15]

In der vom NFC Forum definierten Architektur teilen sich die drei Betriebsmodi die Bitübertragungs- und Datensicherungsschichten. Erst die Mode Switch Prozedur legt den Betriebsmodus fest. [16]

Beim Aufbau einer Verbindung wird zwischen Initiator und Target unterschieden, wobei jedes NFC Gerät im Normalzustand im Target Modus ist, um Energie zu sparen. Der Initiator startet die Kommunikation, in dem er ein Signal mit einer beliebigen Übertragungsgeschwindigkeit aussendet. Das Target antwortet im Anschluss mit derselben Übertragungsgeschwindigkeit. Im passiven Modus stellt der Initiator mit seinem Signal die Energieversorgung für das Target. Im aktiven Modus teilen sich die beiden Geräte den Energieaufwand, in dem beide ihr eigenes Trägersignal erzeugen. Damit zwei aktive Geräte miteinander kommunizieren können, muss zusätzlich ein Kollisionsvermeidungsprotokoll durchlaufen werden. Hierbei prüft das Gerät zunächst, ob bereits ein Trägersignal vorhanden ist, bevor es sein Eigenes aussendet. Für den Fall, dass es mehrere Targets gibt, bestimmt jedes Target aufgrund einer Zufallszeit die Wartezeit auf die Anfrage. Wenn es trotzdem Überschneidungen gibt, beginnt die Aktivierungsprozedur von neuem.[17]

Sobald die Verbindungsherstellung erfolgt ist, können Daten ausgetauscht werden. Hierbei kommt das NFCIP-1 Datenaustauschprotokoll zum Zug (siehe ISO 18092). Über dieses Protokoll werden Informationen im NDEF Format ausgetauscht.

Hierbei können mehrere Datensätze zusammen übertragen werden. Jeder dieser Datensätze enthält einen Header, der angibt, um was es sich handelt (Identifier), wie lange die Nachricht ist (Length) und in welchem Dateiformat diese abgespeichert ist (Type). Hinter dem Header wird die eigentliche Nachricht (Payload) übertragen. Die Daten werden hexadezimal übertragen.

[14] (Langer & Roland, 2010), S.87 ff
[15] In Anlehnung an (Keen, 2009), S.6
[16] (Langer & Roland, 2010), S.90 f
[17] (Langer & Roland, 2010), S.91 ff

Das NFC Forum hat verschiedene Dateiformate (RTD) definiert. Hierzu zählen unter anderem Text, Uri und Smart Poster. [18]

Diese Nachricht kann auf die NFC Tags geschrieben werden. Im Folgenden wird besonders auf den MIFARE Tag eingegangen. Der Speicher des Tags ist in Sektoren unterteilt, die jeweils vier Blöcke haben. Jeder Block enthält 16 Bytes.

```
------------------------Sector 0-------------------------
Block 0  02 AE CD 36 57 88 04 00 C1 85 14 93 45 10 27 11
Block 1  14 01 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1
Block 2  03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1
Block 3  00 00 00 00 00 00 78 77 88 C1 00 00 00 00 00 00
------------------------Sector 1-------------------------
Block 4  00 00 03 14 D1 01 10 55 05 2B 31 20 32 31 32 20
Block 5  35 35 35 20 31 32 31 32 FE 00 00 00 00 00 00 00
Block 6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Block 7  00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00
```

Abbildung 4: Speicherarchitektur Tag

Die Bedeutung der Bytes nimmt von links nach rechts und von oben nach unten ab. Somit ist Das erste Byte in Block 0 das MSB (Most Significant Byte) und das letzte im Block das LSB (Least Significant Byte). In ihnen sind die Daten hexadezimal im ASCII Code abgespeichert.[19]

Der Sektor 0 enthält die Herstellerdaten und den Unique Identifier. Über diesen können die Lesegeräte einzelne Tags auseinander halten.[20]

Byte Nummer innerhalb eines Blocks															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Key A						Access Bits			GPB	Key B					

Abbildung 5: Aufbau Block 3 (Sector Trailer)

Die vierte Zeile jedes Sektors enthält die Access Bits (Byte Nummer 6-8), das General Purpose Byte (GPB) und die Schlüssel (Key A und Key B). Über die Access Bits wird der Zugang für das Protokoll gesteuert. Deswegen ist es wichtig, diese nicht zu überschreiben. Über Key A und B wird der Zugang für Lesen und Beschreiben gesteuert. Es wird empfohlen ungenutzte Sektoren über Schlüssel zu sperren, damit keine ungewünschten Änderungen vorgenommen werden können.[21] Über das GDP wird angegeben, mit welcher Methode die Daten angesteuert werden können.[22]

Der Speicheralgorithmus einer Nachricht wird am folgenden Beispiel verdeutlicht.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Block 4	00	00	03	14	D1	01	10	55	05	2B	31	20	32	31	32	20

Abbildung 6: Beispiel NDEF Nachricht

[18] (Nokia, 2011), S.17 ff
[19] (NXP Semiconductors, Rev. 1.2; 3. Mai 2011 a), S.6
[20] (NXP Semiconductors, Rev. 1.2; 3. Mai 2011 a), S.6
[21] (NXP Semiconductors, Rev. 1.2; 3. Mai 2011 a), S.10
[22] (NXP Semiconductors, Rev. 1.2; 3. Mai 2011 a), S.11

Die ersten zwei Bytes sind unbeschriebene Platzhalter. Bytes 2-4 enthalten den TLV Block. TLV steht für Type, Length und Value.

Byte Nummer	TLV	Wert	Bedeutung
2	Type	03	Der Block enthält eine NDEF Nachricht.
3	Length	14	Diese Nachricht enthält 14 Zeichen.
4	Value	D1	Gibt das Format für NDEF Nachricht an.

Abbildung 7: Erläuterung TLV

Nachdem im TLV Block definiert wurde, was abgespeichert wurde, wie lange es ist und welches Format es hat, geben die Bytes 5 bis 8 an, wie das NDEF Format genau aussieht.

Byte 5 gibt an, dass es sich um ein dem NFC Forum bekanntes Format handeln. Danach wird die Länge des Payloads angegeben. Im übernächsten Byte wird das genutzte NDEF Format angegeben. In diesem Fall sagt die „55“ an, dass es sich um einen URI-Nachrichten Type handelt. [23]Dieser wird in Byte 8 (URI Identifier) weiter spezifiziert. Es handelt sich um eine Telefonnummer. In Abbildung 8 sind weitere Beispiele für URI Identifier angegeben. Insgesamt sind 24 definiert.

Wert	Protokoll	Typ
01	http://www.	Webseite
03	http://	Webseite
05	tel:	Telefonnummer
06	mailto:	E-Mail Adresse

Abbildung 8: Uri Identifier[24]

Ab Byte 8 beginnt der eigentliche Nachrichtentext, der im ASCII Format hexadezimal abgespeichert ist. Dieser endet im nächsten Block mit der Angabe „FE“, welches die NDEF Nachricht abschließt.

Zusammenfassend ergibt sich für die Nachricht folgendes Bild:

Byte Nummer	Bezeichnung	Beispiel	Bedeutung
0	Platzhalter	0	Leer
1	Platzhalter	0	Leer
2	Typ	03	NDEF Nachricht
3	Length	14	Enthält 14 Zeichen
4	Value	D1	Nachrichten Format
5	Type length	01	Anzahl von Bytes für den Typ (s.7)
6	Pay load length	10	Anzahl Payload
7	Type	55	Uri
8	Identifier	05	Telefonnummer
9	Payload	2B	Beginn der Nachricht mit „+“
…			
Block 5 Byte 9	Payload	FE	Nachricht zu Ende

Abbildung 9: Entschlüsselte NDEF Nachricht

Eine Nachricht kann über mehrere Sektoren und Blöcke geschrieben werden. Insgesamt kann ein Tag auch mehrere Nachrichten enthalten.

[23] (Nokia, 2012)
[24] (Fried, 2012)

3. Stand der Technik

Auf dem Markt befinden sich zahlreiche NFC-Geräte, NFC-Shields, NFC-fähige Karten und Tags. Dabei unterscheiden sich die einzelnen Geräte in ihrem grundlegenden Aufbau nur geringfügig voneinander. In diesen ist entweder ein NFC-Chip[25] eingebaut oder als zusätzliches Gerät oder Modul integriert. Bei den NFC-Shields handelt es sich um Platinen mit einem NFC-IC (beispielsweise dem PN533 oder PN532 von NXP) die in Verbindung mit einem Mikrokontroller (beispielsweise dem ATmega 328 im Arduino Duemilanove) verwendet werden können.[26] NFC-fähige Karten sind beispielsweise die MIFARE Classic 1K, ebenfalls von NXP Semiconductors. Auf weitere Kartentypen wie MIFARE Ultralight, Plus S und ähnliche von NXP Semiconductors wird an dieser Stelle nicht weiter eingegangen, da sich diese Projektarbeit primär der Kommunikation zwischen Arduino und Handy über NFC widmet.

Ein Tag ist, in diesem Zusammenhang, als Chipkarte in beliebiger Form zu verstehen. Grundsätzlich handelt es sich bei einem Tag um einen Datencontainer mit hinterlegten Informationen. Diese können mit NFC-Geräten wiederum ausgelesen werden. Die Tags können mit einem ausschließlichen Lesezugriff oder auch wiederbeschreibbar konfiguriert werden. Das in Abschnitt 2 angesprochene NFC Forum definiert vier verschiedene Typen von Tags.

	NFC Spezifikation Typ 1 Tag	**NFC Spezifikation Typ 2 Tag**	**NFC Spezifikation Typ 3 Tag**	**NFC Spezifikation Typ 4 Tag**
Basis-Norm	ISO/IEC 14443A	ISO/IEC 14443A	FeliCa (JIS X 6319-4)	ISO/IEC 14443
Konfiguration	Les- /wieder-beschreibbar	Les- /wieder-beschreibbar	Les- /wieder-beschreibbar	Les- /wieder-beschreibbar
Speicher-kapazität	96 bytes um 2 kB erweiterbar	64 bytes (Statisch); > 64 bytes (dynamisch)	Maximal 1 MB	Maximal 32 KB

Tabelle 1: NFC-Tag Definitionen und Unterschiede[27]

3.1 Hardware

Nachfolgend werden die drei Hardware-Hauptkomponenten der Projektarbeit dargestellt. Hier liegt das Augenmerk auf den unterschiedlichen Varianten und Ausprägungen der Geräte und Shields.

Als Plattform für die NFC-Shields wird ein sogenanntes Arduino Mikrocontrollerboard mit bereits eingerichteten Schnittstellen und Schaltkreisen verwendet. Es wurde entwickelt um Nichtinformatikern das Programmieren zu erleichtern. Hierbei kann es als Steuereinheit für verschiedene Installationen genutzt werden. Ein Vorteil des Arduinoboards ist, dass es leicht mit weiteren Schnittstellen erweitert werden kann.

Bei den NFC-Shields für Arduino ist zwischen den zwei Haupt-Herstellern Seeed Studio und Adafruit Industries zu unterscheiden. Auf dem Markt existieren zwar weitere NFC-Shields für Arduino wie das NFC Modul von „libelium“ (www.cooking-hacks.com), diese sind jedoch kaum verbreitet. Daher existieren dafür auch fast keine Forumsdiskussionen und kaum Support der Community.

[25] Vgl. (NXP Semiconductors, 2009 c)
[26] Vgl. (NXP Semiconductors, Rev. 3.3; 16. Juli 2012 e). S. 1
[27] Vgl. (NFC Forum, 2011-04-13), (NFC Forum, 2011-05-31), (NFC Forum, 2011-06-28 a), (NFC Forum, 2011-06-28 b), (NFC Forum, 2012-07-11)

Die beiden großen Hersteller bieten ein NFC-Shield, basierend auf dem PN532 von NXP an. Die zwei Shields unterscheiden sich durch ihren Aufbau und das verwendete Bussystem. Bei dem Shield von Seeed Studio wird ein Serial Peripheral Interface (nachfolgend SPI) nach dem Master-Salve Prinzip verwendet.[28] Dadurch können die meisten analogen und digitalen Pins auf dem Arduino für weitere Funktionen verwendet werden. Das SPI verwendet vier Pins auf dem Arduino, beim Arduino Duemilanove sind das Pin 10,11,12 und 13. Definiert sind diese als der Slave Select (SS) auf Pin 10, der Master out, Slave in (MOSI) auf Pin 11, der Master in, Slave out (MISO) auf Pin 12 und die Serial Clock (SCK) auf Pin 13.[29]
Bei dem Shield von Adafruit Industries können zwei Bussysteme verwendet werden: Das SPI und der I-Quadrat-C (nachfolgen I2C), entwickelt von NXP. Der I2C ist als Master-Slave Bus konzipiert und benötigt nur drei Signalleitungen, im Fall des Arduinos nur drei Pins.[30] Bei dem Arduino Duemilanove sind das Pin 4 und Pin 5 für den I2C und ein frei wählbarer Pin als Unterbrechungsanforderung (kurz IRQ).[31]

Als dritte Komponente, zum Auslesen der beschrieben Tags, wird ein NFC-fähiges Smartphone verwendet. Für die Projektarbeit wird ein Samsung Galaxy Nexus verwendet. Auf dem deutschen Markt sind zum Zeitpunkt der Projektarbeit mehrere NFC-fähige Smartphones erhältlich. Dabei handelt es sich um das Google Galaxy Nexus und Galaxy Nexus S von Samsung, Samsung Galaxy S3, Samsung Wave 578, RIM BlackBerry 9900 Bold, Acer Liquid Express, Nokia 700, Nokia 701, RIM BlackBerry Curve 9360, Samsung Wave M, und Samsung Wave Y.
Die Smartphones haben die NFC Antenne meist auf ihrer Rückseite platziert, dabei ergeben sich kleine Unterschiede in der Platzierung. Beim Samsung Galaxy Nexus ist die Antenne im Akku integriert, bei Anderen wiederum im Akkudeckel. Daraus resultieren geringfügige Unterschiede in der Reichweite der NFC.

3.2 Software

Bei dem ausgewählten Smartphone wird Android als Betriebssystem genutzt. Dies ist ein Opensource-Betriebssystem und erleichtert somit das Programmieren von zahlreichen Applikation. Die Android-Version auf dem Galaxy Nexus ist 4.1.1 (Jelly Bean). Für Android existieren im Play Store von Google (Android Apps Market) zahlreiche Apps für das Lesen und Beschreiben von Tags. Einige Apps sollen auch in der Lage sein Tags emulieren zu können. Nachfolgend Tabelle zeigt die Funktionen und Namen der installierten Apps auf dem verwendeten Smartphone in der Projektarbeit.

Name App	Read Tags	Write Tags	Tag Emulation
TagWriter (NXP)	+	+	+
TagInfo (NXP)	+	-	-
NFClassic	+	+	+
GoToTags	+	+	-
NFC Reader	+	-	-
NFC TagReader	+	+	+

Tabelle 2: Übersicht verwendete NFC Apps für Android Smartphones

[28] Vgl. (Seeed Wiki, 2011)
[29] Vgl. (Library von Seeed Technology Inc., 2011)
[30] Vgl. (NXP Semiconductors, Rev. 5; 9. Oktober 2012 f)
[31] Vgl. (Fried, PN532 RFID/NFC Tutorial, 2012)

Für die Programmierung des Arduino sind verschiedene Versionen der Arduino Software verfügbar. Für das Betreiben des NFC-Shields sind von den beiden Herstellern Seeed Studio und Adafruit Industries Libraries verfügbar. Wie in Abschnitt 7 untersucht wird, muss bei dem Zusammenspiel der Software miteinander auf die Kompatibilität der unterschiedlichen Versionen geachtet werden. Für den Arduino sind die integrierten Entwicklungsumgebungen (IDE) der Versionen 0001 bis 0023, 1.0 und 1.0.1 für Windows, Mac OS X, Linux 32 bit und 64 bit vorhanden. Getestet werden in der Projektarbeit die Versionen 0022, 1.0 und 1.0.1, da ab Version 0022 die Library von Seeed Studio für Arduino funktionieren soll.[32] Da der Wissensstand auf diesem Gebiet nicht gesichert ist, wird in dieser Projektarbeit die Kompatibilität der einzelnen Versionen zueinander in Abschnitt 7 ebenfalls kurz untersucht.

Für die NFC-Shields existieren zwei unterschiedliche Libraries. Leider ist eine klare Versionshistorie durch das Wiki von Seeed Studio nicht möglich. Aus diesem Grund sind die verwendeten Libraries auf der beigefügten CD enthalten. Dabei handelt es sich um die Library von Seeed Studio „PN532_SPI“ und die von Adafruit Industries „Adafruit_PN532“ bei Verwendung des Bussystem SPI und für die Verwendung des Bussystem I2C „Adafruit_NFCShield_I2C“.

[32] Zahlreiche Forumsdiskussionen

4. Anwendungsmöglichkeiten

Es gibt vielfältige Anwendungsmöglichkeiten für NFC. Bedingt durch die kurze Entfernung zwischen den Geräten, gilt NFC als sichere Möglichkeit zur Authentifizierung.

Deswegen wird NFC verstärkt im bargeldlosen Zahlungsverkehr eingesetzt.[33] Beispielsweise haben die Sparkassen 2012 Girokarten mit NFC Chip an ihre Kunden verschickt. Hierbei kann der Kunde Geld im Voraus auf seine NFC Funktion laden und bei bestimmten Einzelhändlern hiermit bezahlen. Der Vorteil ist, dass der Authentifizierungs- und Abbuchungsvorgang wesentlich schneller ist als über die EC-Funktion.[34] Allerdings ist NFC nicht vollständig gegen Lauschangriffe geschützt[35] und soll deswegen als Ersatz des Bargeldes dienen, aber keinen vollen Zugriff auf das Konto geben.

Besonders in Asien werden Mobiltelefone mit NFC zur bargeldlosen Bezahlung eingesetzt. Ebenso gibt es einzelne Versuchsprojekte in Deutschland, beispielsweise „Touch and Travel“ der Deutschen Bahn.[36]

Weiterhin kann NFC genutzt werden, um den Zutritt zu Bereichen zu regeln. Dadurch, dass NFC Authentifizierung offline stattfinden kann, würde Kosten eingespart werden.[37] Außerdem könnten die Zugriffsrechte temporär gemacht werden, beispielsweise indem das NFC Mobiltelefone in bestimmten Abständen nach einem anderen Tag gefragt wird. Der Vorteil in der Card Emulation ist, dass Mobiltelefone mehrere Tags simulieren und weitere jederzeit erzeugen können.

In der Produktion könnte NFC zum Einstellen und Aktivieren der Maschinen dienen. Hier würden die notwendigen Daten vom Mobiltelefon oder Tag übertragen werden. Ein großer Vorteil bei den Mobiltelefonen wäre, dass die Einstellungen täglich aktualisiert werden können. Ein Szenario könnte sein, dass sich der Werker morgens per NFC einstempelt und daraufhin das Fertigungsprogramm inklusive der Routinen für die Maschinen zugesendet bekommt. Die gefertigten Produkte könnten dann ebenfalls einen Tag bekommen, welches mit dem Mobiltelefon oder direkt durch die Maschine beschrieben wird. Somit könnte bereits während der Fertigung die Dokumentation erstellt werden, dabei würde der Tag in jedem Produktionsschritt um das Verfahren und die Fertigungszeit erweitert werden. Am Ende könnten die Daten wiederum von einem NFC fähigen Gerät ausgelesen und daraus die Dokumentation für den Kunden erstellt werden. Außerdem wäre damit die individuelle Kennzeichnung im Falle einer Gewährleistung sichergestellt.

[33] (Gernert, 2012)
[34] (DPA, 2012)
[35] (Langer & Roland, 2010), S.105 ff
[36] (NFC Forum, 2011), S7 ff
[37] (Warnke, 2008), S.3

5. Sicherheit

Auf Grund der Benutzung sensitiver Daten ist die Absicherung der Datenübertragung sehr wichtig. Hierbei geht der größte Schutz auf die geringe Reichweite zurück. Da Sender und Empfänger sich physisch auf ungefähr 4 cm annähern müssen, ist ein Abhören der Verbindung durch Dritte schwierig, aber nicht unmöglich. Eine zusätzliche Absicherung gegen das Abhören bildet das implementierte Antikollisionsprotokoll. Dieses bemerkt, wenn mehr als zwei Geräte beteiligt sind, und kann den Vorgang abbrechen. Bei bargeldlosen Transaktionen kann zusätzlich ein Pin abgefragt werden, um den Vorgang zu verifizieren. Hierbei muss auf ein sicheres Element im NFC Gerät zurückgegriffen werden. Dies kann ein spezieller Chip sein oder das sichere Element der SIM Karte. Dieses würde bei erfolgter Verifizierung die notwendigen NFC Funktionen freischalten beziehungsweise aktivieren.[38]

Zusätzlich kann die Kommunikation zwischen den Geräten verschlüsselt werden. Dieses schützt aber wie die anderen Vorkehrungen nicht gegen eine sogenannte Man-in-the-Middle-Attack. Hierbei würde beide Gerät anstatt zueinander eine sichere Verbindung zum ausspionierenden Gerät aufnehmen. Dieses könnte die erlangten Daten dann über eine Technologie mit längerer Reichweite, wie Bluetooth und WLAN, weitergeben.[39]

Eine weitere Bedrohung ist die Manipulation von Tags. Wenn diese an öffentlichen Orten, zum Beispiel in Postern zur besseren Interaktion mit dem User, angebracht sind, können diese durch NFC fähige Geräte verändert oder überschrieben werden. Hierbei könnten Funktionen im Mobiltelefon ungewollt aktiviert werden; wie das Anwählen einer kostenpflichtige Hotline oder das Herunterladen von Malware über das UMTS Netz. Eine Absicherung hiergegen sind vollständig beschriebene Tags, die nicht mehr beschrieben werden können. Eine Manipulation an den Daten würde dem Reader auffallen. Dadurch würde er den Tag als nicht valide einstufen. [40]

Es wäre außerdem denkbar, dass Mobiltelefone gezielt gestohlen werden, damit sich der Dieb Zutritt zu Veranstaltungen verschaffen kann. Dies wäre dadurch abgesichert, dass NFC nur aktiviert ist, wenn das Telefon entriegelt und das Display aktiviert ist. Somit wäre eine Nutzung des gestohlenen Telefons zur Eintrittskontrolle nicht möglich.

Alles in allem gibt es mehrere Sicherheitslücken bei NFC. Allerdings lassen sich diese mit Ausnahme der Man-in-the-Middle-Attack durch Sicherheitsmaßnahmen beheben. Wie bei allen Technologien wird die verantwortungsvolle Benutzung durch den User entscheidend sein.

[38] (Langer & Roland, 2010), S.87ff
[39] (Langer & Roland, 2010), S.105
[40] (Jovanovic & Muñoz Organero, 2011), S.5

6. Verbreitung

Bisher ist die Durchdringung mit NFC sehr gering. Die Ursache hierfür ist die geringe Anzahl an Mobiltelefonen mit dieser Technologie. Außerdem gibt es nur wenige Pilotprojekte, bei denen eine Zahlung mit Mobiltelefon möglich ist. Die Einführung der Girokarte mit NFC Tag durch die Sparkassen wird wahrscheinlich die Zahlung, auf diesem Wege vorantreiben. Ein hohes Hindernis hierbei wird vermutlich der User sein, da er die Vorteile schnellerer Transaktionen gegen einen höheren Aufwand wie Aufladen der Funktion und gegen eventuelle Bedenken einer drahtlosen Technologie aufwiegen muss.[41]

Die Ursache für die langsame Verbreitung der Technologie liegt in dem Interessenkonflikt der beteiligten Unternehmen.[42] Dieser entsteht, da unterschiedliche Kernkompetenzen gefragt sind. Für eine bargeldlose Bezahlung wird ein modernes Smartphone, das Betriebssystems dieses Smartphones, ein Dienstleister für die Transaktion, der Verkäufer und eventuell ein Mobilfunkanbieter, der das sichere Element zur Verfügung stellt, benötigt.

Tabelle 1 zeigt eine Auswahl von Akteuren, die an einer bargeldlosen Zahlung über NFC beteiligt sein können.

Smartphone Hersteller	**Smartphone Betriebssystem**	**Anbieter Transaktion**	**Verkäufer Ware**	**Mobilfunkanbieter**
Samsung	iOS (Apple)	Sparkasse	Deutsche Bahn	Telekom
Apple	Android (Google)	Visa	Edeka	Vodafone
HTC	Symbian (Nokia)	Mastercard	Karstadt	Eplus
Nokia				

Tabelle 3: Beteiligte bei der bargeldlosen Bezahlung mit Beispielen

Bei dieser Vielzahl der Akteure ist ersichtlich, dass eine Einigung auf die Verteilung der Entwicklungskosten und der Gewinne schwierig ist. Dies führt ebenso dazu, dass einzelne Kooperation unterschiedliche Standards entwickeln, welche wiederum die Verbreitung erschweren.[43] Am weitesten scheint hierbei GoogleWallet (Android App) zu sein, das in Kooperation mit Paypass (Mastercard) und Paywave (Visa) bargeldlose Zahlung über ein NFC Terminal und ein NFC Mobiltelefon ermöglicht.[44] Hierfür ist ein im Mobiltelefon zusätzlich implementierter Chip, der das Sichere Element enthält, notwendig. Diese ist bisher nur im Nexus S verbaut. Außerdem ist nur in den USA die notwendige Infrastruktur vorhanden.[45]

Ein weiteres Hindernis bei der Verbreitung von NFC könnte sein, dass das neue IPhone wider Erwarten keine NFC Funktion hat. Abgeleitet aus der Apple Strategie der Systemabschottung (siehe Itunes) könnte dies daran liegen, dass Apple noch keinen Weg gefunden hat, die Geldtransaktion eigenständig abzuwickeln.

[41] (Gernert, 2012)
[42] Vgl. (Jovanovic & Muñoz Organero, 2011), S.2
[43] (Jovanovic & Muñoz Organero, 2011), S.2
[44] (Gernert, 2012)
[45] (DPA/DAPD, 2011)

Bei der neuen Einführung der Sparkassen Karte mit NFC handelt es sich um eine Kooperation zwischen Ketten im Einzelhandel und der Bank. Hierbei wird die Girokarte zur Zahlungsabwicklung genutzt und auf den Einsatz eines Mobiltelefons wurde verzichtet.

In Asien wurden bereits Nachrüstungen von Mobiltelefonen mit NFC getestet. Hierbei wird die Antenne über die Simkarte mit dem Telefon verbunden. Somit ist das Sichere Element in der Simkarte.

Alles in allem wird bereits seit Jahren mit einer flächenweiten Verbreitung von NFC gerechnet. Diese scheint allerdings an finanziellen Interessenkonflikten stark gehemmt zu sein.

B) Praktischer Teil

7. Projektverlauf

Wie bereits in Abschnitt 3 erläutert, verfügt das NFC-Shield über verschiedene Modi. Um das Zusammenspiel der Komponenten Arduino, NFC-Shield, Tag und Smartphone systematisch zu untersuchen wurden folgende Fragestellungen beantwortet:

1. Welche Libraries und welche Beispiele daraus, arbeiten mit welchem NFC Shield und welcher Arduino Version zusammen.
2. Mit welcher App lässt sich ein mit dem NFC-Shield beschriebener Tag auslesen und beschreiben.
3. Lässt sich ein vom Smartphone beschriebener Tag mit dem NFC-Shields auslesen.
4. Funktioniert der Card Emulation Mode des NFC-Shields.
5. Funktioniert der Card Emulation Mode des Smartphones.
6. Lässt sich mit dem NFC-Shield ein emulierter Tag des Smartphones lesen.
7. Funktioniert eine Peer-to-Peer Kommunikation zwischen NFC-Shield und Smartphone.

Für die Versuche wurden folgende Komponenten verwendet:

- Arduino Duemilanove mit Atmega 328 (siehe Abbildung 10)
- PN532 RFID/NFC Shield 13.56 MHz RFID von Adafruit Industries (siehe Abbildung 11)
- NFC Shield V1.0 by HQF&Nancy 29/06/2011 von Seeed Studio (siehe Abbildung 12)
- NFC-Card/Tag: MIFARE Classic 1k beschrieben und unbeschrieben (siehe Abbildung 13)
- Software Arduino 1.0 mit Libraries „Adafruit_NFCShield_I2C", „Adafruit_PN532", „PN532_SPI", „PN532_SPI_V1"[46]

- Handy Samsung Galaxy Nexus (Android 4.1.1)
- Apps: Siehe Tabelle 2 in Abschnitt 3.2

In den nachfolgenden Abschnitten werden die hier aufgeführten Fragstellungen erläutert und deren Ergebnisse aufgeführt.

7.1 Kompatibilität der Libraries mit den NFC-Shields

Um die Kompatibilität der einzelnen Libraries mit den jeweiligen Arduino Versionen zu testen, wurden die drei Arduino Versionen 0022, 1.0 und 1.0.1, jeweils zusammen mit den vier Libraries von Seeed Studio und Adafruit Industries, installiert. Daraufhin wurden die einzelnen Beispiele der Libraries aufgerufen und in der Entwicklungsumgebung über die Schaltfläche „Verify" auf Fehler überprüft. Anschließend wurden in der Entwicklungsumgebung (der Version 1.0, ohne auftretende Fehler) die beiden NFC-Shields, mit den Beispielen aus den Libraries, auf Kompatibilität geprüft. Dafür wurde jeweils ein Tag mit den NFC-Shields beschrieben und ausgelesen. Um die Lesefunktion zu kontrollieren, wurde ein unbeschriebener Tag jeweils nach dem Auslesen des beschriebenen Tags mit dem jeweiligen Beispiel ausgelesen. Dabei ergaben sich die in Tabelle 3 dargestellten Ergebnisse.

[46] Siehe beigelegte CD im Ordner „Libraries"

Zeile	Arduino Version	Library[47]	Beispiel	Verwend-bares NFC-Shield[45]	Kom-patibel	Tag lässt sich be-schreiben	Tag lässt sich auslesen
1	0022	1	iso14443a_uid	a	+	-	+
2	0022	1	mifareclassic_formatndef	a	+	+	-
3	0022	1	mifareclassic_memdump	a	+	-	+
4	0022	1	readMifare	a	+	-	+
5	0022	2	iso14443a_uid	s	+	-	+
6	0022	2	mifareclassic_formatndef	s	+	+	-
7	0022	2	mifareclassic_memdump	s	+	-	+
8	0022	2	readMifare	s	+	-	+
9	0022	2	readMifareClassic	s	+	-	+
10	0022	3	alle	s	-	-	-
11	0022	4	alle	s	+	-	-
12	1.0	1	iso14443a_uid	a	+	-	+
13	1.0	1	mifareclassic_formatndef	a	+	+	-
14	1.0	1	mifareclassic_memdump	a	+	-	+
15	1.0	1	readMifare	a	+	-	+
16	1.0	2	iso14443a_uid	s	+	-	+
17	1.0	2	mifareclassic_formatndef	s	+	+	-
18	1.0	2	mifareclassic_memdump	s	+	-	+
19	1.0	2	readMifare	s	+	-	+
20	1.0	2	readMifareClassic	s	+	-	+
21	1.0	3	PtoPInitiator	s	-	-	-
22	1.0	3	PtoPTarget	s	-	-	-
23	1.0	3	readAllMemoryBlocks	s	+	-	+
24	1.0	3	readMifareMemory	s	+	-	+
25	1.0	3	readMifareTargetID	s	+	-	+
26	1.0	3	writeMifareMemory	s	+	+	-
27	1.0	4	readAllMemoryBlocks	s	+	-	+
28	1.0	4	readMifareMemory	s	+	-	+
29	1.0	4	readMifareTargetID	s	+	-	+
30	1.0	4	writeMifareMemory	s	+	+	-
31	1.0.1	1,2,3,4	alle	a und s	-	-	-

Tabelle 4: Kompatibilität Arduino, NFC-Shields, Libraries

7.2 Apps zum Auslesen und Beschreiben von Tags

Als nächster Schritt sollten die mit dem NFC-Shield beschriebenen Tags mit den Apps auf dem Smartphone ausgelesen werden. Dabei ergaben sich keine Probleme und alle genannten Lesefunktionen der Apps konnten genutzt werden. Daraufhin wurden die Schreibfunktionen der Apps getestet. Dafür wurde die App „TagWriter“ von NXP genutzt um den zuvor mit dem NFC-Shield beschriebenen Tag mit neuem Inhalt zu beschreiben. Der mit neuem Inhalt beschriebene Tag konnte mit allen dafür geeigneten Apps aus Tabelle 2 ausgelesen werden.

[47] Verwendete Nummern und Buchstaben siehe Tabelle:

Library	Nummer	NFC-Shield	Buchstabe
Adafruit_NFCShield_I2C	1	PN532 RFID/NFC Shield von Adafruit Industries	a
Adafruit_PN532 (für SPI)	2	NFC Shield V1.0 von Seeed Studio	s
PN532_SPI	3		
PN532_SPI_V1	4		

7.3 Auslesen eines vom Smartphone beschriebenen Tags mit dem NFC-Shield

Um die Fragestellung 3 zu beantworten wurde ein leerer Tag mit Hilfe des NFC-Shields ausgelesen. Daraufhin wurde ein zweiter Tag mit dem Smartphone beschrieben. Dieser beschrieben Tag wurde nun mit dem NFC-Shield und dem Beispiel „mifareclassic_memdump“, unter Verwendung der Kombination aus Zeile 18 (vergleiche Tabelle 4), ausgelesen. Der mit dem Smartphone erzeugte Inhalt konnte erfolgreich ausgelesen werden. Daraufhin wurden die anderen Beispiele der Libraries, für das Auslesen von Tags, an diesem ausgetestet. Dabei wurde entdeckt, dass die Kombination aus Zeile 27 nicht mehr funktionierte. Diese Kombination konnte nur den unbeschriebenen Tag auslesen.

7.4 Card-Emulation Mode des NFC-Shields

Grundsätzlich ist der PN532 Chip in der Lage eine Tag, beziehungsweise eine Mifare Card zu emulieren.[48] Doch weder die Library von Seeed Studio oder Adafruit Industries haben diese Funktion implementiert. In zahlreichen Foren wird darüber spekuliert und diskutiert, wie eine solche Funktion realisiert werden könnte. Jedoch haben sorgfältige Nachforschungen keinen funktionierenden Ansatz bezüglich einer Umsetzung einer Emulation entdeckt.

7.5 Card-Emulation Mode des Smartphones

Hinsichtlich eines Card-Emulation Modes des Smartphones verhält es sich ähnlich wie mit dem NFC-Shield. Theoretisch ist dies möglich, jedoch ist noch keine App mit dieser Funktion verfügbar. Ein Problem bei einer solchen Applikation für das Smartphone ist der Unique Identifier (UID), eine eindeutige Identifikationsnummer.[49] Durch einige Forenbeiträge konnte festgestellt werden, dass durch eine geschickte Nutzung von dem in Abschnitt 6 genannten GoogleWallet eine Emulation schon erfolgreich durchgeführt werden konnte.[50] Dieses Vorgehen detailliert zu erläutern oder am Smartphone ebenfalls umzusetzen, würde den Rahmen dieser Projektarbeit überschreiten. Damit erübrigt sich ebenfalls Frage 6.

7.6 Peer-to-Peer Kommunikation

Als letzte Fragestellung sollte die Peer-to-Peer Kommunikation primär zwischen Arduino und Smartphone untersucht werden. Insgesamt müssen drei verschiedene Möglichkeiten der Peer-to-Peer Kommunikation über NFC unterschieden werden: Ersten die Kommunikation zwischen zwei Arduinos über das NFC-Shield, zweitens die Kommunikation zwischen zwei Smartphones und drittens die Kommunikation zwischen Smartphone und Arduino mit Hilfe des NFC-Shields.

Zunächst wird die erste Variante betrachtet. Dazu enthält die Library von Seeed Studio zwei Beispiele. Diese konnten jedoch wie in Zeile 21 und Zeile 22 der Tabelle 4 zu sehen nicht kompiliert werden. Die Beispiele „PtoPInitiator“ und „PtoPTarget“ enthielten Fehler im Programmcode und konnten nicht behoben werden. Recherchen in den Foren von „Stackoverflow“ und dem „Seeed Studio“ Forum konnten den Fehler ebenfalls nicht beheben.
Die angesprochenen Programmbeispiele sind für die Peer-to-Peer Kommunikation zwischen zwei Arduinos mit einem NFC-Shield konzipiert. Aus diesem Grund würde selbst eine fehlerfreie Library noch keinen direkten Nutzen für die Kommunikation zwischen Smartphone und Arduino über das NFC-Shield bieten.

[48] Vgl. (NXP Semiconductors, Rev. 3.2; 20. September 2012 d)
[49] Vgl. (BR STen, 2012)
[50] Vgl. (TechShek, 2012)

Die dritte Möglichkeit der Kommunikation via NFC bietet das Smartphone selbst, über Android Beam ab Version 4.0.[51] Hier können NDEF Nachrichten zwischen zwei Android betriebenen Geräten ausgetauscht werden. Auch diese Methode für eine Kommunikation zwischen Arduino und Smartphone zu verändern, beziehungsweise eine eigene App dafür zu konzipieren kam für diese Projektarbeit, in Anbetracht des Aufwands, nicht in Frage. Somit musste ein anderer Weg, anstelle einer möglichen Peer-to-Peer Kommunikation zwischen Smartphone und Arduino via NFC gefunden werden.

8. Schwierigkeiten und Lösungsansätze

Bei der beschriebenen Vorgehensweise konnten nicht alle Fragestellungen endgültig beantwortet werden. Dies resultiert unter anderem aus Schwierigkeiten mit der vorhandenen Software und Hardware.
Zu Beginn der Projektarbeit bestand die Hauptproblematiken darin, die unterschiedlichen Hardware- und Softwarekomponenten zu betreiben. Darunter fallen die Kompatibilität der unterschiedlichen Bibliotheken der NFC-Shields und deren Versionen mit der jeweiligen Arduino Version. Aus diesem Grund wurden Kompatibilitätstests durchgeführt. Die miteinander kompatiblen Versionen sind in Tabelle 4 aufgeführt.

Bei den Beispielen der Library „Adafruit_PN532 (für SPI)“ waren die vier Pins SCK, MOSI, SS und MISO nicht definiert. Deshalb wurden aus den Beispielen von Seeed Studio die jeweiligen Belegungen der Pins, unter Verwendung des SPI-Bussystems, in diese Beispiele übernommen. Dabei gilt es zu beachten, das unterschiedliche Arduinoboardvarianten unterschiedliche Pinbelegungen besitzen.

Das in Abschnitt 7.3 angesprochene Problem des Auslesens eines beschriebenen Tags, in der Kombination aus Zeile 27 in Tabelle 4, wurde versucht durch Änderung des Public Keys zu lösen. Dafür wurden im Programmbeispiel „readAllMemoryBlocks“ als Key A unterschiedliche Public Keys verwendet.

Um etwaige Programmierungsfehler des Beispielcodes auszuschließen wurden sämtliche Key A aus der Dokumentation von NXP ausgetestet. Tabelle 5 zeigt die dabei verwendeten Kombinationen.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
A0h	A1h	A2h	A3h	A4h	A5h
D3h	F7h	D3h	F7h	D3h	F7h
FFh	FFh	FFh	FFh	FFh	FFh

Tabelle 5: Verwendete Key A[52]

Keiner der verwendeten Schlüssel konnte den Programmcode zum gewünschten Ergebnis hin ändern. Aus diesem Grund wurden für das weitere Vorgehen nur noch die funktionierenden Programmbeispiele von der Adafruit Industries Library „Adafruit_NFCShield_I2C“ genutzt.

Der Versuch, einen mit dem Smartphone emulierten Tag mit dem Arduino auszulesen, war erfolglos. Um das zu testen wurde ein zuvor mit dem NFC-Shield beschriebener Tag mit dem Smartphone und der App „TagWriter“ (NXP) ausgelesen. Anschließend wurde der eingelesene Inhalt über die Option „Share Tag“, in der App, zum Senden freigegeben. Dieser emulierte Tag des Smartphones konnte mit dem zuvor verwendeten NFC-Shield nicht eingelesen werden. Wie in Abschnitt 7.6 angesprochen, kann diese Art von Nachricht scheinbar nur zwischen zwei Android betriebene Geräten ausgetauscht

[51] Vgl. (Android Developers, 2012)
[52] Vgl. (NXP Semiconductors, Rev. 1.3; 2. Oktober 2012 b) Table6, 9 und Figure 9

werden. Dennoch konnte bei diesem Versuch die UID des Smartphones ausgelesen werden. Diese veränderte sich bei jedem Einlesevorgang erneut, da es sich in diesem Fall nicht um die UID des NFC-Chips des Smartphones handelte, sondern um eine zufällig generierte ID.

Um dennoch eine Art Kommunikation zwischen Smartphone und Arduino herzustellen sollten nun die Erkenntnisse aus der bisherigen Vorgehensweise genutzt werden. Dadurch entstand der in Abschnitt 10 erläuterte Workaround.

9. Ergebnis

Abschließend sind aus dem Vorgehen folgende Ergebnisse festzustellen:

Die Kompatibilität der Software mit der getesteten Hardware ist nur beschränkt. Vergleiche dazu Tabelle 4, die alle getesteten Kombinationen aufführt und deren Funktion angibt.

Mit beiden NFC-Shields und dem Arduinoboard können die Inhalte auf einem Tag (MIFARE Classic 1k) gelesen werden. Dieser kann mit den NFC-Shields ebenso beschrieben werden. Durch das Beispiel „mifareclassic_memdump“ von Adafruit Industries kann jedes Byte eines NFC-Tags ausgelesen werden. Auf dem Android-Smartphone ist dies ist mit den genannten Apps nur bedingt möglich, einzig die App „NFClassic“ bietet eine vergleichbare Funktion. Diese ist allerdings auf MIFARE Classic 1k Karten beschränkt.

Mit dem Smartphone und den genannten Apps in Tabelle 2 können Tags ausgelesen und beschrieben werden. Der sogenannte „Card-Emulation Mode“ der Apps beschränkt sich auf die Verwendung zwischen Smartphones, basierend auf Arduino ab Version 4.0.

Eine direkte Kommunikation zwischen Smartphone und Arduino via NFC ist nur durch einen Workaround möglich. Die direkte Peer-to-Peer Kommunikation konnte nicht hergestellt werden. Allerdings können Smartphone und Arduino mit NFC-Shield NFC-Tags beschreiben und auslesen, daraus folgt der in Abschnitt 10 beschriebene Workaround Ansatz.

10. Programmierung mit Arduino

Nachfolgend wird der Workaround zum Herstellen einer Art Kommunikation zwischen Smartphone und Arduino beschrieben. Detaillierte Erläuterungen zu den einzelnen Bausteinen des Programmcodes finden sich in diesem, in auskommentierter Form, wieder.

Grundlegende Idee für den Programmcode wie er auf der beiliegenden CD im Order „Programm-Code/nfc_beschrieben_lesen_8“ zu finden ist, war die Herstellung einer Kommunikation mit Hilfe des MIFARE Classic 1k Tags. Da der Tag von Smartphone und Arduino les- und beschreibbar ist, sollte dieser als eine Art Zwischenhändler für das Überbringen der Daten von Smartphone zu Arduino und umgekehrt fungieren. Wie vorangegangen erläutert muss dabei das Kollisionsvermeidungsprotokoll beachtet werden. Das bedeutet, nur ein Gerät kann zur gleichen Zeit den Tag beschreiben oder auslesen.

Als erster Schritt wurden dabei die beiden Beispiele „mifareclassic_memdump“ und „mifareclassic_formatndef“ aus der „Adafruit_NFCShield_I2C-Library” als Grundlage für einen Programmcode

genutzt. Da beide Beispiele über den Serial-Monitor der Entwicklungsumgebung mit der gleichen Baudrate angesprochen werden können, bereitete dies keine Probleme. Zunächst wurden die beiden notwendigen Grundfunktionen *void setup ()*, die Vorbereitung, und *void loop ()*, die Ausführung des Programmes, zusammengelegt. Um einzelne Schritte in der nachfolgenden Programmierung besser nachvollziehen zu können, wurde entschieden, im Serial-Monitor den Initiierungsprozess des NFC-Shields und den Inhalt aller 64 Blöcke anzuzeigen. Ein erstes Zwischenergebnis des eingelesenen Tags ist im Anhang in Abbildung 14 zu finden.

Als nächster Schritt wurden die fixen Nachrichten, die auf den Tag geschrieben werden, in Nachrichten mit variablem Text umgewandelt. Dazu wurde der Befehl *nfc.mifareclassic_WriteDataBlock ()* genutzt, mit dem jeder Byte eines Tags beschrieben werden kann. Dabei muss beachtet werden, dass vor dem Beschreiben eines Blocks, dieser authentifiziert werden muss. Dies geschieht mit *nfc.mifareclassic_AuthenticateBlock().* Um den Tag mit einem beliebigen Text zu beschreiben, wurde die Funktion *serial.read ()* verwendet. Damit der eingegebene Text im Serial Monitor auf den Tag geschrieben wird, wurden zwei Eigenheiten der Entwicklungsumgebung genutzt. *Serial.read ()* kann nur die ersten 64 Bytes der Konsoleneingabe lesen und mit der Funktion *nfc.mifareclassic_WriteDataBlock ()* können Datenarrays auf den Tag geschrieben werden.
Damit wurden sechs Arrays, für sechs zu beschreibende Datenblöcke auf dem Tag erstellt und nach dem Aufbau TLV, Payload (Standardmäßig bestehend aus Nullen) und FE strukturiert. Durch eine for-Schleife werden die Daten aus dem Serial-Speicher solange in die „Standarddate-Arrays“(gefüllt mit Nullen) geschrieben, bis der Serial-Speicher keine Daten mehr enthält. Diese Abfrage wird mit S*erial.available()* realisiert. Bei vorhergehenden Versuchen des Beschreibens und Lesens des Tags mit Smartphone und Arduino wurde festgestellt, dass das Handy vorhandene Nachrichten auf dem Tag überbeschreibt, aber die restlichen, unbeschriebenen Blöcke nicht „zurücksetzt“. Deshalb bleiben alte Daten, von vorhergehenden Vorgängen, teilweise fragmentarisch auf dem Tag erhalten. Somit existieren neben der neuen Nachricht auf dem Tag weitere Daten (siehe in Abbildung 14 des Sektors 3). Das kann zu Komplikationen führen. Für das Smartphone konnte keine Lösung des Problems gefunden werden.
Beim Beschreiben mit dem Arduino wurde das Problem gelöst, indem nach den beschriebenen Sektoren alle restlichen Sektoren mit Nullen aufgefüllt werden. Dabei muss beachtet werden, dass der vierte Block eines jeden Sektors nicht vollständig mit Nullen beschrieben werden darf, da der Tag sonst beschädigt wird. Aus diesem Grund wurde im Programmcode ab Zeile 233 die for-Schleife mit der entsprechenden Bedingung genutzt. Abbildung 15 im Anhang zeigt das Ergebnis des beschriebenen Vorgehens. Der Programmcode befindet sich auf der beigelegten CD unter dem Ordner „Programm-Code/ndf_beschrieben_lesen_8.“

Auf diesen Grundfunktionen aufbauen müssten für einen vollständigen Workaround folgende Funktionen in den Programmcode eingebaut werden:
Der Arduino liest alle 3 Sekunden den Inhalt eines Tags aus und gleicht diesen mit einem Array ab, in dem der Inhalt aus der vorangegangenen Abfrage gespeichert wurde. Sind die Inhalte gleich, wird in weiteren 3 Sekunden die Abfrage wiederholt. Ändert sich der Inhalt im Vergleich zur letzten Abfrage, können über Eingabe in der Konsole neue Daten auf den Tag geschrieben werden.

Dabei ist es wichtig, dass das NFC-Shield in den 3 Sekunden Pause nicht mehr aktiv ist und den Tag somit „freigibt“ (Beachten des Antikollisionsprotokolls). Somit kann das Handy den Tag auslesen und mit neuem Inhalt beschreiben.

II. Fazit

Zusammenfassend lässt sich sagen, dass NFC an Bedeutung gewinnt. Zum einen wird die Technologie kommerziell, in Form kontaktloser Zahlungssysteme (z.B. GoogleWallet, GiroGo) verwertet, zum anderen nutzen Privatanwender diese, um eigene Ideen (z.B. KegDroid[53]) zu verwirklichen. Die Pläne für weitere Anwendungen[54] liegen seit Jahren in den Schubladen und der große Durchbruch von NFC wurde des Öfteren angekündigt.[55] Allerdings wird dieser bisher durch die fehlende Verbreitung von NFC-fähigen Smartphones gehemmt.

Ein Vorteil von NFC ist, dass die Standards und Protokolle bereits genau ausgearbeitet und durch das NFC Forum definiert wurden. Dadurch gab es Praxisbeispiele von Privatanwendern im Internet, an denen sich diese Projektarbeit orientieren konnte. Des Weiteren ist das Arduino ein weitverbreitetes Mikrocontrollerboard und die Hersteller der NFC Shields haben Libraries für diese herausgegeben. Trotz dieser Voraussetzungen gestaltete sich die Programmierung als schwierig. Oft stellten sich Bibliotheken und Quellcode als fehlerhaft oder mit dem Arduino nicht kompatibel heraus. Deswegen baute diese Projektarbeit auf einer funktionierenden Library auf und die gewünschten Funktionen wurden selbstständig programmiert.

Hierbei wurde das Zusammenspiel von Tag und Arduino durch Versuch und Irrtum analysiert. Ein Beispiel hierfür ist die hexadezimale Datenübertragung. Die Beschreibungen dieser Übertragung sind hoch komplex und unterschiedliche Autoren haben diese jeweils nur teilweise behandelt. Durch die Programmierung wurde ausprobiert, welche Stellen im Code entscheidend sind und die Funktion des Tags bestimmten.

Für diese Versuche war es hilfreich, dass es bereits Apps zum Beschreiben der Tags für Android Smartphones gibt. Alllerdings galt auch hierbei, dass längst nicht alle in gewünschter Weise funktionieren. Eine grundlegende Schwierigkeit war, dass eine Peer-to-Peer Kommunikation zwischen Arduino und Android nicht definiert ist. In Verbindung damit, dass der Card-Emulation Mode zwar für Android existiert, aber für Arduino noch nicht umgesetzt ist, konnte keine direkte Kommunikation zwischen den beiden Geräten hergestellt werden.

Mit der Kommunikation über den Tag wurde allerdings eine Alternative gefunden, Informationen zwischen den NFC-fähigen Geräten auszutauschen. Hierbei hat das Arduinoboard den Vorteil, dass es mehre NDEF Nachrichten auf einen Tag schreiben und lesen kann. Die Android Apps lesen jedoch nur die erste Nachricht auf dem Tag.

Alles in allem, wurde die Technologie NFC theoretisch vorgestellt. Basierend auf diesen Kenntnissen wurde der derzeitige Stand der frei verfügbaren Technik evaluiert und getestet. Im letzten Schritt wurde im Rahmen dieser Projektarbeit die die Programmbeispiele einer Library derart erweitert, dass eine Kommunikation zwischen Arduino mit NFC-Shield über ein MIFARE Tag mit einem Android Smartphone möglich ist.

[53] (Crider, 2012)
[54] Z.B. (Steffen, Preißinger, Jörg, Schöllermann, Müller, & Schnabel, 2010)
[55] (Jovanovic & Muñoz Organero, 2011)

Anhang

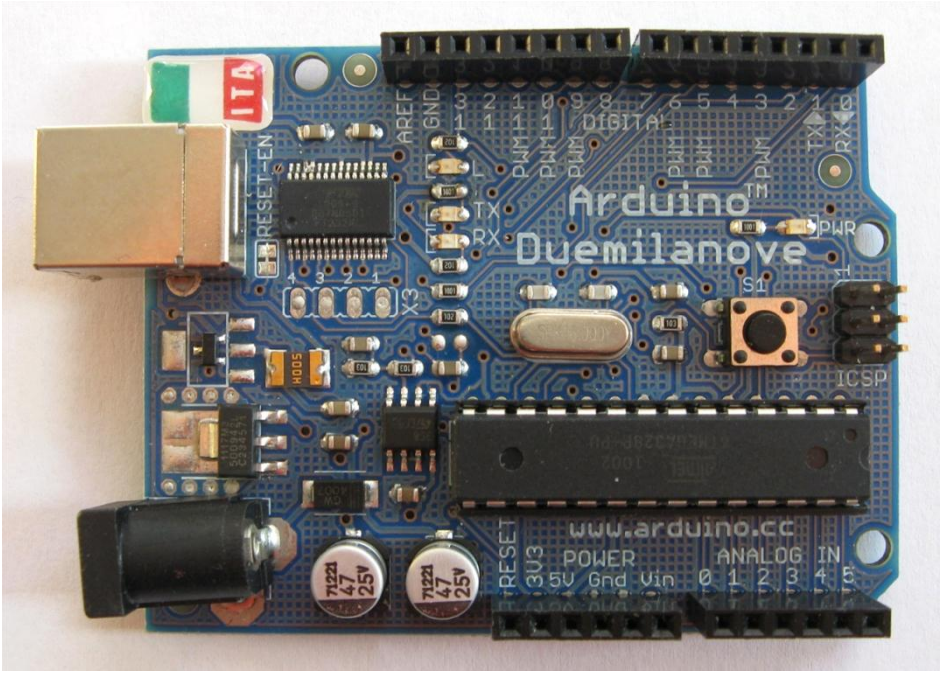

Abbildung 10: Arduino Duemilanove mit Atmega 328

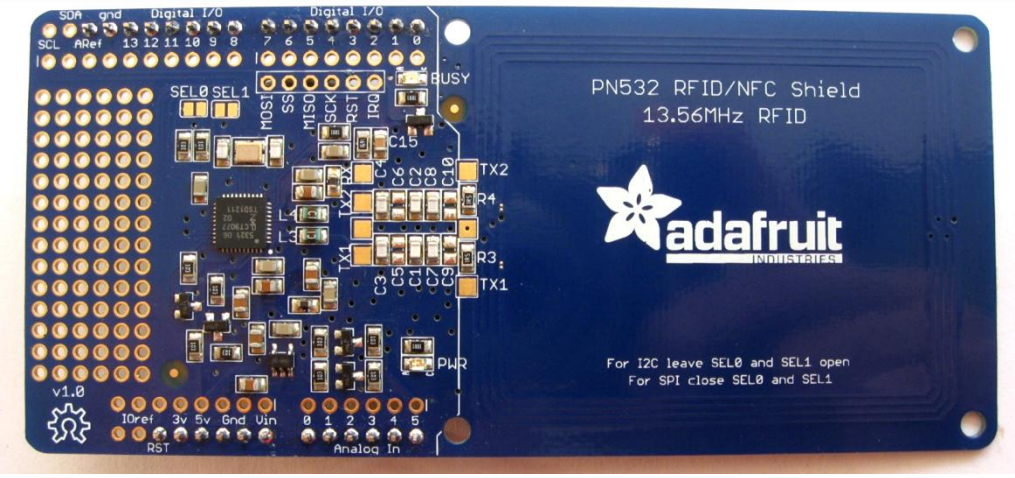

Abbildung 11: Adafruit PN532 NFC/RFID Controller Shield

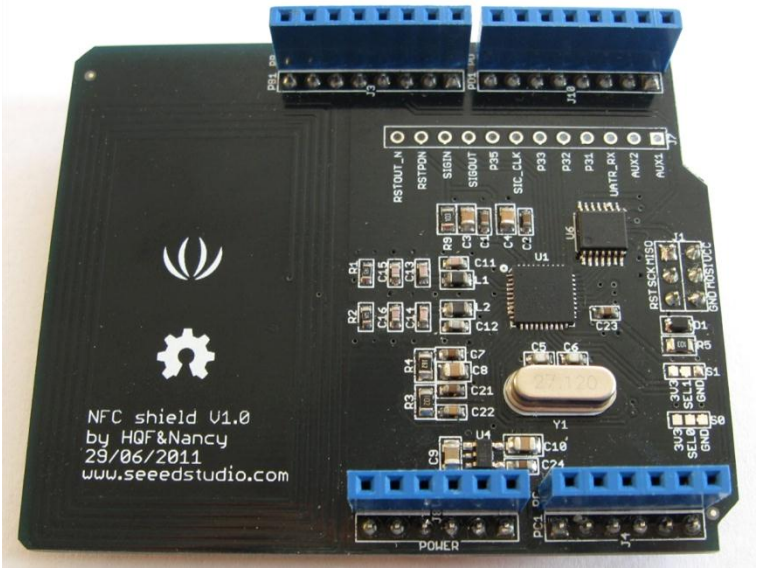

Abbildung 12: Seeed Studio NFC Shield V1.0

Abbildung 13: Tag (MIFARE Classic 1k)

```
Looking for PN532...
Found chip PN532
Firmware ver. 1.6

Place your Mifare Classic card on the reader to format with NDEF
and press any key to continue ...
Found an ISO14443A card
  UID Length: 4 bytes
  UID Value: 0x02 0xAE 0xCD 0x36

Seems to be a Mifare Classic card (4 byte UID)
Card has been formatted for NDEF data using MAD1
Writing URI to sector 1 as an NDEF Message
NDEF URI Record written to sector 1


Done!
-----------------------Sector 0-----------------------
Block 0  02 AE CD 36 57 88 04 00 C1 85 14 93 45 10 27 11  .®Í6W□..Á□.□E.'.
Block 1  14 01 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1  ...á.á.á.á.á.á.á
Block 2  03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1  .á.á.á.á.á.á.á.á
Block 3  00 00 00 00 00 00 78 77 88 C1 00 00 00 00 00 00  ......xw□Á......
-----------------------Sector 1-----------------------
Block 4  00 00 03 11 D1 01 0D 55 01 61 64 61 66 72 75 69  ....Ñ..U.adafrui
Block 5  74 2E 63 6F 6D FE 00 00 00 00 00 00 00 00 00 00  t.comþ..........
Block 6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 7  00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00  ......□.□@......
-----------------------Sector 2-----------------------
Block 8  6E 73 65 72 65 72 20 41 72 62 65 69 74 2E 20 76  nserer Arbeit. v
Block 9  69 65 6C 65 20 47 72 C3 BC C3 9F 65 20 45 6D 69  iele GrÃ¼Ã□e Emi
Block 10 6C 20 75 6E 64 20 41 6E 69 6B 61 0A FE 00 00 00  l und Anika.þ...
Block 11 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00  ......□.□@......
-----------------------Sector 3-----------------------
Block 12 37 35 6D 77 39 6F 75 FE 00 00 00 00 00 00 00 00  75mw9ouþ........
Block 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 15 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00  ......□.□@......
-----------------------Sector 4-----------------------
Block 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
Block 19 00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00  ......□.□@......
```

Abbildung 14: Ausschnitt aus „nfc_beschrieben_lesen_1.pdf“

```
Suche nach PN532...
Folgender Chip gefunden: PN532
Firmware ver. 1.6

Lege den Tag auf das NFC-Shield um diesen im NDEF Format zu beschreiben.
Die in der Konsole eingegebene Nachricht wird nach Druecken der Eingabetaste auf
den Tag geschrieben.
NFC-Tag gefunden.
  UID Laenge:: 4 bytes
  UID Value: 0x02 0xAE 0xCD 0x36

Scheint eine Mifare Classic Card (4 byte UID) zu sein
Karte/Tag wurde fuer NDEF formatiert.
URI wird auf Sector/en als NFDEF Nachricht geschrieben.
NDEF URI wurde auf Sector/ern geschrieben.


Fertig!
-----------------------Sector 0-----------------------
Block 0  02 AE CD 36 57 88 04 00 C1 85 14 93 45 10 27 11  .®Í6W□..Á□.□E.'.
Block 1  14 01 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1  ...á.á.á.á.á.á.á
Block 2  03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1 03 E1  .á.á.á.á.á.á.á.á
Block 3  00 00 00 00 00 00 78 77 88 C1 00 00 00 00 00 00  ......xw□Á......
-----------------------Sector 1-----------------------
Block 4  00 00 03 74 D1 01 74 55 00 54 65 73 74 20 FC 62  ....Ñ..U.Test üb
Block 5  65 72 20 77 69 65 20 76 69 65 6C 65 20 42 6C 6F  er wie viele Blo
Block 6  63 6B 73 20 68 69 6E 77 65 67 20 65 69 6E 20 00  cks hinweg ein .
Block 7  00 00 00 00 00 00 7F 07 88 40 00 00 00 00 00 00  ......□.□@......
-----------------------Sector 2-----------------------
```

Abbildung 15: Ausschnitt aus "nfc_beschrieben_lesen_8.pdf"

Quellenverzeichnis

Android Developers. (2012). *Beaming NDEF Messages to Other Devices*. Abgerufen am 28. Oktober 2012 von http://developer.android.com/guide/topics/connectivity/nfc/nfc.html#p2p, siehe beiligende CD: "Beaming NDEF Messages to Other Devices.pdf"

BR STen, S. (13. Juli 2012). *Stackoverflow*. Abgerufen am 19. Oktober 2012 von http://stackoverflow.com/questions/11471952/nfc-used-as-mifare-is-it-possible, siehe beiliegende CD:"http://stackoverflow.com/questions/11471952/nfc-used-as-mifare-is-it-possible.pdf"

Crider, M. (30. April 2012). *KegDroid dispenses beer via NFC tags*. Abgerufen am 6. November 2012 von Android Community: http://androidcommunity.com/kegdroid-dispenses-beer-via-nfc-tags-20120430/

DPA. (Januar 2012). *Girokarte mit NFC - Sparkassen führen kontaktloses Bezahlen ein*. Abgerufen am 23. Oktober 2012 von Computerwoche: http://www.computerwoche.de/netzwerke/mobile-wireless/2502512/

DPA/DAPD. (27. 05 2011). *Google Wallet - Bezahlsystem für Smartphones startet*. Abgerufen am 23. Oktober 2012 von Focus Online: http://www.focus.de/digital/handy/google-wallet-bezahlsystem-fuer-smartphones-startet_aid_631496.html

Fried, L. (27. April 2012). *NDEF*. Abgerufen am 25. Oktober 2012 von RFID/NFC Tutorial: http://www.ladyada.net/products/rfidnfc/ndef.html

Fried, L. (Mai 2012). *PN532 RFID/NFC Tutorial.* Abgerufen am 18. Oktober 2012 von http://www.ladyada.net/products/rfidnfc/wiring.html, siehe beiligende CD:"RFID_NFC Tutorial_wiring_ladyada.pdf"

Gallo, F. (2011). *NFC Tags - A technical introduction, applications and products.* Eindhoven: NXP Seminconductors.

Gernert, J. (23.. August 2012). Lass den Dollar stecken. *Die Zeit* , S. 26.

Jovanovic, M., & Muñoz Organero, M. (2011). Analysis of the Latest Trends in Mobile Commerce using NFC Technology. *Journal of Selected Areas in Telecommunications* , 1-12.

Keen, I. (April 2009). NFC Technology Overview. *NFC Forum Presentation at WIMA Conference* .

Langer, J., & Roland, M. (2010). *Anwendungen und Technik von Near Field Communication (NFC).* Berlin: Springer-Verlag.

Library von Seeed Technology Inc. (2011). PN532_SPI_V1. Beispiel: readMifareMemory. Siehe beiliegende CD: "Libraries/PN532_SPI_V1/readMifareMemory".

NFC Forum. (2012-07-11). *NFC Analog Specification. Technical Specification. ANALOG 1.0. NFCForum-TS-Analog-1.0.*

NFC Forum. (2011). *NFC in Public Transportation.* Wakefield: NFC Forum.

NFC Forum. (2011-04-13). *Type 1 Tag Operation Specification. Technical Specification. T1TOP 1.1. NFCForum-TS-Type-1-Tag_1.1.*

NFC Forum. (2011-05-31). *Type 2 Tag Operation Specification. Technical Specification. T2TOP 1.1. NFCForum-TS-Type-2-Tag_1.1.*

NFC Forum. (2011-06-28 a). *Type 3 Tag Operation Specification. Technical Specification. T3TOP 1.1. NFCForum-TS-Type-3-Tag_1.1.*

NFC Forum. (2011-06-28 b). *Type 4 Tag Operation Specification. Technical Specification. T4TOP 2.0. NFCForum-TS-Type-4-Tag_2.0.*

Nokia. (19. April 2011). *Adafruit.* Abgerufen am 23. Oktober 2012 von Benutzerhandbuch für Programmier von NFC Apps für Nokia Telefone: http://www.adafruit.com/datasheets/Introduction_to_NFC_v1_0_en.pdf

Nokia. (13. Juni 2012). *Understanding NFC Data Exchange Format (NDEF) messages*. Abgerufen am 25. Oktober 2012 von Nokia Developer Wiki: http://www.developer.nokia.com/Community/Wiki/Understanding_NFC_Data_Exchange_Format_(NDEF)_messages

NXP Semiconductors. (Rev. 1.2; 3. Mai 2011 a). *AN1304 - NFC Type MIFARE Classic Tag Operation. Application note. 130412 PUBLIC.*

NXP Semiconductors. (Rev. 1.3; 2. Oktober 2012 b). *AN1305. MIFARE Classic as NFC Type MIFARE Classic Tag. 130513 COMPANY PUBLIC.*

NXP Semiconductors. (Juni 2009 c). *First with industry-standard NFC IC*. Abgerufen am 20. Oktober 2012 von http://www.nxp.com/news/news-archive/2009/pn544.html, siehe beiliegende CD: "NXP_PN544_Chip.pdf"

NXP Semiconductors. (Rev. 3.2; 20. September 2012 d). *PN532/C1. Near Field Communication (NFC) controller. 120132 COMPANY PUBLIC.*

NXP Semiconductors. (Rev. 3.3; 16. Juli 2012 e). *PN533. Near Field Communication (NFC) controller. Product short data sheet. 158233 PUBLIC.*

NXP Semiconductors. (Rev. 5; 9. Oktober 2012 f). *UM10204. I2C-bus specification and user manual.*

Seeed Wiki. (4. August 2011). *NFC Shield.* Abgerufen am Oktober 2012 von http://www.seeedstudio.com/wiki/index.php?title=NFC_Shield#Features, siehe beiliegende CD: "NFC Shield Seeed Studio - Wiki.pdf"

Steffen, R., Preißinger, Jörg, Schöllermann, T., Müller, A., & Schnabel, I. (2010). Near Field Communication in an Automotive Environment. *Second International Workshop on Near Field Communication* (S. 6). München: BMW Goup Research and Technology.

TechShek. (3. Januar 2012). *Google Wallet Hack - Applying Card Emulation Patch to Android 2.3.4_r1 Source Code and Flashing it on Samsung Nexus S*. Abgerufen am 25. Oktober 2012 von

http://techshek4u.blogspot.in/2012/01/applying-card-emulation-patch-to_03.html, siehe beiliegende CD: "Card_emulation_Nexus_S.pdf"

Warnke, A. (2008). *Near Field Communication in Mobile Phones Using MIFARE Standard Access Control.* Stockholm: KTH Stockholm.