Master Thesis

Design and Implementation of a Power-Aware NFC Multi-Tag Field Strength Scaling Support for Mobile Devices

Bernhard Kipperer, BSc.

Institut für Technische Informatik
Technische Universität Graz

Reviewer: Ass.Prof Dipl.-Ing. Dr. techn. Christian Steger

Advisors: Dipl.-Ing. Norbert Druml, BSc.
Ass.Prof Dipl.-Ing. Dr. techn. Christian Steger

Graz, October 2012

Kurzfassung

In den letzten Jahren wurden sehr viele neue Technologien vorgestellt, die inzwischen omnipräsente mobile Geräte in deren Funktionalität erweitern sollen, Near-Field Communication (NFC) gehört auf jeden Fall dazu. Eingesetzt in Reader/Card Systemen ermöglicht die kontaktlose Übertragung von Daten mittels NFC viele nützliche Anwendungsfälle für Zutrittskontrollen und die elektronische Geldbörse. Dennoch fehlt es bisher an ausreichenden Energiekonzepten für mobile NFC Geräte. Zudem bieten populäre Betriebssysteme vor allem für Szenarios mit vielen NFC Tags, bekannt als *Multi-Tag*, noch keinerlei Unterstützung an. Die NFC-Technologie kann daher bei weitem noch nicht voll ausgeschöpft werden.

Kontaktlose NFC-Kommunikationen werden durch ein Lesegerät ermöglicht, das ein elektromagnetisches Feld erzeugt. Dieses durchsetzt entfernte passive Tags und versorgt sie dadurch mit der für den Betrieb nötigen Spannung. Die dabei aufgenommene Energie ist proportional zur der vom Lesegerät verwendeten Feldstärke, standardmäßig wird heutzutage stets mit maximalem Feld operiert. Diese Masterarbeit zeigt, dass diese Herangehensweise einerseits natürlich sehr ineffizient mit Energie umgeht, andererseits aber zudem auch grobe Probleme aufwirft. Vor allem in Situationen mit mehreren Tags innerhalb der Reichweite des Lesegerätes kommt es zu einer unerwünschten gegenseitiger Beeinflussung der Tags.

In dieser Masterarbeit wird einerseits eine Unterstützung für Multi-Tag Applikationen im weitverbreiteten Betriebssystem Android entwickelt, wovon viele verschiedene Geräte auf einmal profitieren können. Andererseits wird der relativ neue Ansatz des *Field Strength Scalings* genutzt, um die geringste nötige Feldstärke die ausreicht, mit einen gewissen Tag zu kommunizieren, zu bestimmen.

Es werden mehrere optimierte Algorithmen, die auf dem Prinzip des *Field Strength Scalings* aufbauen, entwickelt und mit dem bisher standardmäßig verwendeten Ansatz verglichen, um das Potential von *Field Strength Scaling* zu demonstrieren. Es kann nicht nur wertvolle Energie einspart werden, zusätzlich sind auch viele interessante Applikationen erst dank *Field Strength Scaling* möglich.

Abstract

In the last few years, many new technologies were introduced to enhance the omnipresent mobile devices; Near-Field Communication (NFC) is clearly one of them. Especially if used in reader/card systems the contact-less data transmission of NFC enables valuable use cases for access control and electronic payment. However, elaborate energy concepts for mobile NFC devices have not been designed yet, and missing support in popular operating systems, especially for multi-tag scenarios, prevents from exploiting the technology to its full potential.

NFC communications are established by a reader generating an electromagnetic field that penetrates a distant passive tag, in order to power it. The consumed energy is proportional to the field strength used by the reader, which by default is set to the maximum level. As will be shown in this thesis, operating that way on the one hand is very inefficient energy wise, but on the other hand even causes sever issues. Especially in scenarios where many different tags are present inside the communication range, undesired mutual interference occurs.

This thesis on the one hand develops support for multi-tag applications in the widespread operating system Android, therefore enabling a broad range of devices to benefit from the developments. On the other hand it also uses the relatively new approach of field strength scaling in order to find the least amount of field strength sufficient to reach a certain tag.

Multiple optimized algorithms based on that principle are developed and compared against the standard approach to demonstrate the potential of field strength scaling. Besides saving valuable energy, field strength scaling even enables use cases that otherwise would remain impossible.

Acknowledgment

I would like to thank Ass.Prof. Dipl.-Ing. Dr.techn Christian Steger for giving me the opportunity to once again make a contribution to the exciting research area of Near-Field Communication. I am very grateful for the pleasant support, shared knowledge and especially the constructive feedback I received so many times from my very capable advisors Norbert Druml and Manuel Menghin. Finally, my greatest gratitude belongs to my family and relatives, my friends and colleagues, and whoever may have supported me along the way. You all have played your part to guide me through life.

Graz, October 2012 Bernhard Kipperer

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....
date (signature)

Contents

List of Figures

List of Tables

Nomenclature

ACS	Advanced Card Systems
APDU	Application Protocol Data Unit
API	Application Programming Interface
BIN	Binary Search Algorithm
CIU	Contactless Interface Unit
CONST	Constant Field Strength Algorithm
CPU	Central Processing Unit
DFVS	Dynamic Frequency and Voltage Scaling
EEPROM	Electrically Erasable Programmable Read-Only Memory
FRI	Forum Reference Implementation
FSS	Field Strength Scaling Algorithm
I/O	Input/Output
META[:SEC:]	Mobile Energy-efficient Trustworthy Authentication Systems with Elliptic Curve based Security
NFC	Near-Field Communication
NI	National Instruments
NMOS	N-Channel Meta Oxide Semiconductor Field-Effect Transistor
NXP	Next eXPerience Semiconductors
PMOS	P-Channel Meta Oxide Semiconductor Field-Effect Transistor
PoC	Proof Of Concept
RF	Radio Frequency
RFID	Radio Frequency Identification
SD	Secure Digital Memory Card
SDD	Single Device Detection
SDK	Software Development Kit
TI	Texas Instruments
UHF	Ultra-High-Frequency
UID	Unique Identifier
USB	Universal Serial Bus

Chapter 1

Introduction

Ever since computers were invented, the main goal of research was to increase their execution speed. Aside from that engineers specialized on only a few other aspects of computing for many decades, trying to reduce the size of the hardware and developing new technologies that are cheaper to produce [SS03]. During the course of the last years it became obvious that many important issues at the other side of the field were missed altogether. The massive heat generation inside modern hardware, caused by high power dissipation, and sever energy supply problems, brought about by the gap in the battery development. As described in [PFW11], external power supplies have progressed very slowly in the past, by no means matching the rate at which modern processors evolve. Given that no outstanding new battery technology has been discovered yet, engineers today are more than ever forced to develop new approaches towards hardware and software design.

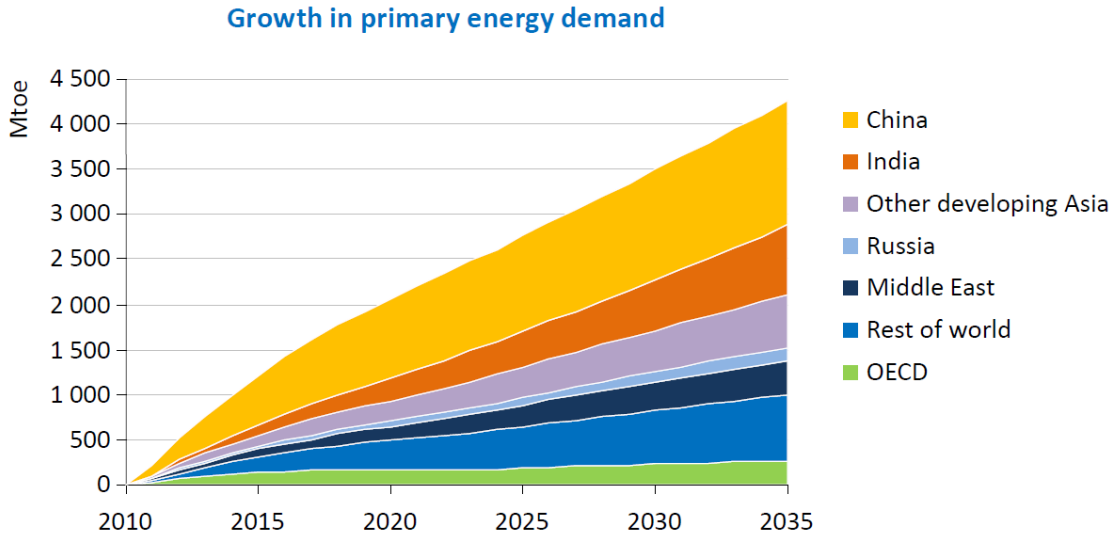


Figure 1.1: World energy demand according to the World Energy Outlook 2011 [IEA11]

Power-aware computing deals with the issues described above. Devices are investigated in order to reveal possible ways of saving valuable energy. Especially one type of modern computers really benefits from developments in this field, mobile hand-held devices that

are omnipresent today. Given these products have to rely on portable power supplies, a lot of research is conducted to enhance their battery life. Nevertheless, the number one approach is still to waste as little energy as possible in the first place, therefore power-aware computing tries to find new ways to improve the overall energy management and design hardware and software in a power-efficient way.

Near-Field Communication

As part of the contemporary trend towards pervasive computing, more and more mobile devices are put on the market, paving the way for many new products that are specifically devised for portable use. Additionally, new technologies are developed to enhance the possible applications of mobile devices. Among these, one exiting new invention called *Near-Field Communication* (NFC) is becoming more important every single day. This master thesis deals with NFC on various levels, develops new and convenient software to exploit the technology's full potential, and finally associates the research with power-aware design. NFC enables wireless communications between two devices over several centimeters, supporting three different operational modes: A reader may generate an electromagnetic field that represents the power supply for a distant tag including some data that can be acquired contactlessly through the field. Two NFC devices may both actively generate fields and communicate peer-to-peer with each other. Or finally, one device may act as if it was a real smart card and every compatible reader can acquire its data in the usual way, given that emulated cards cannot be distinguished from real ones.

1.1 Objectives and Motivation

This master thesis is a part of the superior *META[:SEC:]* project[1], conducted by the Institute of Technical Informatics at TU Graz. META[:SEC:] deals with power and security evaluation on the system level. This thesis covers NFC devices and their associated applications in a power-aware context. Multiple NFC components like reader and contact-less cards known as tags are utilized. New software algorithms are developed and their energy consumptions are measured in order to optimize the whole system and provide new approaches towards power-aware NFC applications. Two different steps are taken and the following components are developed,

- two different algorithms are introduced and their energy consumptions in multi-tag scenarios are measured. All of their characteristics are analyzed and compared with the standard approach of today, in order to save as much energy as possible,
- the algorithms are integrated into the Android operating systems which allows it to access multiple NFC tags inside a single field. This feature will become increasingly important in the future, but is still totally unsupported by most operating systems, including Android.

At first, a new functionality to be called *multi-tag* is developed. It deals with the scenario of multiple NFC tags being inside a single magnetic field of a reader. Although

[1] META[:SEC:] is short for *Mobile Energy-efficient Trustworthy Authentication Systems with Elliptic Curve based Security* and is sponsored by *FIT-IT contract 829586*.

basic software libraries for communications with tags do exist, they usually only support one single tag to be communicated with. The new multi-tag functionality presented in this thesis remedies that shortcoming.

Subsequently, different field strength scaling algorithms are developed to enable power-aware approaches towards NFC multi-tag applications and all of them are based on the functionality described above. Given that the consumed power of NFC readers is proportional to the field strength used while communicating with the tags, valuable energy can be saved by decreasing the field strength. The smart algorithms to be presented in this thesis use different approaches to figure out the least amount of field strength necessary to reach a specified tag and therefore demonstrate multiple useful ways to effectively save energy. This is very relevant to many of today's NFC applications, because up to the present, in most real-life scenarios NFC readers always communicate with the maximum field strength. Originally done to ensure a stable communication as discussed in [DMS+12], it leads to sever energy waste, given that in many cases a small amount of the available field strength would by far be sufficient. Additionally, as this thesis will show, even with lesser field strengths a stable communication can be achieved, rendering the number one argument for using the maximum field mute.

1.2 Structuring

This document is structured into the following chapters:

Chapter 2 surveys the fundamentals of power-aware computing, describes the physical principles of NFC, discusses the foundations of field strength scaling and its inherent issues in multi-tag situations and also presents existing use cases that are based on NFC technology. In conclusion all hardware as well as software components that already existed and upon which this thesis' developments are based on are discussed.

Chapter 3 demonstrates the complete underlying design of this thesis and explains how Android has to be enhanced in order to integrate all the new functionalities. Furthermore, the three different algorithms, two of them using field strength scaling to save energy and the third being today's standard approach that operates with the maximum field only, are introduced. Their consumptions are compared and in depth analyses for all algorithms are carried out that discuss their best, average, and worst case behaviors.

Chapter 4 demonstrates the different implementation steps done during this thesis that led to the final result. All used software packages and tool-chains are presented and the actual implementation of the algorithms in the target platform is discussed.

Chapter 5 presents the measurement results acquired for this thesis, evaluates and analyzes them. Various realistic use cases are executed and the behavior of each algorithm is discussed. On the one hand the limits of the NFC technology are inspected, on the other hand the great potential of multi-tag field strength scaling in current commercial applications is shown as well.

Chapter 6 summarizes the insights gained in this thesis. Additionally, it also takes a look out to future works and describes how they can benefit from the achievements presented in this document.

Chapter 2

State of the Art

2.1 Basics

2.1.1 Power-Aware Computing

Power-aware computing deals with various approaches towards hardware and software development in order to save valuable energy and to design efficient new products. As described in [SS03], the field gained more and more attention over the last few years, initiated by a massive trend towards mobile applications and pervasive computing that is still in full progress. The best strategy to save power without a doubt is to investigate wasteful parts of a system and to improve their energy management. Two very important realizations of power-aware computing are the facts that on the one hand many systems require different amounts of energy depending on the state they currently reside in, and on the other hand each part of a system is obligated to finish a different amount of work in a certain time period. This enables two general approaches to be discussed in the next section, frequency as well as voltage scaling.

Dynamic Frequency and Voltage Scaling (DFVS)

The scaling of a processor's frequency or its voltage enables sophisticated power saving modes. As described in [LMD+09], DFVS varies the provided power and slows down the execution speed, depending on the current state of a processor and its requirements at that moment. Various portable devices used today are already capable of changing the clock rate of their CPUs. This results in an additional benefit, if applied for example to a notebook. While frequency scaling on the one side decreases the required power, on the other side it also reduces the noise level of fans or similar equipment necessary to cool the CPU. Lower clock rates result in less heat generation, therefore fans can run at lower speeds and remain much quieter. Nowadays, most of the time frequency scaling is used alongside voltage scaling. As can be seen in Equation (2.1) that describes the consumption of CMOS transistors with the switched capacitance C, the wasted power P is connected linearly with the frequency but quadratically with the provided voltage, thus making a combination of voltage and frequency scaling even more efficient.

$$P \approx C \cdot V^2 \cdot f \tag{2.1}$$

Unfortunately, both techniques, even when put together, have certain limits. The supply voltage, as discussed by [ZBSF05], cannot be decreased to any desired value, given that all electric components require a minimum amount of voltage to work accordingly and lower levels risk the possibility of logical signals not being interpreted correctly anymore. At some point the control signals will even disappear in the surrounding noise. The clock rate of processors on the other hand can only be decreased as long as the computational speed still remains sufficient for any given application. Especially real-time devices cannot afford to lose much speed as they rely on a certain throughput they have to meet. The given scenario and the requirements at that moment limit how much the clock rate can be decreased without deteriorating the system's performance too much.

Field Strength Scaling

Field strength scaling represents a totally different approach to save energy and was especially developed for use with systems that rely on electromagnetic fields for communication. Instead of decreasing a system's supply voltage, risking issues with logic levels and low signal to noise ratios, or minimizing its clock rate which leads to lower execution speed often also very undesired, field strength scaling alters the actual amount of field generated by the reader when communicating with the targets.

As discussed in [DMS+12], most NFC devices that use electromagnetic fields for contact-less communications nowadays usually always operate with the maximum possible field strength, simply to guarantee at stable communication between the initiator and its targets. This may seem to be a good idea in cases where the contact-less devices require a lot of power, cryptographic operations, or other complex computations come to mind. In most standard use cases however, operating at full field is very wasteful and furthermore as can be seen in Figure 2.1, any superfluous electrical power that reaches the target is cut off by a protective zener diode and therefore simply lost.

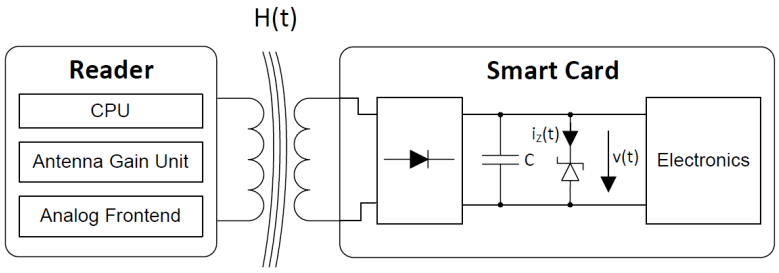


Figure 2.1: The structure of a reader/smart card system [DMS+12]

Fortunately most readers used in contact-less systems contain transmission modules that allow controlling the actual field strength used for communications. That part is known as the *antenna gain unit*, also shown in Figure 2.1, and usually contains nothing more than serial resistances that can be set via several internal registers of the transmission module. By increasing their values, the voltage across the antenna is reduced and as a result less field is generated. This of course also leads to far less power consumption inside the reader, which is especially important for portable devices that solely rely on batteries. It increases their life time and allows longer periods of operation without the need to spare

any of the original functionality. This realizes a power-aware approach that blends in with existing methods and does not require any hardware changes to be made at all.

Which amount of field strength to use in what situation depends on various factors. How to choose just the right field strength in any occasion represents one of the main topics discussed in this thesis in the following chapters. Additionally, it will be shown that reducing the field strength paradoxically widens the possible transmission range and more targets can be found with less field.

2.1.2 Principles of Radio Frequency Identification

Radio Frequency Identification (RFID) is a well established technology and the foundation for NFC, which therefore shares a lot common characteristics. This section offers an overview of the physical principles behind RFID, while more details of NFC will be covered in the following sections.

RFID systems are commonly used for identification and tracking of goods in an entirely contact-less way. As described by [GZ11], they consist of two main components, a transponder or tag attached to an object to be identified and a reader that provides power to the tag's microchip and reads its contents. According to [QC10], especially the possibility of identifying objects from a distance represents a huge advantage of RFID systems over other approaches like printed bar codes that rely on optical scanners. The first experiments, which subsequently lead to RFID systems several decades later, were carried out during world war two. As described in [Lan05] and [MSW08], in the beginning development was based on earlier researches regarding the Radar technology that was commonly used during the war. Even today many RFID systems employ a method called *backscattering* that was directly inherited from the functional principle of Radar. During the 1960s and 1970s in depth research lead to advanced technologies that finally realized the first sophisticated RFID systems used for commercial purposes in the 1980s. All modern devices are based on these preliminary works.

The authors of [Fin10] and [Res10] describe three different types of RFID systems that exist today, categorized depending on the communication ranges that can be achieved.

Close Coupling Systems

Close coupling RFID systems usually only provide ranges up to one centimeter and are mostly used in security-related applications such as access control at entrances. In these scenarios the very limited communication range is desirable as it prevents possible abuse and also simplifies the required RFID hardware. Over short distances considerably less energy is lost, thus making it possible to use common hardware instead of acquiring expensive low power devices.

Long Range Systems

RFID systems that provide ranges of several meters are called long range systems. Communications are carried out over UHF (Ultra High Frequency) and microwave frequencies between 868 MHz and 5.8 GHz, depending on country regulations. The already mentioned principle of backscattering, very similar to Radar, is used to provide a link between reader and tags.

Remote Coupled Systems

Remote coupled RFID systems were developed as a middle ground between the two other categories described above. Communications up to one meter are possible, these systems rely on the principle of inductive coupling that will be described in the following section. Most of today's RFID devices belong to this category, likewise the NFC devices that will subsequently be discussed in the following sections.

2.1.3 Principles of NFC

RFID as well as NFC readers use an inductor carrying electric current that produces a high frequency magnetic field. As explained in [Res10] and [CH07], every current that flows in a conductor is accompanied by an electromagnetic field. The strength of said field depends on the current and decreases with distance from the conductor. The magnetic field strength H describes that physical property. Figure 2.2 shows a straight, current-carrying conductor and the resulting magnetic flux lines.

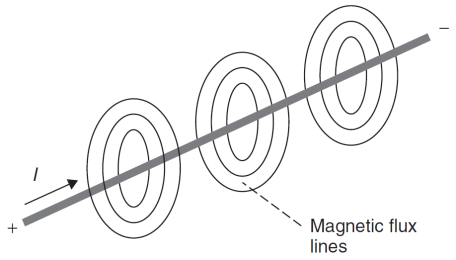


Figure 2.2: A straight current-carrying conductor and its magnetic flux lines [Fin10]

Equation (2.2) defines the *magnetic field strength* H of that simple conductor, with I being the current through the conductor and r the distance from it.

$$H = \frac{I}{2\pi r} \tag{2.2}$$

Additionally, the *magnetic flux density* B was introduced to cover the various magnetic materials. It is connected to the field strength H as shown in Equation (2.3).

$$\boldsymbol{B} = \mu_0 \mu_r \boldsymbol{H} \tag{2.3}$$

The magnetic field generated by the reader's coil spreads over wider ranges and therefore also penetrates a distant tag. Whatever part of the field passes through the tag is called *magnetic flux* ϕ, a physical quantity that depends on the pervaded area, as shown in Equation (2.4).

$$\phi = \boldsymbol{B}\boldsymbol{A} \tag{2.4}$$

Magnetic flux passing through the tag's coil induces a certain voltage V. Faraday's law describes this scenario, known as *induction through a changing magnetic flux*. The flux density B in Equation (2.5) is noted as $B(r, t)$, with r being the distance, and t the

additional dependency on the time in this scenario. Equation (2.5) shows how the flux ϕ is calculated in the case of a changing magnetic flux density B.

$$\phi = \int_A \boldsymbol{B}(\boldsymbol{r}, \mathrm{t})\ d\boldsymbol{A} \tag{2.5}$$

Finally, Equation (2.6) contains Faraday's law of *induction through a changing magnetic flux*. A voltage V is generated that may be used as a power supply for the tag.

$$V_{ind} = -\frac{\partial \phi}{\partial t} \tag{2.6}$$

Figure 2.3 demonstrates the principle described above. The reader as well as the tag are said to be inductively coupled. Figure 2.3 also shows a parallel capacitor C_r, connected in the reader, as well as another capacitor C_1 on the part of the tag. These capacitors are used to form a parallel resonant circuit and the tag has to be tuned to the reader's frequency in order to achieve maximum induced voltage. Section 2.1.7 will describe issues that occur whenever tags, caused by mutual interference, are detuned and the voltage induced is not sufficient to successfully power all their internal electronic components anymore.

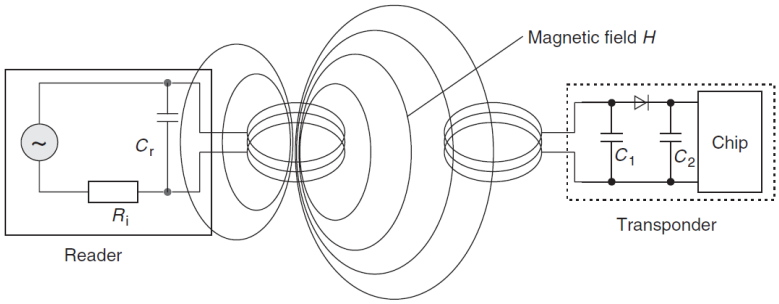


Figure 2.3: A RFID transponder powered by induction [Fin10]

As described in [Wan11], RFID systems employ various different frequencies from low (LF) to ultra high (UHF), depending on the respective use case. The NFC technology however is designed to work within a fixed high frequency band of 13.56 MHz instead. According to [Res10], ranges of a few centimeters and different data rates of 106, 212, or 424 Kbit/s can be achieved. RFID as well as NFC systems can either be built upon active tags using additional batteries as power supplies, or passive tags that only rely on the electromagnetic field provided by the reader. Regardless, the generated field is always used to transmit data in both directions as well. According to [CKK+07], readers apply an *Amplitude Shift Key* (ASK) that modulates the field to send data to a tag or another device. Tags on the other hand rely on one out of two different techniques to return data back to the reader, *load modulation* or *backscattering*. Given their limited communication ranges of a few centimeters, NFC systems only support the former approach, whereas RFID long range systems also employ backscattering.

Load Modulation

Load modulation, as discussed in [Fin10], makes use of the fact that both the reader and the tag are inductively coupled. In the eyes of the reader, a tag consuming parts of the energy provided by the electromagnetic field is seen as a sort of transformed impedance. By adding a load transistor to the tag and switching it on and off the said impedance can be varied. This also changes the voltage sensed at the reader's antenna coil, thus allowing the tag to directly encode data inside the electromagnetic field by switching the transistor on and off, depending on the data bytes to be sent. No additional power supply attached to the tag is necessary in this scenario. The process of load modulation can be seen in Figure 2.4.

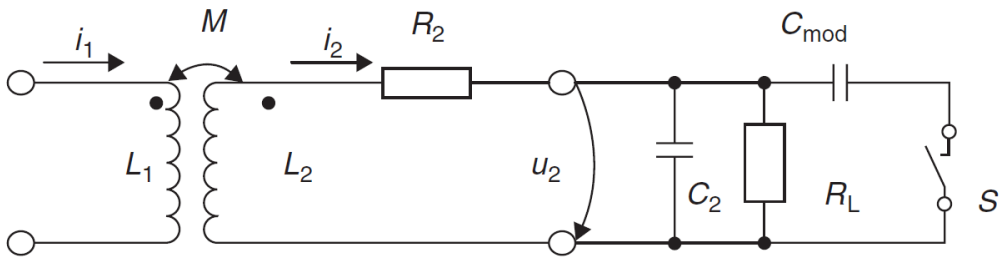


Figure 2.4: Modulating the load of the tag by opening and closing the switch S [Fin10]

Backscattering

Backscattering, as described by [GZ11] and [Fin10], is a communication technique for long range RFID connections over several meters and is very similar to the principle of the Radar technology. Radio waves are usually reflected by solid objects and a certain amount of the wave returns to its origin. In RFID systems an additional load resistor is used to control how much of the power arriving at the tag is to be reflected back to the reader. This enables the tag to vary the amplitude of the power that subsequently will return at the coil of the reader, thus allowing the tag to encode and transmit data by modulating the backscatter accordingly to the data bytes to be sent.

The reflected power consists of one carrier signal and two sidebands. The modulation only affects the sidebands, allowing the reader to easily distinguish between the carrier part and the side bands that contain the required data. Figure 2.5 shows the power levels of the reflected wave that arrives at the reader. The carrier signal the reader himself produces is seen at center, the reflected carrier signal can not be seen anymore, because its level in comparison is very low. The modulated sidebands that contain the data sent back from the tag are visible on both sides of the carrier.

2.1.4 NFC Operational Modes

According to [NXP07b] and [MLKS08], NFC provides three different operational modes called reader/writer, card emulation, and peer-to-peer.

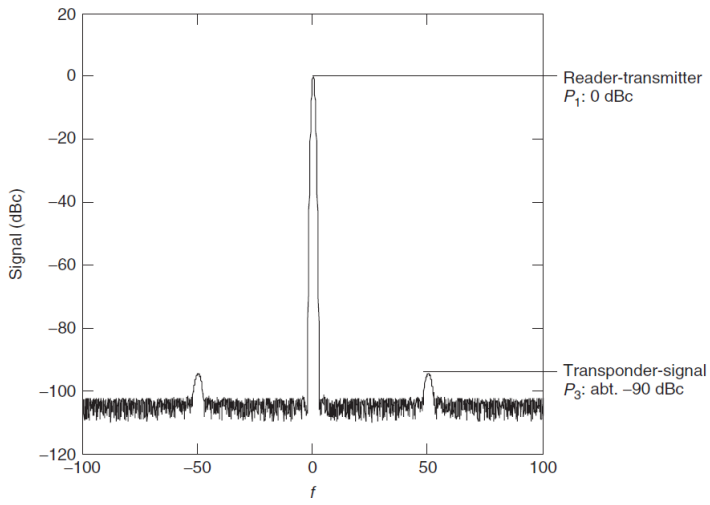


Figure 2.5: Reflected waves at the reader [Fin10]

Reader/Writer Mode

The reader/write mode enables NFC devices to read from or write to passive tags. As already mentioned in a previous section, every communication with tags is carried out by having the reader create an electromagnetic field that on the one hand supplies power by means of induction but on the other hand also allows the tag to respond to any request by load modulating the field accordingly.

Card Emulation

Many NFC devices, like smart phones for example, include an additional hardware part called *secure element* that supports convenient storage and encryption of sensitive data. This allows NFC devices to emulate smart cards for payment and the like. The card emulation mode provides a convenient way of carrying multiple such cards inside a single NFC device. Readers cannot distinguish a smart card emulated by NFC hardware from a real one. This renders any hardware modifications on the part of readers unnecessary.

Peer-To-Peer

The peer-to-peer mode allows one NFC device, the initiator, to connect to another device, which acts like a tag, and exchange data with it. Whenever one device senses the field generated by another one and also receives the appropriate commands, it starts to behave like a tag. Subsequently, after a transmission in one direction is completed, both devices switch their roles and whoever acted as a tag now becomes the new initiator.

2.1.5 NFC Interface and Protocol

The NFC interface and protocol, also known as *NFCIP*, is situated above the physical layer that was described in the previous sections. The general protocol flow of NFC devices is strictly determined by the technical specification found in [ECM08].

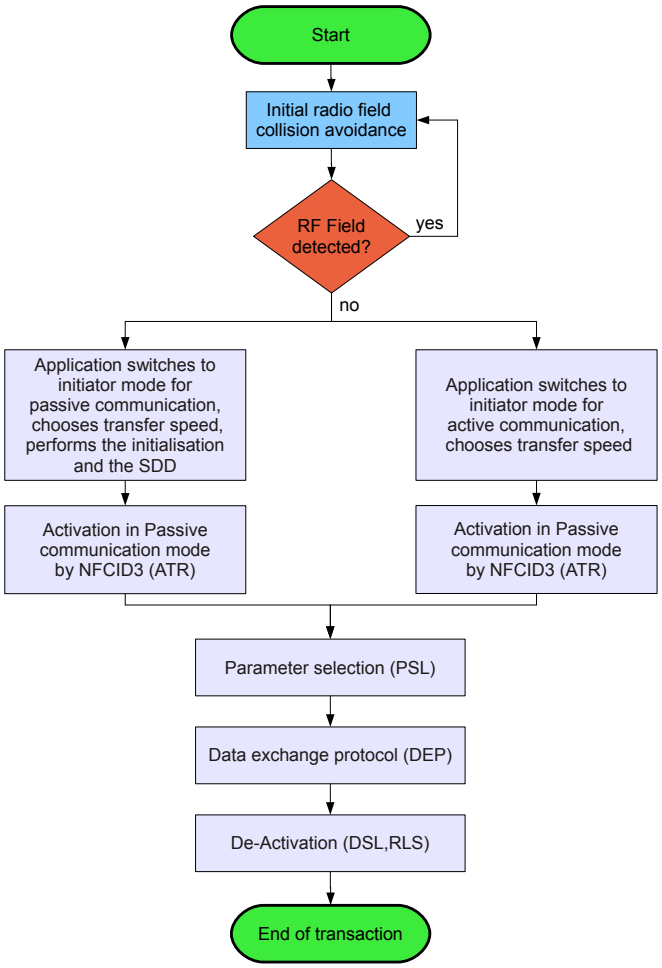


Figure 2.6: The different steps of activating a NFC device as described by [ECM08]

Every NFC device starts up in *target mode*, acts like a tag and remains passive while waiting for an external field to supply power. An application using the NFC device has to settle on one out of three possible transfer speeds and has to choose whether the device should communicate in an active or passive way. A NFC device always remains in its target state until an application using it switches it to *initiator mode*. This state requires an additional step known as *anti-collision* or *initial radio field collision avoidance* as seen on Figure 2.6. It is used right before a field is generated by the device. A test is carried out to see if any other external magnetic field is present in the range. In this case the device is not allowed to turn on its own field, as this would disturb other transmissions taking place in the proximity. Finally, after the collision avoidance test was successfully carried out, the application can use the device to send commands or data bytes and wait for another device to respond in the same speed and transmission mode originally agreed upon. This whole process is shown in Figure 2.6 which is based on the descriptions found in the technical specification [ECM08].

2.1.6 Multi-Tag Applications

A scenario especially interesting for this master thesis is the presence of multiple tags inside the near-field of a reader. An additional technique called *Single Device Detection* (SDD) has to be added to the protocol flow described above, to select one out of many targets to communicate with. Whenever an initiator of a NFC communication starts to request data from another device, each target reachable inside the field answers at the same time. This of course raises the problem of collisions. As defined in [ECM08], each NFC device has its own specific number called *UID* (unique identifier), which is sent to the initiator either entirely or just partly, depending on the current step of the multi-staged SDD process.

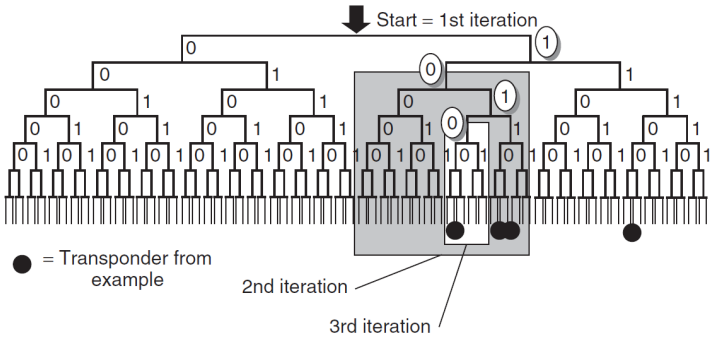


Figure 2.7: Single Device Detection (SDD) [Fin10]

The initiator starts by issuing a command that forces all targets inside the field to answer at the same time and return their entire UID. If more than one target is present a collision happens. All bits are transmitted using the *Manchester code*, demonstrated in [Fin10], which stores the value of a bit in terms of a negative or positive transition. Whenever different transitions arrive, transmitted by two targets, they channel each other out, leading to a state not allowed by the Manchester system. The initiator therefore is able to find the first bit inside the returned UIDs where a collision occurred. In the next step the initiator announces all n valid bits found before the collision and only those targets that have UIDs with n bits equal to the valid bits answer. The number of targets to respond is already narrowed. This process is repeated until only one target remains. By executing a kind of binary search the initiator has found one complete and unique UID and can establish a communication with the respective target, while all others remain silent. If another target should be contacted afterwards, the initiator simply starts additional SDDs. Figure 2.7 demonstrates the process described above.

2.1.7 Mutual Interference between Tags

Whenever more than one tag is present inside the near-field of a reader, which of course is the case in the multi-tag scenarios discussed in this document, several malicious effects occur that corrupt communications or even prevent any NFC operation at all. The authors of [WW08] have inspected these effects more closely and are able to provide explanations.

As discussed before, NFC tags are powered through the field generated by the reader.

Magnetic flux which penetrates their coil induces a certain voltage that can be used as a supply, although taken by itself, this voltage is far too small to operate any electronics. Therefore, as described in [Fin10], another physical effect known as *resonance* and the associated *voltage step-up* have to be aimed at. The tag contains an additional capacitor C_1, which together with the coil forms a parallel resonant circuit tuned to the readers transmission frequency.

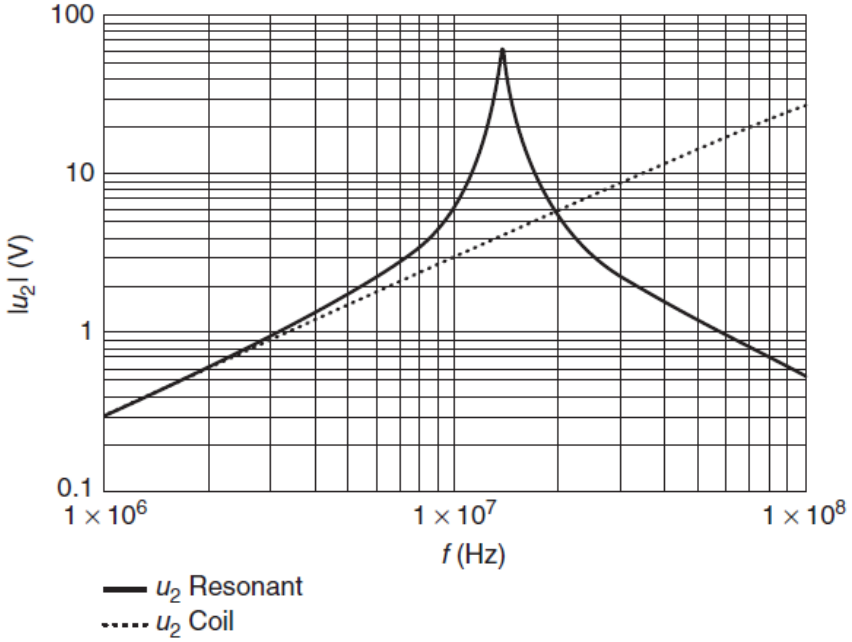


Figure 2.8: Voltage step-up at the resonance frequency of 13.56 MHz [Fin10]

As can be seen in Figure 2.8 whenever the frequency of the field provided by the reader comes close to the tag's resonance frequency, a step-up inside the tag occurs, which increases the voltage available across its coil by more than ten times. This amount finally is sufficient to supply the tag with enough power to operate correctly. For this step-up to work however, as can be seen in Figure 2.8, the frequency has to remain within a certain small band, lower frequencies only result in the original amount of voltage being available, whereas higher frequencies even drastically decrease what voltage is available to the tag.

While these facts do not impose much of a problem in single tag scenarios, in multi-tag applications however the more tags present at a given time, the more their coils detune each other. As a result, the resonance frequency of their resonant circuits no longer matches the frequency provided through the field, a step-up not as effective or no step-up at all occurs and the voltage being available to the tag is not sufficient anymore.

Figure 2.9 demonstrates how the voltages inside the tags behave, if one tag A is close to the reader R, while another tag D approaches both. It can be clearly seen in the right part of the picture how the voltages of both tags drastically decrease, when D comes closer to A. This is the main reason, why multiple tags inside a small area present a huge problem. They detune each other which prevents them from acquiring enough power through the

field. As a consequence only few or even none of the tags can react to commands anymore. Therefore, one of the main goals of this thesis, beside saving energy, is to develop new approaches that circumvent these issues and realize multi-tag applications that actually work, despite the malicious effects of detuning.

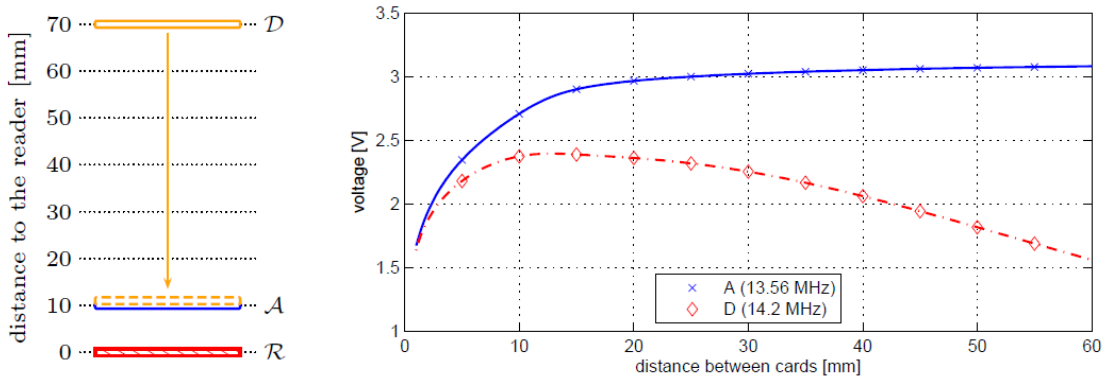


Figure 2.9: Decreasing supply voltage due to detuning [WW08]

2.2 Related Work

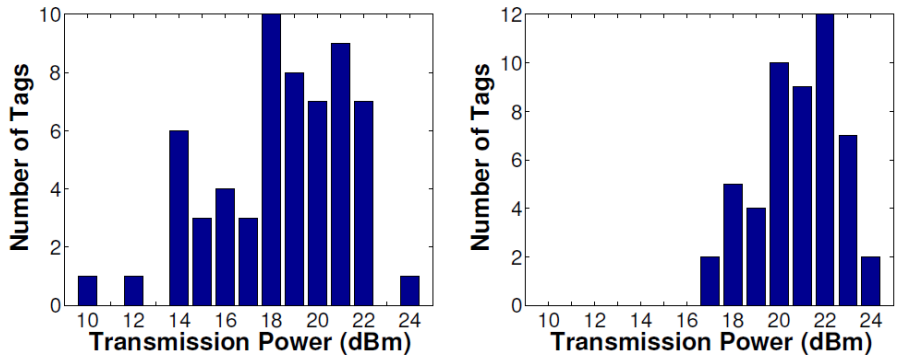
2.2.1 Power Management in RFID/NFC Systems

Although the RFID technology is well established for decades, many new studies covering the field are still being regularly released. Over the last few years, the number of portable RFID readers that rely on battery technology increased significantly. As a result, elaborate approaches to actively reduce their over-all power consumption have to be found.

The authors of [XGW+11] present a new algorithm named *Automatic Power Stepping* (APS) to be run on a RFID reader. It approximately determines the number of tags to be read at a certain moment and subsequently scales the reader's field in order to provide the least amount of power sufficient to successfully supply the tags, while also reducing the number of collisions that occur.

In common RFID applications, many different distributed tags are utilized that reside in various locations, whereas the actual field strength required to reach certain tags depends on their actual positions. By incrementally stepping through different power settings of the reader, the authors of [XGW+11] were able to inspect the responsiveness of the tags. Figure 2.10 demonstrates that independent of whether the tags are randomly placed or positioned on a flat plan parallel to the reader's coil, they can be grouped to subsets regarding the field strengths required to reach them.

Additionally, through repeated measurements the authors of [XGW+11] have revealed that RFID tags can reside in three different states. An inactive state where the tag is insufficiently powered by the reader, a stable state that enables the tag to successfully reply to any request, and additionally a third state called *lossy* that has been ignored by most studies so far. It describes a scenario where a tag on the one hand is supplied with enough power to respond to the reader, but on the other hand cannot guarantee any communication to remain uninterrupted. It was shown by [XGW+11] that a tag receiving



(a) Tags are randomly placed

(b) Tags are placed on a flat plane

Figure 2.10: The amount of tags reached at different field strength settings [XGW+11]

a power, which level is just a little above the threshold between the lossy and the stable state, can actually successfully communicate with the reader. Additionally, as soon as a tag is in stable state, its performance regarding the response rate already is optimal. Therefore, instead of using the maximum amount of power a reader provides, it is sufficient to operate with tags at a certain power P_{stable}, just above the threshold between the lossy and the stable state. Based on the discovery described in the last section, tags can be grouped into subsets by gradually scaling the used field strength of the reader. As a result, only one subset of tags is reached in each step, which of course reduces the amount of possible collisions between several tags. The APS algorithm, discussed in this section, is executed directly on a reader and is able to successfully save as much as 60% of energy of a RFID system. The following sections present two different power-aware approaches that aim at the NFC technology, which of course is especially relevant to this thesis.

The authors of [MDS+12] have introduced a new power-aware strategy called *PTF-Determinator* that inspects various characteristics of a reader as well as a tag and the physical relation between both components. As a result, the *Power Transfer Function* (PTF) is determined which allows to dynamically adapt the power transfer during runtime, dependent on the current needs. As shown in Figure 2.11, the contact-less communication between a reader and a tag can be described as consisting of two different channels, a power transfer as well as a data transmission channel. As explained before, power is transfered to passive NFC tags via inductive coupling. This principle can be represented by the PTF, which subsequently allows to decide which field strength to use at a certain moment in order to save as much power as possible. To determine the PTF however, several parameters need to be known beforehand. The characteristics of all coils involved, their physical relationship as well as the reader's properties have to be inspected. Subsequently, the data transmission channel allows to acquire important information about the involved tag, especially to inspect whether it receives sufficient power to be operated as desired. Based upon the collected data it is possible to determine the PTF and enable the reader to scale its own field strength accordingly. An enhanced NFC system using this process can be seen in Figure 2.11.

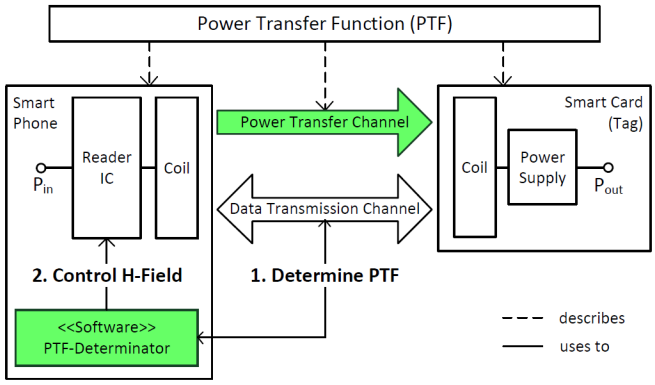


Figure 2.11: The *PTF-Determinator* used inside a NFC system [MDS+12]

The PTF-Determinator method is executed during tag discovery. Although, as a result of the additional logic, establishing a communication takes about four times as long, an average of 13% of energy can be saved compared to a standard tag discovery process. Figure 2.12 demonstrates the actual power profiles of both the standard procedure as well as the PTF-Determinator method.

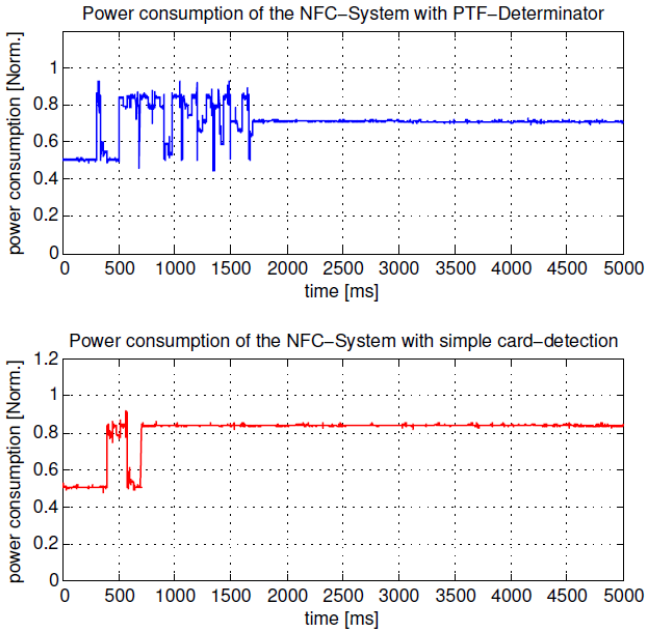


Figure 2.12: The optimized power consumption of the PTF-Determinator [MDS+12]

A different approach at saving power in NFC devices is proposed in [DMS+12]. A new methodology named *Adaptive Field Strength Scaling* (AFSS) is introduced, which actively alters the reader's field strength dependent on the instantaneous power requirements of a certain smart card. Whenever special operations have to be carried out that consume very much power, for example cryptographic computations, the smart AFSS approach scales the field strength of the reader accordingly in order to supply the tag with sufficient power.

During idle times and operations that are more power efficient, the used field strength can be heavily reduced instead.

AFSS provides two different methods to save power. On the one hand, the algorithm can use a *smart card request power model* which assigns a certain power consumption to each smart card operation in order to create an approximated model. On the other hand, the smart card itself can examine its own instantaneous power requirements and notify the reader, which subsequently scales its field strength accordingly. While the former approach only requires some additional logic that can be implemented solely in software, for the latter of course both the reader as well as the tag hardware has to be extended.

The software-based power model approach requires the actual power consumptions of different smart card operations to be measured beforehand. Subsequently, that knowledge is either integrated into a reader's firmware or the tag itself. A power model approach with logic integrated into the reader is shown in Figure 2.13. A request r is the input of the *power model firmware component* which estimates the power required by a tag at that moment. The reader subsequently scales its field strength by using the antenna gain unit.

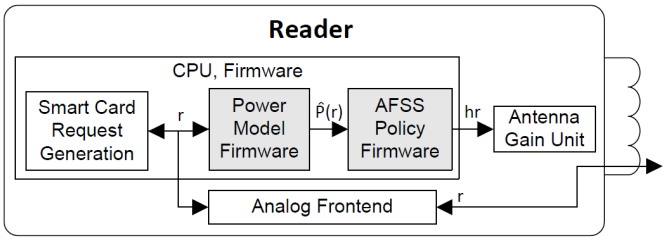


Figure 2.13: AFSS based on a power model integrated into the reader [DMS+12]

A different approach integrating the power model into the smart card is shown in Figure 2.14. It improves the estimation of the power consumption as the smart card has access to several relevant values like its current internal state or the supply voltage. Therefore, a much finer grained field strength scaling is possible. Whenever the current setting is not sufficient anymore, the tag uses a certain message which prompts the reader to scale the field strength as desired.

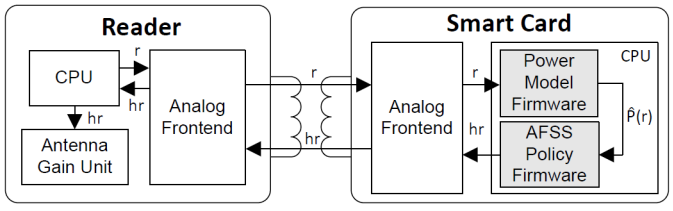


Figure 2.14: AFSS based on a power model integrated into the smart card [DMS+12]

The second, more complex method presented in [DMS+12] is called *Instantaneous Power Consumption-Based AFSS*. It requires hardware to be extended as demonstrated in Figure 2.15, both the reader as well as the smart card has to include additional AFSS components. The one great advantage of this hardware-based approach of course is the unmatched execution speed compared to software-based implementations. Similar to the

strategy described before, the smart card monitors how much voltage is currently available and instantly informs the reader whenever more is required. The reader receives these messages and subsequently scales the field strength through its antenna gain unit.

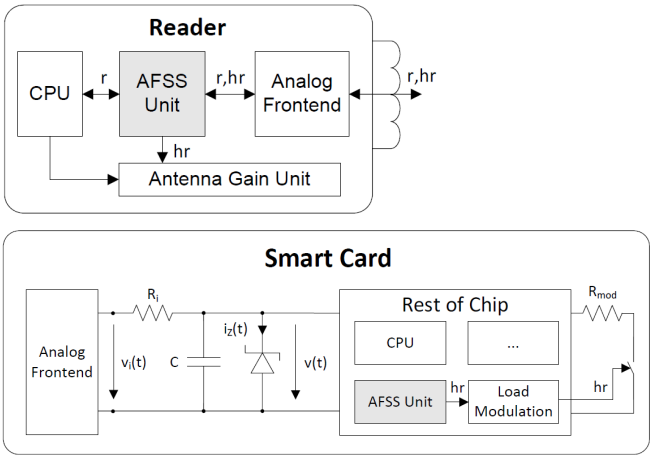


Figure 2.15: AFSS based on a hardware extension [DMS+12]

The software-based AFSS approach that integrates the power model inside the reader realizes to save 22% of energy compared to the standard approach of a constant maximum field strength. By including the model inside the smart card 25% of energy can be saved. Using AFSS hardware components however even saves as much as 41.9 % of energy.

2.2.2 NFC Use Cases

NFC has realized many new possibilities, but even though the technology was already introduced over five years ago, developers still have not exploited its full potential. Therefore, every year scientific works are being released that cover new aspects of NFC and demonstrate further ways of putting it to good use. Especially the inclusion of NFC hardware into mobile devices has enabled interesting new uses cases. According to [LRL10], the by far most common application of NFC today is mobile payment. This includes any kind of money transaction done through the contact-less technology, like buying goods in a store or purchasing tickets for public transport. Besides that, other categories of NFC use cases exist as well. Access control system for example, nowadays usually based on RFID technology, benefit from NFC's limitations, given that its short transmission range only makes the whole system safer. NFC can also be used to transfer small files or user specific information between two compatible devices like smart phones without the need to hook up any cables. The Android[1] operating system has already included such functionality, named *Android Beam*[2] inside its more recent versions. Nevertheless, payment applications are by far the most common use cases of NFC and will therefore be the main focus in the subsequent chapters of this document.

[1] Android is an open source operating system mostly used for smart phones and embedded systems. It is based on Linux and developed by the Open Handset Alliance http://www.android.com/ [2012-09-19]

[2] http://developer.android.com/guide/topics/connectivity/nfc/nfc.html [2012-09-19]

NFC Mobile Payment Use Cases

Smart phones have become ubiquitous utensils over the last few years and therefore represent the perfect devices for everyday business activities, given that they almost always are carried around by users. As described in [MLKS08], NFC enabled smart phones already include so called *secure elements* that can be used to store sensitive data, particularly suitable for payment use cases. Carefully developed systems that withstand attacks are very important in applications that deal with someone's possession like money. As discussed in [PaGAP10], a huge issue with payment systems is the fact that they on the one hand need to guarantee the user's privacy and prevent spying of data, but on the other hand also require the users to remain traceable to track down possible misuses. Therefore, a complete anonymity can never be allowed.

The idea of electronic money however is anything but new, even more than 15 years ago several systems were tested that relied on the mutual authentication between a merchant and a customer to exchange money in a secure way. As demonstrated in [Fin10], NFC is able to greatly simplify the process of a payment inside a store. Clients who do not wish to pay in cash usually have to provide a credit or debit card and either input a PIN or sign a bill. When using mobile payment it is sufficient to draw the smart phone close to a NFC terminal at the cash desk and the appropriate card inside the secure element will be charged in a contact-less way. Figure 2.16 shows the international symbol that denotes terminals allowing contact-less payment.

Figure 2.16: The international symbol to denote possible contact-less payments [LRL10]

2.2.3 Remote Control Interface for Android Devices

The author of this thesis himself has already worked with mobile NFC devices before. His document [Kip12] on the one hand describes a newly invented NFC health care use case, and on the other hand also demonstrates a web service-based interface. It was implemented to remote control Android applications, even world-wide over the Internet, to fully automate measurements of devices like smart phones or card readers. The preliminary work done for that project is still very relevant for this thesis, given that measurements results to be shown here in this document had to be acquired with the help of many different independent devices that all had to be controlled as well as linked together. The remote control interface enables to start measurement devices as well as applications under test at the same time and it controls their behavior in order to measure only the required time scope and create the exact scenario wanted at that moment.

This is achieved through the technology of web services, small programs being executed on a web server. One component that takes the control issues requests to the web service sever which stores them. Another component that is being controlled pulls these requests and executes them accordingly. This indirect route over the web service provides two great advantages. On the one hand, devices can be controlled from all over the world, an Internet connection is sufficient. Therefore, expensive and sensible measurement devices

can remain in one room, while the operator, wanting to acquire some data, can issue requests to the devices from anywhere else. All results are subsequently provided to him after the run trough was completed. On the other hand, using a general web service allows different platforms, usually not compatible at all, to work together. This is also relevant to this thesis, as it incorporates diverse operating systems on multiple devices.

Figure 2.17 shows the measurement setup that was used in a previous project to acquire the power consumption of smart phones and NFC card readers. As will be shown in following chapters, the measurements done for this thesis have employed a similar, but yet extended setup, that will be discussed in detail.

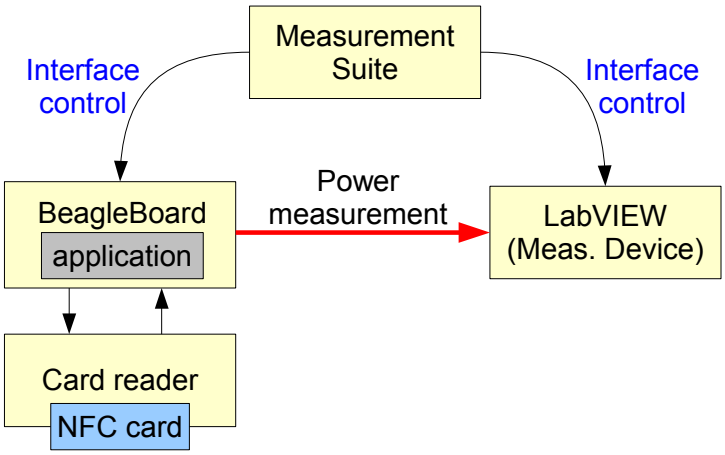


Figure 2.17: Measurement setup with the remote control interface [Kip12]

The remote control interface was developed for the superior META[:SEC:] project and provided to it after completion. It is currently in use in various different projects and has since proved itself to be a convenient tool to automate measurements and handle complex setups containing multiple independent devices in an easy way.

2.3 Design Prerequisites

This thesis develops much new functionality and provides a greater insight into branches of NFC that have not been inspected at all or were only discussed superficially. Nevertheless, several components necessary to reach the objectives already existed from preliminary works and could be used during the course of this thesis. This chapter aims at presenting the hardware and software parts upon which new developments are based on, including

- a National Instruments measurement device PXI-1042Q with additional integrated controller cards PXI-8187 and PXI-6221, and a shielded connector block SCB-68. A custom made USB connector, developed by META[:SEC:], was attached to the connector block and used to acquire current and voltage consumptions of the NFC reader. The remote control interface started and stopped measurements at exact times, therefore only samples actually required were measured,

- a LabVIEW[3] application, also called *virtual instrument*, running in LabVIEW 2011, version 11.0. It was provided by the superior META[:SEC:] project as well, and was used to acquire the samples from the measurement device,
- an AM37X evaluation board made by Texas Instruments and also provided by META[:SEC:] used to execute a proof of concept application. It represents a generalized mobile device, and dependent on the actual test case, can emulate commercial devices like smart phones or tablets alike. The board was also controlled via the remote control interface, which allowed to start the proof of concept application in time with the measurement device,
- a NFC reader used to communicate with the several tags inside its near-field, which was purchased by the author himself. One of the main goals of this thesis was to minimize the amount of energy consumed by this very reader,
- several NFC tags Infineon kindly offered that could be used for all further testing and development for free,
- the software remote control interface already mentioned, designed by the author himself in a previous project. The web service-based technology enabled different devices to be controlled via the Internet to achieve a concurrent execution. This was especially important to automate measurements of many independent components that needed to be individually controlled and managed. The interface allowed to alter the behavior of the evaluation board AM37X, the measurement device, and the NFC reader alike,
- a MATLAB[4] based graphical user interface, called *measurement suite*, developed by the META[:SEC:] project. It offers multiple features to receive the measurement data and forward it to external evaluation scripts or different plotting tools. It therefore represents the actual front-end of the whole measurement setup.

The Evaluation Board

This board was chosen mainly for two reasons. On the one hand it provides multiple useful ports to connect external hardware, on the other hand it includes a touchscreen directly mounted on the surface which emulates several commercial hand-held devices like smart phones or tablets and therefore further enables to simulate any new developments in an environment which is actually close to the target platform. Figure 2.18 shows the evaluation board AM37X used in this thesis.

The NFC Reader

Prior to any development, different types of NFC readers have been considered, extensive research and comparison of provided features as well as costs have led to one specific product being purchased by the author. Shown in Figure 2.19, this ACR122U reader was

[3] http://www.ni.com/labview [2012-09-19]

[4] MATLAB is a software package, developed by Mathworks, which is specialized on matrices manipulations and plotting. http://www.mathworks.com [2012-09-19]

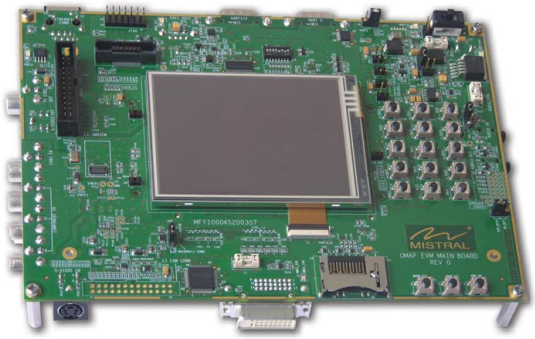


Figure 2.18: The TI evaluation board AM37X [Ins10]

chosen especially for its low-budget characteristics. On the downside however, the support by its manufacturer as well as the features actually provided are both very limited. Even the most basic functionalities relevant to this thesis had to be build up from scratch. Additionally, given that the interface of the reader did not satisfy the common standards of how such smart card devices are usually controlled, the design for this thesis had to be altered and adapted early on in the development process.

Figure 2.19: The NFC reader used for this thesis

The Measurement Suite

A MATLAB based graphical user interface called *measurement suite* was provided by META[:SEC:] to allow convenient measurements of various test cases and control multiple devices being part of the whole setup at once. Thanks to the remote control interface and its underlying web service technology, this component is entirely independent of the actual measurement device as well as any other hardware or software being involved. Requests for measurements can be issued from any place wanted, given that the suite connects to the Internet and transmits the commands over a web service to the measurement device as well as a proof of concept application being executed on the evaluation board. Therefore, fully automated measurements of multiple test cases can be realized, and results are being sent back to the measurement suite after the execution has finished. The test cases are

defined through simple text files containing all commands to be issued to the involved components in sequence.

The Big Picture

Chapter 3 that describes the design of this thesis as well as Chapter 4 which covers the actual implementation, will further explain how these components are actually linked together and a cooperation that leads to meaningful results is realized. The proof of concept application developed as well as the test cases used to retrieve measurement results will be discussed in detail. Additionally, extended information on the characteristics of the NFC reader hardware will be provided.

Chapter 3

Design

Given the overall complexity of this thesis, a carefully executed step-by-step design process was necessary before implementations of any components could be started. Figure 3.1 demonstrates an overview of the four major steps in the design process. Initially, all the possibilities of directly communicating with a NFC reader from inside an application had to be analyzed, given that any further functionality is inevitably based upon controlling the reader's behavior. Subsequently, research has shown that different smart card interfaces already exist and encapsulate the required protocol overhead in order to send commands to a reader or a tag. In depth analyses resulted in one interface being chosen for the further work to be based on. During the next step the target operating system, in this case Android was analyzed. Unfortunately, as opposed to many other systems, no direct integration of the design developed in the previous steps was possible, caused by several issues with the Android system structure. As a result multiple iterations of adapting to the target platform's limitations were carried out. Finally, after an approach was developed that bypassed all of Android's inherent issues, the design of the field strength scaling algorithms could be started.

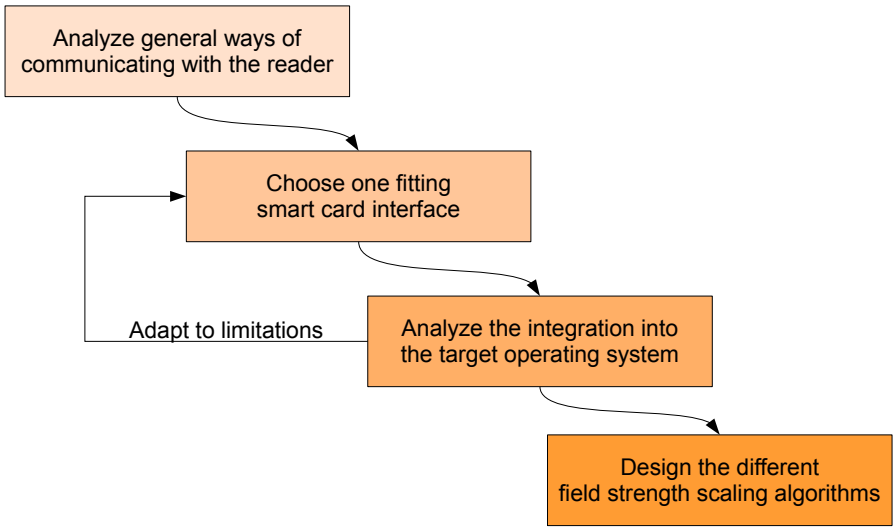


Figure 3.1: The design flow of this thesis

This section demonstrates quite well, how important a careful executed design process actually is. The algorithms developed in the last step are partially dependent on the operating system's structure and especially on all characteristics of the NFC reader. Any design flaws in earlier steps would definitely cause huge problems in latter ones or even require a total reworking of any design developed so far. The following sections explain in detail how the four steps discussed above were actually carried out.

3.1 Communication with the NFC Reader

One of the first major steps for this thesis was to research possible ways of communicating with a NFC reader directly from any newly developed software. Naturally, any field strength scaling algorithm presented in this document requires low level access to a NFC reader to achieve its goals. Commands have to be sent in order to control the behavior of a reader as well as the tags inside its field. Research has shown that usually two different ways of communicating with a reader are possible. Either directly by sending protocol compliant byte blocks and decoding the ones it returns or through a convenient programming interface, an additional abstraction layer introduced above the protocol itself. For the reader used in this thesis however no such a programming interface was available, dealing with low level commands therefore was the only option. Given that only a few bytes in the correct order are sufficient to alter a reader's behavior, a possible route would be to directly send them through any available bus the reader is connected to. The underlying protocol of the reader's hardware as well as the bus' characteristics would have to be dealt with however. To simplify matters, different smart card interfaces were already developed to encapsulate the complex protocol requirements.

3.2 Smart Card Interfaces

Depending on the operating system and the desired programming language, different software packages exist that deal with smart card hardware like NFC readers. They encapsulate the required steps for successful communications as well as the specific protocol characteristics and provide simplified interfaces. As a result, developers only have to care about the commands they want to issue to the hardware, whereas the actual process of how they are transmitted or received is mostly managed by the used smart card interface. During the research for this thesis, several such interfaces have been inspected,

- Java Smart Card I/O[1] is an extensive interface that provides access to various smart card components via low level commands. Any functionality is developed in the Java language which, thanks to its platform independence, leads to software that can be run in many different operating systems like Windows, Linux or Mac OS,
- WinSCard API[2] was specifically developed for Windows systems and requires software to be developed in C,

[1] http://docs.oracle.com/javase/6/docs/jre/api/security/smartcardio/spec/ [2012-09-19]
[2] http://msdn.microsoft.com/en-us/library/ms924513.aspx [2012-09-19]

- pcsc-lite API[3] is based on the functionality of WinSCard but has been adapted to support UNIX-related systems like Linux.

Additionally, several projects exits that especially aim at the subset of the smart card technology that is NFC, rather than providing an all-round package for any smart card related scenario. The by far most sophisticated project is an open source library called *libNFC*[4] that provides access to several NFC transmission modules, especially the one integrated in the reader used for this thesis. libNFC requires software to be developed in C, the package itself is available for Windows, Linux, and Mac OS.

The general way of communicating with NFC readers through any kind of smart card interface can be seen in Figure 3.2. A certain application running on a device issues several commands by passing them to the smart card interface, which pre-processes them. The interface establishes a low level communication with one of the device's hardware buses, in most cases USB is utilized. The original commands are wrapped up according to the protocol and sent through the bus to the actual NFC reader which executes them. Any results are sent back through the bus, decoded by the interface, and finally passed to the application. Dependent on the outcome of such a communication, the application may issues other commands and as a result actively control the NFC reader and its environment.

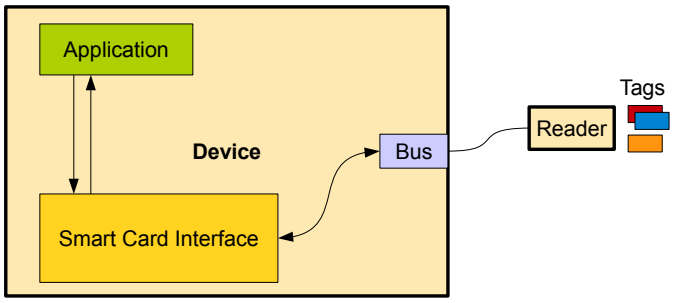


Figure 3.2: The general way of communicating with NFC readers

In order to send commands to the tags instead, the whole process is augmented by one additional step. The commands accepted by the reader are forwarded to the tags, their responses are decoded and finally sent back to the interface where the application can pick them up. From the application's point of view nothing has changed, independently of whether commands aim directly at the reader or are forwarded to the tags, the interface always provides the results to the application.

For this thesis libNFC was initially chosen to be the smart card interface used for further developments. The package, given that it especially aims at only the NFC technology alone, features a much simpler structure than any other interface providing access to all general smart card components. Unfortunately, as will be described in the following section, the target platform Android introduced several unsolvable issues and finally the Java Smart Card I/O had to be used instead.

[3] http://pcsclite.alioth.debian.org/api/group__API.html [2012-09-19]
[4] http://www.libnfc.org/documentation/introduction [2012-09-19]

3.3 Enhancing the Target Platform

Windows as well as Linux systems have been used for development, and both platforms natively support several of the smart card interfaces discussed above. For this thesis however the Android operating system was the intended target platform, chosen for several good reasons,

- it is optimized for mobile devices and adapts to their special characteristics like low data storage capacities, slower CPUs than on desktop computers, alternate input devices like touch-screens and so on. The energy saving approaches of this thesis especially aim at that kind of portable hardware, given that mobile devices have to solely rely on batteries with their very limited life time and any optimizations are badly needed nowadays,
- it is widely spread on many different popular devices like smart phones, tablets or for that matter even desktop computers. Developing multi-tag functionality for the Android system enables new use cases on many different devices at once,
- it offers easy high-level software development for everybody. The complex low-level functionality implemented during this thesis is provided through an interface, programmers without any prior knowledge can benefit from all the new possibilities,
- it was already the target platform for the remote control interface developed by the author prior to this thesis [Kip12]. Re-usability of course predicts that new developments should be based on earlier achievements, which was shown to work especially well in this case.

Unfortunately Android does not natively support any of the existing smart card interfaces at all. As a first approach, the operating system was extended by altering its source code and compiling it from scratch. An extensively enhanced libNFC, including the new field strength scaling functionality, was integrated as a low level library into Android. This approach however led to several issues with Android Linux drivers required by the reader, yet unavailable from its manufacturer. Subsequently libNFC was dropped altogether. In the meantime, an Android based library allowing communications with the reader was issued by the manufacturer. It provides high level access without the need of recompiling Android, unfortunately however leads to severe issues with multiple USB ports. No stable communication could ever be achieved. As a result, a totally different approach was developed. Any functionality required for a successful communication with the reader was integrated into the Java based web service of the remote control interface. It is executed in Windows and therefore bypasses all the issues with Android. A simplified abstraction layer above the complex Java Smart Card I/O was designed that finally forwards commands to the reader. The Android system at least provides access to web services, which allows applications executed on this platform to transmit commands to the Windows-based web service, which then forwards them to the actual NFC hardware.

The Remote Control Interface

The remote control interface was originally intended for use during measurements, in order to control multiple independent software as well as hardware components that all work

together and have to be measured simultaneously. In this thesis however, the interface serves another additional purpose, it forwards commands sent by Android applications to NFC hardware accessed with Windows. This section describes the over-all design of the remote control interface in order to explain how it contributes to the functionality developed in this thesis.

Each platform that uses the interface features an own specific client that is used to access the web service being executed on a server. If for example a Windows application requires some results from another program being run in Android, it sends the associated request to the web service. The Android application may ask the web service, whether new requests are available and subsequently executes the jobs demanded. Finally, any results are uploaded to the web service, where the Windows application can pick them up again. Therefore, the web service can be seen as a sort of temporary clipboard, storing requests, results, and state information until the respective clients pick them up again. Figure 3.3 demonstrates the design principle of the interface.

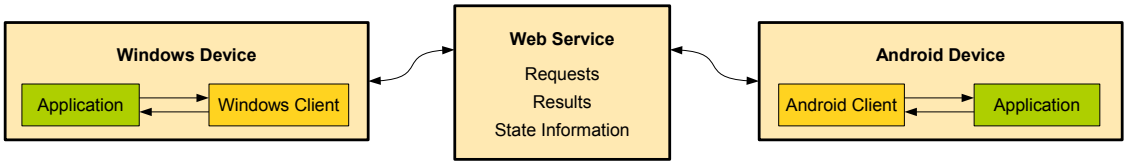


Figure 3.3: The design principle of the remote control interface

For this thesis however the web service component was extended to include the Java Smart Card I/O as well as a simplified abstraction layer above it. It encapsulates the quite complex functionality of the underlying interface and only provides one single method, used to send commands and to receive respective answers. Similar to the way results are returned during measurement, whenever an Android application needs to transmit commands to the NFC reader it calls the web service and passes them. The great difference however is the web service's behavior in that case. As shown in Figure 3.4, instead of storing the commands for the Windows client to pick them up, it uses the simplified smart card interface to instantly forward the commands to the reader and to return the results back to the calling Android client. This of course is only possible due to the fact that the NFC reader is directly attached to the Windows PC hosting the web service. It of course would also be possible to connect the reader to any distant computer with its own client, for this thesis however the more direct approach was chosen instead.

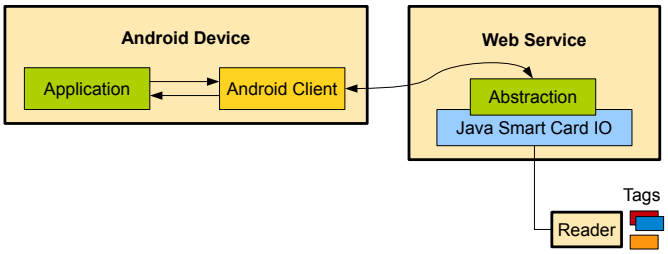


Figure 3.4: Accessing the NFC reader over the web service

As a result of the design decisions discussed above, no support for NFC at all is required on the Android device, the ability to send commands to the web service is sufficient in this case to actually control the reader and communicate with the tags. The applications using the reader and everything associated to them however can very well be implemented to run in Android. Any logic that evaluates results is part of the Android application and commands are built up inside the mobile device, ready to be issued to a reader. As soon as Android's issues with Linux drivers as well as USB ports are solved in the future, another smart card interface like libNFC can be integrated and the exact same commands used during this thesis can be sent to a reader directly connected without the indirection of the web service.

3.4 Design of the Field Strength Scaling Algorithms

Based upon the considerations of the previous sections the actual design of all algorithms used in this thesis is presented. NFC readers usually supply tags with the maximum amount of field strength to ensure a stable communication at all times [DMS+12]. While this seems to be a reasonable idea, it leads to a severe waste of energy. Research during this thesis has shown that only a small amount of the maximum field strength is sufficient for a successful detection and communication, even if multiple tags are present at the same time. From a power-aware point of view, one can clearly understand that less field strength results in less energy consumption, which is especially important for mobile readers that rely on batteries and their limited operational time.

These facts are valid for single tag as well as multi-tag applications, whereas the latter are the main interest of this thesis. Two different algorithms that scale the reader's field strength to find the least amount of field sufficient to communicate with one specified tag T inside the near-field will be presented and compared to the standard approach of always using the maximum available field strength. It will be shown that multiple tags lead to additional effects, not present in single tag scenarios, which are quite surprising. Readers using the maximum field strength are not able work with a larger number of tags in the vicinity because the mutual interference between them, as already discussed in this document, renders any communication or even detection impossible. By scaling the field strength gradually, only a few of the tags can be reached in every step, which decreases the amount of detune that occurs. This leads to a rather paradoxical case where more tags can be found with less field strength. Additionally, many tags not found with the maximum field strength at all can easily be reached with a fraction of it. The following sections will discuss this discovery in more detail. For this thesis two new algorithms to handle multiple tags in the vicinity are developed, which were named

- *FSS*, gradually scaling the field strength,
- *BIN*, executing a binary quadratic search through the field strengths.

Both algorithms are compared to the one existing standard approach to be called *CONST* that always operates with the maximum field strength, regardless of the actual situation at hand. The different algorithms will be explained in short, and subsequently compared to highlight their performances in the best, average, and worst cases. Approximation equations are developed that describe the behavior of all algorithms in relation to each

other, which allows quick comparisons of their energy consumptions in a very general way. Chapter 5 will then discuss the measurement results of all algorithms and compare them to the approximated predictions.

3.4.1 Definitions

This thesis deals with the scenario of multiple NFC-tags in the vicinity, all powered by one single field generated by the reader. As already mentioned, in order to acquire the full UID of a tag and establish a communication with it, a Single Device Detection (SDD) has to be executed. The author himself implemented a multi-tag functionality that is able to list every reachable tag, one after the other. All algorithms developed for this thesis are based upon this functionality.

The total amount of tags that could possibly be reached because they are inside the range of NFC is called n. The fraction of tags that can be seen with the current field strength is called $n_{seen} \leq n$. Most NFC readers use NXP's transmission module PN532. As described in the technical documentation [NXP07c], the module and therefore the reader being based on it can only ever communicate with one tag T at a time. As a consequence, at the beginning of every communication, the algorithm has to search trough the n_{seen} tags it can reach at the moment and look for the wanted tag T. All algorithms described in this thesis discover the tags consecutively, although the actual order of those detections can not be predicted and may change over time.

The energy consumption of the algorithms described in this document can be computed as shown in Equation (3.1).

$$E = \sum_i (P_i \cdot t_i) \tag{3.1}$$

whereas P_i describes the actual power consumed at that specific moment, depending on the currently set field strength. The time t_i consists of three different parts,

- the time t_{set} it takes to set the field strength once,
- the duration t_{find} necessary to find the wanted tag. This requires one or several Single Device Detections (SDD) to be executed, each taking a time of t_{SDD},
- and the time t_{read} required to read from the wanted tag, if it could be found in the previous steps.

The following sections describe those three time ranges in more detail, given that they largely affect the algorithm's execution time and therefore the actual consumed energy. In order to remain as general as possible, the time ranges are not expressed in any usual unit like seconds but described in relation to each other. As a result, an abstract time factor *tu* is introduced. The actual fraction of a second *tu* represents is not of any interest, given the factor is only used to compare the different algorithms in relation to each other, rather than actually compute a numerical solution. Figure 3.5 shows one possible measurement result of the FSS algorithm. In this specific example, FSS had to set the field strength once, shown here in red, and list one of the present tags, shown here in green. Close inspections of huge amounts of measurement data have proved that an simple correlation between those two time ranges exists, which is even independent of the algorithm actually used. The time period required to list one tag always takes approximately 1.2 of the

time that passes while the field strength is set once. Therefore, by definition, one field strength setting operation is said to be done in $1 \cdot tu$, whereas listing one tag takes $1.2 \cdot tu$. Additionally, after a tag was successfully found, usually reading operations are carried out. The algorithms demonstrated in this document for example read 1 KB each. The power profile shown in Figure 3.5 demonstrates 64 such cycles, each acquiring 16 bytes of data, resulting in the 1 KB being read. If compared to the other time ranges, it can be shown that the reading operations, although varying a bit over time, require an average amount of $1.25 \cdot tu$ each. Therefore $80 \cdot tu$ are necessary to acquire 1 KB of data from a tag. With these three values at hand, the energy consumptions of the algorithms presented in this document can actually be compared quite easily.

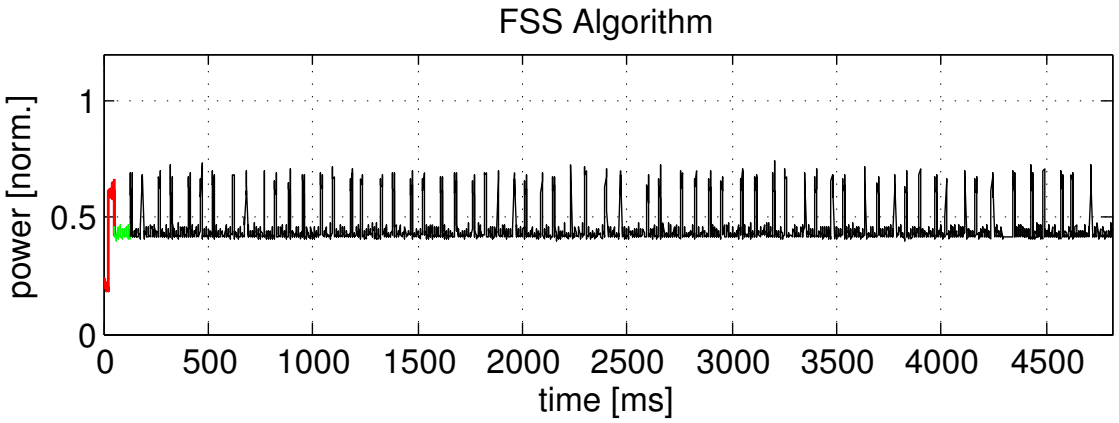


Figure 3.5: Acquiring the the abstract time value tu

t_{set}

Every algorithm discussed in this thesis has to set the field strength used during transmission at least once. CONST for example always transmits with the maximum available field strength and configures the reader to use that setting right at the start. The other two algorithms of this thesis however are based on the principle of field strength scaling and therefore may even set the field multiple times during their execution. To cover the resulting delay that influences the energy consumption, t_{set} is introduced which describes the amount of time required to set the field strength once. By definition one single field strength setting operation takes tu time.

t_{find}

In order to communicate with a certain tag, independently of how many targets are actually inside the near-field at that time, the transmission module of the reader has to acquire its full UID. This process can be repeated until either all tags have been inspected or the wanted one was successfully found. If only one tag is reached with the current field strength, its UID is instantly returned. Whenever collisions occur at that moment because more than one target answers, a SDD has to be executed that represents a kind of binary search. It requires several steps to return one full UID.

Measurements have revealed however, that the communication between an algorithm and the reader hardware consumes a lot more time than several SDDs inside the reader. Therefore, the delay to inspect one tag's UID is mostly determined by the transmission delay between an algorithm and the reader. It can be taken as approximately independent of the current amount of tags present in the near-field and the number of collisions that may occur during tag discovery. Even at times when no tag is found at all, the amount of delay is still very similar. All subsequent discussions therefore use the term *listing operation* rather than *SDD*, because situations where only one tag or none is reachable do not require an actual SDD to be executed. The delay of said *listing operation* however remains the same in all cases.

As a result, t_{find} describes the total time required to actually reach the wanted tag. One listing operation requires an average of $1.2 \cdot tu$, and if $n_{tags-listed}$ describes the number of tags that have to be tested by the NFC transmission module inside the reader until the wanted one is found, t_{find} can be expressed as $t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu$.

t_{read}

In order to demonstrate the correct behavior of all algorithms discussed in this document, an amount of 1 KB has to be read from the specified tag after it was successfully reached. This process takes approximately 80 time units and is independent of the algorithm used. All approaches rely on the exact same commands and the transmission rate is never changed. Only the used field strength and therefore the momentary power consumption of each read cycle is set by the algorithm during tag discovery. The value of t_{read} therefore is constant and can be written as $t_{read} = 80 \cdot tu$.

3.4.2 CONST Algorithm - Constant Maximum Field Strength

This section inspects the standard approach used in current NFC systems. CONST sets the field strength once right at the beginning and subsequently operates with the maximum available field during all of the following steps. The required power P can be written as $P = P_{max} = const.$ It would be wrong to assume that CONST, using the maximum field strength, can detect all n tags that are inside the range of the field. The mutual interference between the tags prevents this in many cases, the more tags present and the closer they are put together, the fewer tags can be successfully found with CONST. Nevertheless, to be able to compare this algorithm with the following ones, for this analysis a scenario is assumed, where a few tags, but still much more than just a single one, are present and kept in such locations that all n of them can be found. This is not unrealistic or contrived at all and enables to analyze CONST's behavior, especially compared to all other algorithms.

Best Case

In the best case the wanted tag is the first one to be listed by the transmission module, $n_{tags-listed} = 1$, which leads to a time range of

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot tu$$

CONST of course only sets the field strength once

$$t_{set} = 1 \cdot tu$$

and finally 1KB is read from the tag, which as discussed before always takes

$$t_{read} = 80 \cdot tu$$

The total time can be written as

$$t = t_{set} + t_{find} + t_{read} = 82.2 \cdot tu$$

The field strength used by CONST always introduces a power consumption of $P = P_{max}$. The energy consumption of the best case is therefore seen in Equation (3.2).

$$E = P_{max} \cdot 82.2 \cdot tu \tag{3.2}$$

Worst Case

The wanted tag is only found after all the others have been inspected, which results in

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot n \cdot tu$$

with n being the number of tags present in the near-field at the time

$$t_{set} = 1 \cdot tu$$

because the field strength is set only once right at the beginning and as usual

$$t_{read} = 80 \cdot tu$$

Therefore, the entire time range required is

$$t = t_{set} + t_{find} + t_{read} = (1.2 \cdot n + 81) \cdot tu$$

and with the consumed power of

$$P = P_{max}$$

the energy consumption of the worst case can be written as shown in Equation (3.3).

$$E = P_{max} \cdot (1.2 \cdot n + 81) \cdot tu \tag{3.3}$$

Average Case

For the average case it is assumed that after listing half of the tags, the wanted one is found. As a consequence, $\frac{n}{2}$ tags have to be inspected

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot \frac{n}{2} \cdot tu = 0.6 \cdot n \cdot tu$$

the number of field setting operations and the time required to read from the tag of course remain the same

$$t_{set} = 1 \cdot tu$$

$$t_{read} = 80 \cdot tu$$

and the entire time range can be written as

$$t = t_{set} + t_{find} + t_{read} = (0.6 \cdot n + 81) \cdot tu$$

Given that the consumed power is

$$P = P_{max}$$

the energy consumption of the average case can computed as shown in Equation (3.4).

$$E = P_{max} \cdot (0.6 \cdot n + 81) \cdot tu \tag{3.4}$$

Table 3.1 summarizes the different cases of the CONST algorithm.

	Energy [J]
Best Case	$P_{max} \cdot 82.2 \cdot tu$
Average Case	$P_{max} \cdot (0.6 \cdot n + 81) \cdot tu$
Worst Case	$P_{max} \cdot (1.2 \cdot n + 81) \cdot tu$

Table 3.1: Different cases of the CONST algorithm

3.4.3 FSS - Gradual Field Strength Scaling

Working with multiple tags and a constant maximum field strength introduces two issues,

- a lot of energy is wasted in cases where a small amount of the field strength would be sufficient to reach a certain tag,
- the more tags that can possibly be reached, which would be especially many with the maximum field, the bigger the resulting undesired mutual interference would become. In the worst case, the maximum field being able to reach all tags leads to such a heavy detune that not a single communication with any of the tags is possible anymore.

To prevent said issues and to optimize the energy consumptions, FSS was developed during the research for this thesis. The field strength is dynamically scaled to 12 different values, from now on called *shells*, that consume a power of $P = P_i$. In the last shell, number 12, the power consumption is equivalent to the one of the CONST algorithm, which always uses the maximum field strength. FSS simply scales the field strength gradually until the wanted tag can be found or the maximum amount was already set and non could be detected. Unwanted tags that were reported in earlier steps have to be remembered and need to be excluded in later steps of the search. This is especially important for FSS and the other optimized algorithm as well in order to work correctly. Section 3.4.4 will further explain why neglecting to store already found tags leads to sever problems.

One big advantage of FSS lies in the fact that each of the 12 steps only reports a single or a few new tags and the algorithm just needs to handle a small part of all the n tags that could possibly have been reached with the maximum field. This solves the first problem described above, as only a small amount of the tags is active at one moment in time and the mutual interference is kept low. While CONST fails to communicate with tags in many scenarios where a lot of them are close to each other, FSS through its gradually field strength scaling is able to successfully communicate with all of them. The main goal of this thesis however is to reduce the required energy, and working with lower field strengths of course results in less overall energy consumption.

A disadvantage of the FSS algorithm however is the fact that it may have to step through multiple field strengths, in the worst case all 12 of them, which leads to an increased overhead, depending on the number of times the algorithm has to switch to another field strength.

The actual power consumptions P_i of each shell had to be acquired by repeated measurements, because unfortunately no such information is provided in any data sheet of the used transmission module. Following the tag discovery process, during which several field strengths may be used in a quick succession, the algorithms settle on a certain field strength that is subsequently used while reading from the tag. By inspecting multiple use cases that result in different shells being used during reading and averaging over multiple similar measurements to filter noise or deviations, it is possible to compute the mean power consumption of all shells step by step as shown in Table 3.2.

Best Case

In the best case of FSS, the wanted tag can be found in shell 1 with $P = P_1$, the required power is therefore equivalent to the one of the minimum field strength. According to the values shown in Table 3.2

$$P = P_1 = 0.633 \cdot P_{max}$$

Additionally, in the best case the first tag listed by the transmission module is already the wanted one, $n_{tags-listed} = 1$

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot tu$$

Furthermore, since the lowest field strength is sufficient, the field has to be set only once

$$t_{set} = 1 \cdot tu$$

and finally 1KB is read from the tag, which always takes

$$t_{read} = 80 \cdot tu$$

The total required time can therefore be written as

$$t = t_{set} + t_{find} + t_{read} = 82.2 \cdot tu$$

which is equivalent to the best case of CONST. FSS however uses the lowest field strength possible, which consumes a power of P_1, to reach the wanted tag and subsequently read from it. The resulting energy consumption can be written as

$$E = P_1 \cdot 82.2 \cdot tu = 0.663 \cdot P_{max} \cdot 82.2 \cdot tu$$

	Power [W]	**Relation [%]**
P_1	0.3315	66.30
P_2	0.4115	82.30
P_3	0.4550	91.00
P_4	0.4700	94.00
P_5	0.4775	95.50
P_6	0.4825	96.50
P_7	0.4875	97.50
P_8	0.4900	98.00
P_9	0.4925	98.50
P_{10}	0.4950	99.00
P_{11}	0.4975	99.50
P_{max}	0.5000	100.00

Table 3.2: Power consumption of the different FSS shells

or simplified to the form seen in Equation (3.5).

$$E = P_{max} \cdot 54.50 \cdot tu \tag{3.5}$$

Worst Case

FSS has to scale up to the last shell and the wanted tag is the last one to be reported. Therefore, every tag in each shell has to be tested. It can be shown however that the worst case is the scenario where all n tags are inside the last shell, number 12, whereas any other constellation, with one or more tags in other shells leads to less energy consumption and therefore cannot be the worst case anymore.

As discussed before, it takes much longer to communicate with the reader and tell it to list a tag, than it takes the reader to actually execute the respective logic. Therefore, even in cases where no tag is found in a shell, the approximated $1.2 \cdot tu$ per each listing operation and $1 \cdot tu$ to set the associated field strength can be used. As a result, in every shell up to 11, FSS executes one field setting operation and one listing operation that cannot find any tag,

$$t_{set} = 1 \cdot tu$$

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot tu$$

which results in a energy consumption of

$$E = \sum_{i=1}^{11} \Big(P_i \cdot (t_{set} + t_{find}) \Big) = P_{max} \cdot 22.40 \cdot tu \tag{3.6}$$

In shell 12 with a power consumption of P_{max} a total of n actual tags have to be inspected to find the wanted one. This scenario of course is equivalent to the worst case of CONST, which was shown to consume an energy of

$$E = P_{max} \cdot (1.2 \cdot n + 81) \cdot tu \tag{3.7}$$

The entire energy required in the worst case of FSS can therefore be computed as a sum of the tag discovery processes in the empty shells 1 to 11, shown in Equation (3.6) as well as the actual tag discoveries in the last shell 12, that contains all n tags, shown in Equation (3.7). This leads to a total energy consumption as demonstrated in Equation (3.8).

$$E = P_{max} \cdot (1.2 \cdot n + 103.4) \cdot tu \tag{3.8}$$

The worst case constructed above however is not very realistic. The number of tags that could possibly be present inside the last shell, number 12, is of course limited by space and mutual interference and putting all n tags that close together will definitely not work in most scenarios. Therefore, to compare FSS and CONST in a situation that is closer to any real life application, the average case has to be analyzed.

Average Case

It is assumed that after half of the tags have been inspected the wanted one is found. Figure 3.6 shows the field strengths as a function of the current shell. It is obvious that out of the 12 shells available, shell 6 does provide way too much field strength to be the average shell. Shell 2 is by far the closest to the mean value, noted by the horizontal line in Figure 3.6. For an average case it can be assumed that the wanted tag can be found with that amount of field strength. By the definition above, after inspecting $\frac{n}{2}$ tags, the wanted one is reached and these $\frac{n}{2}$ tags are most probably to be found inside the first two shells, which represent the lower half of the field strengths. However, as demonstrated in Figure 3.6, shell 1 only offers quite a low field strength, there is a visible step from shell 1 to 2. The probability of finding the wanted tag in shell 1 with the lowest field strength setting is a lot smaller than in shell 2. Given that it provides close to half of the maximum field, the probability is quite high that the wanted tag is in fact found within shell 2. As a result, for an average case it can be assumed that FSS has to try two shells, with the power consumptions P_1 and P_2, while executing $\frac{n}{2}$ SDD operations to find the wanted tag. In both shells of course the field strength has to be set only once, leading to

$$t_{set} = 1 \cdot tu$$

in each of two shells an average of $n_{tags-listed} = \frac{n}{4}$ tags have to be inspected

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 0.3 \cdot n \cdot tu$$

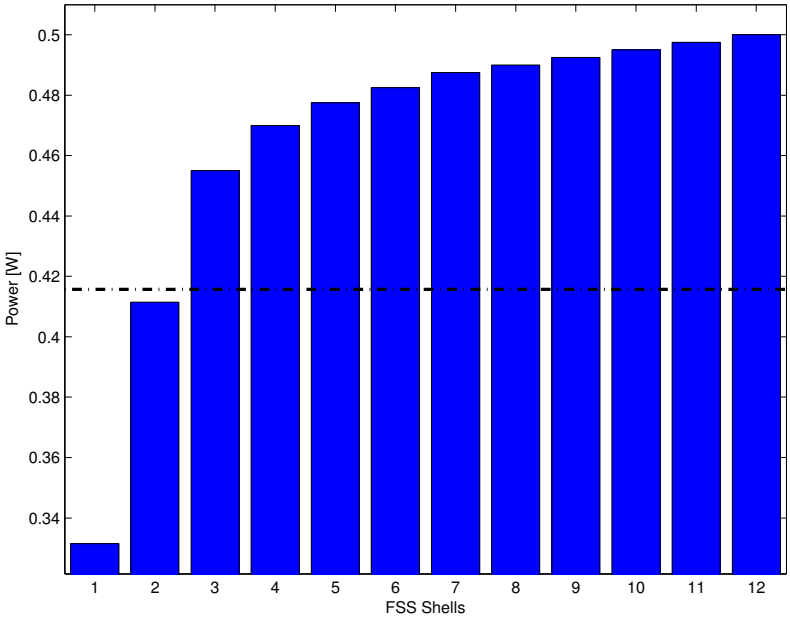


Figure 3.6: Resulting power consumption over the used shell

whereas in the second shell the tag is found and has to be read

$$t_{read} = 80 \cdot tu$$

This leads to an energy consumption of

$$E = P_1 \cdot (t_{set} + t_{find}) + P_2 \cdot (t_{set} + t_{find} + t_{read})$$

or when inserting the actual values results in the final form as seen in Equation (3.9)

$$E = P_{max} \cdot (0.45 \cdot n + 67.33) \cdot tu \tag{3.9}$$

Table 3.3 shows a summary of the different cases of the FSS algorithm.

	Energy [J]
Best Case	$P_{max} \cdot 54.50 \cdot tu$
Average Case	$P_{max} \cdot (0.45 \cdot n + 67.33) \cdot tu$
Worst Case	$P_{max} \cdot (1.2 \cdot n + 103.4) \cdot tu$

Table 3.3: Different cases of the FSS algorithm

3.4.4 The Naive FSS and its Issues

By remembering the tags already found, it is ensured that tags not matching the wanted one in a certain shell are not being tested again in another shell. This distinguishes the FSS from a naive approach towards field strength scaling that would always report any tag reachable and would, of course, lead to a huge amount of tags being tested again and again. The required energy for the worst case of the naive approach is at first sight equivalent to the worst case of FSS but there is a huge difference that needs to be discussed. As can be

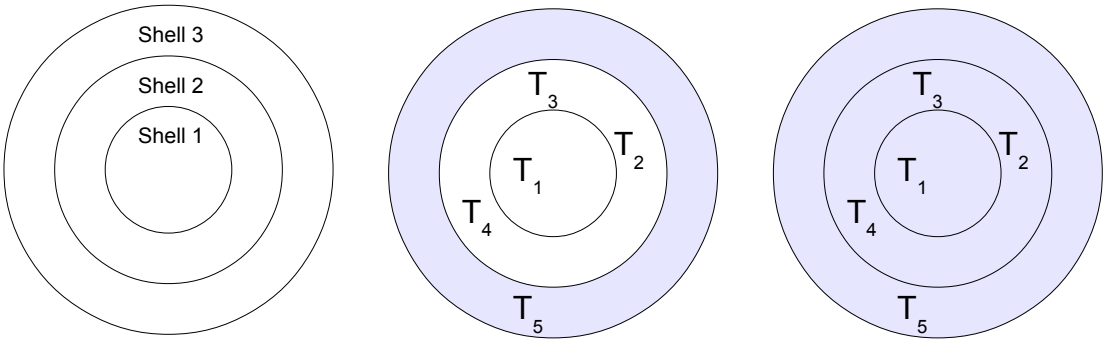


Figure 3.7: Different shells, FSS and the naive approach

seen in Figure 3.7 on the far left side, there are various different shells, in this example only three to keep it simple. The image in the middle shows how the FSS algorithm usually behaves. It detects tag T_1 in shell 1 and remembers it, in shell 2 tags T_2, T_3 and T_4 are found and noted, while T_1 is not being tested anymore. If it is now assumed, that FSS is currently searching shell 3, only the gray shaded area is being inspected and only one new tag, T_5, is being reported. FSS only ever discovers tags in the current shell and at the end of the execution has reported the five tags present inside the three shells.

The naive approach on the far right however acts totally different. In shell 1 tag T_1 is found, in shell 2 however, besides the newly discovered T_2, T_3 and T_4, tag T_1 is being reported yet again. Therefore, if it is assumed that the naive approach currently inspects shell 3, it detects all five tags once again, as shown by the shaded area. Therefore, the naive approach discovers all tags in shells smaller or equal than the current one and at the end of the execution has reported ten tags, twice as much as FSS. This behavior of course always leads to far more than n tags being inspected and results in a very bad performance energy wise. It can be shown that utilizing the naive approach consumes much more energy than CONST and is therefore useless in all scenarios.

3.4.5 BIN - Binary Quadratic Search through the Field Strengths

FSS strictly scales the field strength from the lowest to the highest shell, which in some cases may be suboptimal, if for example it has to step through all 12 of the possible shells. BIN was invented to handle these scenarios more efficiently and to find the required field strength with only as much as five steps through the shells or even less. This is achieved with a quadratic binary search through the available field strengths. Given that the field strengths do not grow linearly, which was already shown in Figure 3.6, a commonly used

linear binary search would be very inefficient. Starting with shell 6, which already provides 96.5% of the maximum field strength, would lead to huge amounts of mutual interference and yield in a situation very similar to the standard approach of CONST. As a result, a linear search would impair BIN's efficiency a lot.

Therefore, an optimized quadratically binary search was designed for use with BIN. The algorithm handles the 12 field strengths as a sorted list, starts with a field strength close to the mean one, for this matter shell 2 was chosen, tests whether the wanted tag can be found with this field strength or not, and recursively continues in the lower or upper half of the list to look for another field strength that can reach the tag.

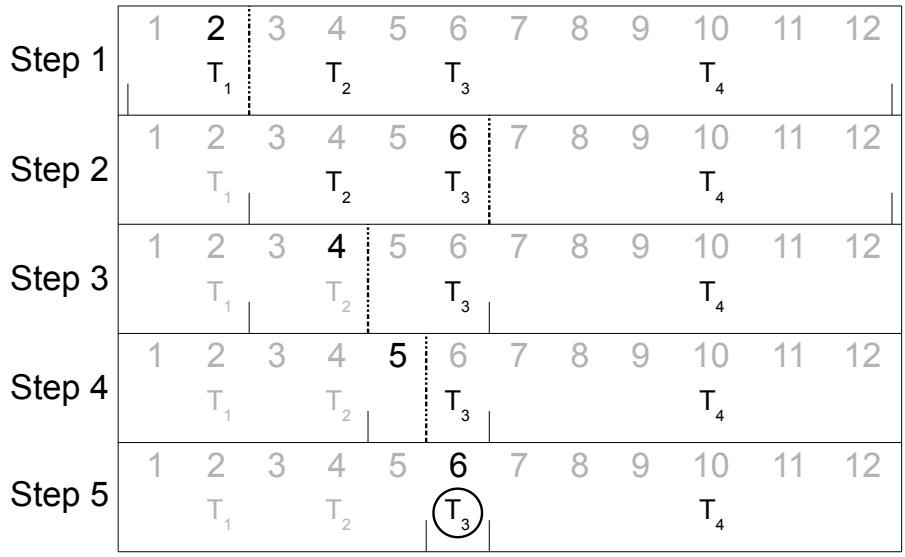


Figure 3.8: Quadratic binary search of the BIN algorithm

Figure 3.8 demonstrates this functionality in an example. The five steps BIN has to take are shown in five different rows. Each of them has numbers on the top, representing the 12 shells. A vertical dashed line demonstrates BIN's chosen splitting point that divides the field strengths in a lower and an upper part. The markers on each side on the bottom represent the current subset of the field strengths being analyzed. The used field strength is highlighted, whereas the others are kept in gray. On the bottom of each row all tags present in a certain shell are shown. Tag T_3 is the wanted tag in this example,

- during step 1, BIN sets a splitting point of 2 and is able to reach T_1. Given that it is not the wanted tag, T_1 is deactivated and excluded in future searches. The wanted tag could not be found in the lower part of the field strengths and BIN therefore continues with the upper part, shells 3 to 12,
- in step 2, BIN finds a new splitting point of 6 and starts another tag discovery process. T_1, which was already deactivated in step 1 is shown in gray here because it cannot be found anymore. T_2 and T_3 are discovered, T_2 is deactivated whereas T_3, the wanted tag, is kept active. Given that BIN was able to find the wanted tag in the lower part of the field strengths, shells 3 to 6, it continues its search in that area,

- in step 3, BIN sets a splitting point of 4. No tag is found however because none is active inside the inspected lower part of shells anymore. More importantly, the wanted tag T_3 cannot be reached and in the next step the upper part of the shells 2 to 6, which is 5 to 6, has to be inspected,
- in step 4, only two possible field strength shells still remain. BIN tests the lower one first, which is shell 5, but is not able to reach the wanted tag,
- In step 5, BIN tries shell 6 and successfully discovers the wanted tag. The search has narrowed down the possible field strength settings to one single value. BIN can be certain that the wanted tag, which was already found within shell 6 in step 2, is still present and can therefore start to read from it.

In the example discussed above, BIN requires five steps to find a tag in shell 6. FSS however would have needed six steps to scale from shell 1 to 6. Additionally, during measurements scenarios could be inspected where BIN's quadratic search is even more efficient. The actual order of tag discovery and the point in time when a certain tag that of course interferes with others is deactivated differs in each algorithm. FSS inspects shells in a strict sequence from 1 up to a certain number, whereas BIN may jump a lot between the different shells. Interestingly, situations exist for example where FSS requires nine steps and finally reads the tag in shell 9, whereas BIN can successfully narrow down a possible field strength in only four steps and subsequently read the tag in shell 2. The seemingly inefficient behavior of FSS however is no malfunction, but is rather caused by the actual order in which tags are discovered and the states of all other tags, being active or deactivated. This leads to totally different effects between the multiple tags present and especially to varying amounts of mutual interference.

Best Case

By design, BIN starts in shell 2 with $P = P_2$. In the best case, the wanted tag can be found in shell 1, while both shells 1 and 2 contain no other tags. BIN is able to find the wanted tag in shell 2. As a result just one additional test is necessary, given that the only other shell lower than shell 2 that could also possibly reach the wanted tag of course is shell 1. In the best case, this test also succeeds and the reading operation can be started in shell 1.

BIN therefore requires one field strength setting operation in each shell

$$t_{set} = 1 \cdot tu$$

and has to list only one tag every time

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot tu$$

When also considering that the tag is read in shell 1

$$t_{read} = 80 \cdot tu$$

this leads to an energy consumption of

$$E = P_2 \cdot (t_{set} + t_{find}) + P_1 \cdot (t_{set} + t_{find} + t_{read})$$

As a result, the best case of BIN can be written as shown in Equation (3.10).

$$E = P_{max} \cdot 56.31 \cdot tu \tag{3.10}$$

Worst Case

For the worst case it is assumed that all n tags are inside shell 12 and BIN can only find the wanted one after testing all others and scaling to the highest field strength. The algorithm starts with shell 2, cannot find the tag and therefore has to search for it in the upper part of the field strength shells. Subsequently, shells 6, 9, 11, and finally 12 are tried. As discussed before, given the large communication overhead, even shells without tags require a quite similar amount of time to be inspected. Therefore, BIN has to execute one field setting as well as one listing operation in the first four of its five steps. With

$$t_{set} = 1 \cdot tu$$

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot tu$$

the energy required for the first four steps can be written as

$$E_{1-4} = P_2 \cdot (t_{set} + t_{find}) + P_6 \cdot (t_{set} + t_{find}) + P_9 \cdot (t_{set} + t_{find}) + P_{11} \cdot (t_{set} + t_{find})$$

$$E_{1-4} = 2.2 \cdot (P_2 + P_6 + P_9 + P_{11}) \cdot tu = P_{max} \cdot 8.29 \cdot tu$$

In the last step however, BIN has to search through n different tags to finally find the wanted one and read from it. At first, the field strength has to be set the maximum amount

$$t_{set} = 1 \cdot tu$$

subsequently n tags are tested

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot n \cdot tu$$

and finally the wanted tag is found and the reading operation is started

$$t_{read} = 80 \cdot tu$$

This leads to an energy consumption equivalent to the worst cases of CONST as well as FSS, which was already shown to be

$$E_{worst} = P_{max} \cdot (1.2 \cdot n + 81) \cdot tu$$

The worst case of BIN can therefore be written as a sum of both energy consumption, E_{1-4} and E_{worst}, as shown in Equation (3.11)

$$E = P_{max} \cdot (1.2 \cdot n + 89.29) \cdot tu \tag{3.11}$$

Average Case

For the average case it is assumed that the wanted tag can be found in shell 2 which provides close to half of the possible field strength and that shells 1 and 2 taken together contain $\frac{n}{2}$ tags. Therefore, BIN as usual starts in shell 2 and has to test $\frac{n}{2}$ tags until it discovers the wanted one. Subsequently BIN tries to reach the tag with even less field strength in shell 1 but does not succeed. BIN has to use shell 2 again, even thought it already reached the wanted tag with that amount of field strength in step one. This behavior seems to be quite wasteful, but can be explained easily. In order to read from a tag, it has to be the one currently active and prepared for communications. Step two of the scenario discussed in this section does not find any tag, therefore no target is actually activated and prepared, in other scenarios an unwanted tag may be active at that moment. As a result, it is necessary to ensure that the tag active in the last step is always indeed the wanted one.

In the average case described here the field strength is set once

$$t_{set} = 1 \cdot tu$$

and by the definition above, $\frac{n}{2}$ tags have to be inspected until the wanted one is reached,

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 0.6 \cdot n \cdot tu$$

The energy required for this first step can be written as

$$E_1 = P_2 \cdot (0.6 \cdot n + 1) \cdot tu$$

All tags but the wanted one are deactivated and therefore excluded in future searches. In the second step, BIN tries shell 1 but cannot find any tags at all, because the wanted one is present inside shell 2 and all others are deactivated. Given that one field setting and one listing operation are necessary, the energy required in this second step can be computed as

$$E_2 = P_1 \cdot 2.2 \cdot tu$$

Finally, BIN returns to shell 2 with one single field setting operation, only discovers the wanted tag and starts to read it.

$$t_{set} = 1 \cdot tu$$

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot tu$$

$$t_{read} = 80 \cdot tu$$

$$E_3 = P_2 \cdot (1 \cdot tu + 1.2 \cdot tu + 80 \cdot tu) = P_2 \cdot 82.2 \cdot tu$$

As a result, the energy required in the average case of BIN can be written as a sum of all the energy equations shown above. Equation (3.12) demonstrates the final result.

$$E = P_{max} \cdot (0.49 \cdot n + 69.93) \cdot tu \tag{3.12}$$

Table 3.4 shows a summary of the different cases of the BIN algorithm.

	Energy [J]
Best Case	$P_{max} \cdot 56.31 \cdot tu$
Average Case	$P_{max} \cdot (0.49 \cdot n + 69.93) \cdot tu$
Worst Case	$P_{max} \cdot (1.2 \cdot n + 89.29) \cdot tu$

Table 3.4: Different cases of the BIN algorithm

3.4.6 Comparison of the Algorithms

Table 3.5 demonstrates that FSS performs very well energy wise in the best as well as the average case, being the most efficient of all the algorithms. In the worst case FSS is not the best but rather the worst choice of all, caused by the fact that 12 field strength scaling operations are necessary and a lot of time is wasted while searching for a tag that can only be found in the absolute last step.

	Best Case	**Average Case**	**Worst Case**
CONST	$P_{max} \cdot 82.2 \cdot tu$	$P_{max} \cdot (0.6 \cdot n + 81) \cdot tu$	$P_{max} \cdot (1.2 \cdot n + 81) \cdot tu$
FSS	$P_{max} \cdot 54.50 \cdot tu$	$P_{max} \cdot (0.45 \cdot n + 67.33) \cdot tu$	$P_{max} \cdot (1.2 \cdot n + 103.4) \cdot tu$
BIN	$P_{max} \cdot 56.31 \cdot tu$	$P_{max} \cdot (0.49 \cdot n + 69.93) \cdot tu$	$P_{max} \cdot (1.2 \cdot n + 89.29) \cdot tu$

Table 3.5: Comparison of the algorithms

BIN is a bit less efficient than FSS in the best case but still much better than CONST. In the average case BIN performs better than CONST but worse than FSS, given its complex internal logic. In the worst case, BIN is the second best choice and can abort an inconclusive search much faster than FSS. Therefore, overall BIN is a good choice, given that it performs nearly as good as FSS in the best and even much better than FSS in the worst case.

CONST, the standard approach always using the maximum field strength naturally leads to a lot of unnecessary energy waste. CONST however offers an otherwise unmatched performance in worst case scenarios. No other algorithm can abort an inconclusive search that fast. Nevertheless, CONST does perform very badly in scenarios with many tags close to each other, whereas the other algorithms, trough their incrementally increased or step-wise set field strengths usually do not have any problems finding most or all of the tags. Figure 3.9 graphically demonstrates the behavior of the algorithms as a function of the amount of tags in the field. For a more general view, the results are shown in relation to the abstract time unit tu, and the measured $P_{max} = 0.50\,\mathrm{W}$.

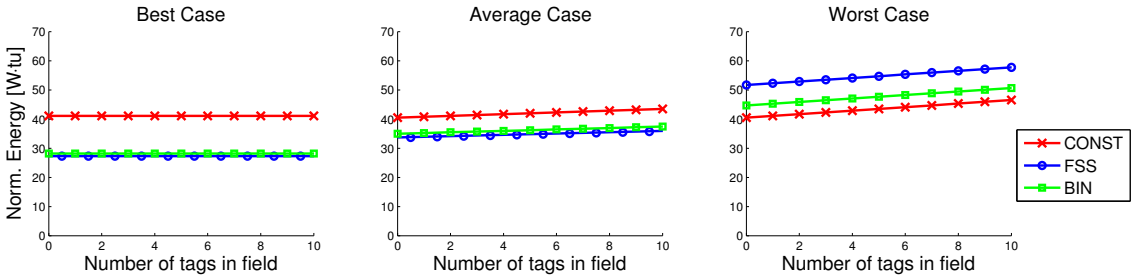


Figure 3.9: The different cases of all algorithms

3.5 The Complete Design

The last sections explained how a communication with the NFC reader is realized, how the target operating system is enhanced to support external NFC devices, and finally also how the field strength scaling algorithms are designed in order to save as much energy as possible. This section concludes the chapter on the design as it describes how all the different components are actually interconnected and demonstrates an overview of the entire design used for this thesis.

The algorithms described in the previous sections are integrated into a single Android program, to be called *proof of concept application*, or simply PoC. It also includes the necessary Android client in order to reach the web service and to issue commands to the reader and the tags. Requests regarding which tag to read from and which algorithm to use for that matter are sent through the remote control interface and passed to the PoC. This approach was selected for this thesis because it allows to assemble use case scenarios and measure their actual energy consumptions, given that every single program step can be triggered inside the PoC in time with the respective measurement.

Alternatively, the desired behavior of the PoC can of course be directly integrated into its program flow. A real life application running on a mobile device would not rely on requests being sent by the remote control interface but rather use a finished program logic to achieve its goals.

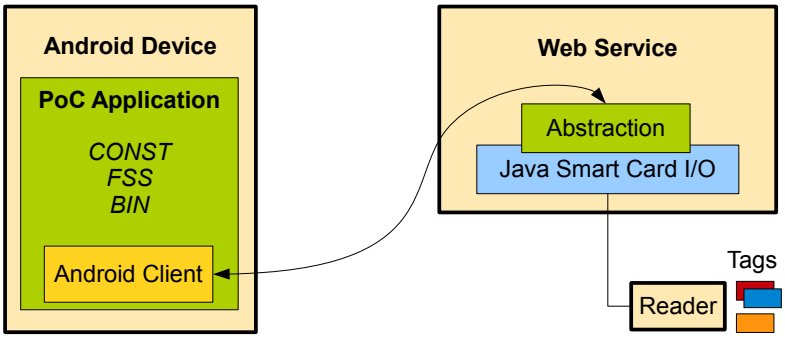


Figure 3.10: The complete design

As shown in Figure 3.10, the PoC executes a certain algorithm and tries to establish a communication with a specified tag by sending commands through the Android Client,

which forwards them to the smart card interface inside the web service. Any additional components which are required during measurements only, like the Windows client or the actual measurement device, are not shown here. Generally speaking, four components are part of the actual design of the algorithms,

- an Android device, including the PoC application,
- a web service executed on a Windows PC, including the smart card interface and its abstraction layer
- the NFC reader hardware,
- as well as the tags themselves.

Chapter 4 will cover the implementation of all software components shown here in detail and explain the actual commands used by the algorithms to control the reader's behavior.

Chapter 4

Implementation

4.1 Used Tool-Chains

In order to realize all the objectives of this thesis, a broad range of different tool-chains and software packages had to be used. Figure 4.1 demonstrates the actual order in which these components were actually utilized,

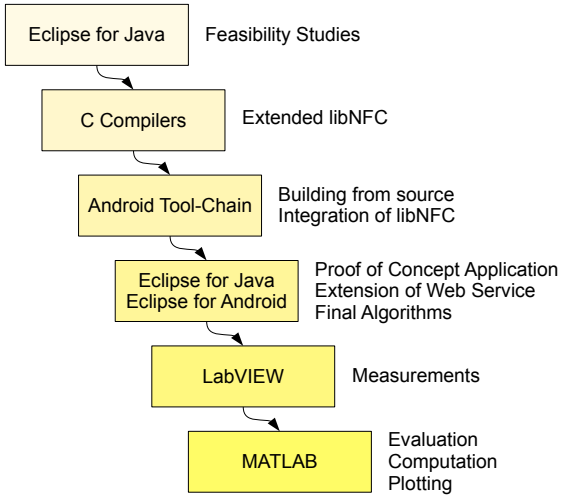


Figure 4.1: Software packages and tool-chains used in this thesis

- the open-source integrated development environment (IDE) *Eclipse for Java*[1] was used right at the beginning of this thesis in order to executed first tests during the feasibility studies,
- temporarily diverse *C compilers* for Linux were used to build an enhance version of the libNFC interface,
- Android's *official tool-chain*[2] was also utilized to compile the system from its sources, and to directly add the extended libNFC to Android's low level libraries,

[1] http://www.eclipse.org/ [2012-19-09]
[2] http://source.android.com/ [2012-09-19]

- since the approach of the last two steps lead to several unsolvable issues, the development returned to *Eclipse for Java* to extend the web service as well as to engineer the algorithms in their final form. *Eclipse for Android*, an extension plug-in, was used to implement the proof of concept application that is executed on the target hardware,
- for the measurement process itself *LabVIEW* was utilized to acquire the resulting data from the measurement device,
- and finally, *MATLAB* allowed to evaluate the measured results, compute equations, and present data in many different forms like plots or tables.

4.2 Software and Hardware Components Involved

This section provides additional implementation-based information for important software as well as hardware components of this thesis. Being involved are

- a NFC reader that communicates with multiple tags,
- ten different tags of MIFARE Ultralight type [NXP08], distinguishable by their unique UIDs, each also containing some bytes and a lot of white spaces stored on them, which can be transmitted in order to demonstrate that reading operations are also possible,
- a mobile device that connects to the reader, in this case TI's evaluation board AM37X, which represents a smart phone or tablet, users in real life scenarios would take with them. The mobile aspect of that device is very important, as especially this kind of hardware requires smart power-aware designs to allow acceptable battery life times,
- an Android operating system installed on that mobile device. Part of this thesis' goals of course is to enable multi-tag support in Android, as a result, such an operating system is used on the mobile device,
- a proof of concept application running on the device. It is on the one hand controlled from the outside and told what to do, and on the other hand forwards requests to the algorithms it implements,
- the algorithms developed for this thesis. Their job is to clearly demonstrate how the energy consumptions of a standard approach towards multi-tag scenarios differ from fully optimized designs that were especially developed for that matter. The algorithms are three independent software components which are controlled by the proof of concept application they were integrated into,
- finally the measurement device that includes different test cases, lists of requests executed stepwise to create different scenarios and to control the proof of concept application from the outside, while the measurement device acquires the voltage and current consumption at the same time.

4.2.1 The NFC Reader

This thesis utilizes the multi-format contact-less card reader ACR122U by Advanced Card Systems (ACS)[3]. It includes the commonly used NFC transceiver module PN532 by NXP[4]. A software development kit, provided by manufacturer ACS, can be purchased in bundle with the reader, however it only offers a few simple additional features like switching the LED that is mounted on top of the reader, or turning on the integrated buzzer. No direct low level programming of the NXP transmission module is possible that way, multi-tag and especially field strength scaling operations however require full access to the PN532 hardware and its internal registers. Figure 4.2 demonstrates the block diagram of the PN532 module. It includes a lot of different components, upon which the contactless interface unit (CIU) is the only one of real interest for this thesis. According to the data sheet [NXP11], the CIU includes demodulator and decoder for MIFARE tags as well as error handling and framing functionality. It provides a RF level detector required to see whether interfering external fields are present at the moment as well as internal registers containing user adjustable parameters that set the module's actual transmission characteristics.

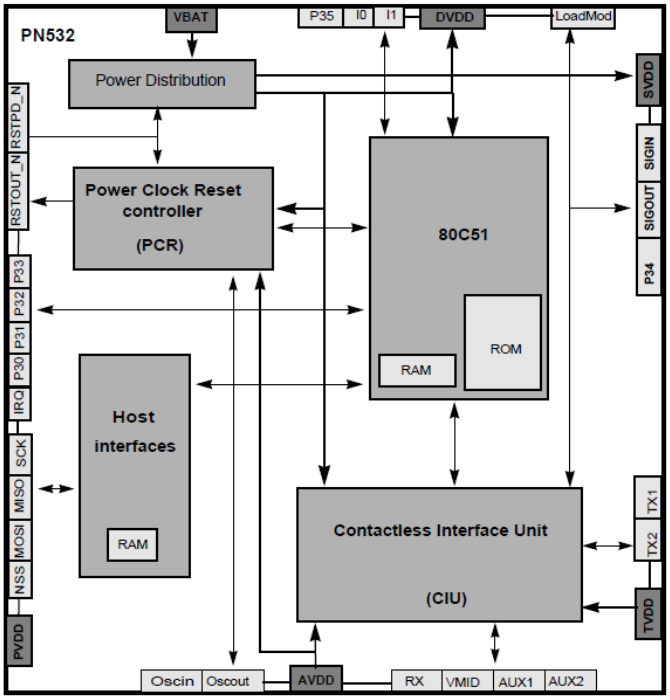


Figure 4.2: The block diagram of the PN532 NFC module used in this thesis [NXP11]

APDU commands[5] were developed to provide a standardized communication with different smart card components. It is possible to control readers and transmit data back and forth to and from tags. The reader for this thesis however introduces another abstraction

[3] http://www.acs.com.hk [2012-09-19]
[4] http://www.nxp.com [2012-09-19]
[5] Application Protocol Data Unit

layer above the APDUs that shuffles the byte order and is therefore incompatible with the commonly used APDUs as describe in the data sheet of the transmission module [NXP11]. Although NXP provides a well written technical documentation on the PN532 that lists every single APDU the module understands, none of these byte sequences could be used as is with ACS' reader. Tedious trail and error research finally led to an understanding of the differences between common APDUs and the ones used by the ACR122U.

4.2.2 The NFC Tags

In order to test, develop, and finally demonstrate all the algorithms of this thesis, multiple NFC tags were necessary. Infineon kindly offered to lent the author any number of tags required, the amount of ten was found to be sufficient. One has to keep in mind that NFC, in contrast to RFID, was never designed to support such great amounts of tags, especially given its short transmission ranges and the resulting closeness in which all tags have to be kept.

The tags used in this thesis are all based on the MIFARE Ultralight technology, originally developed by NXP but provided in this case by Infineon. Figure 4.3 demonstrates the block diagram of such a tag. On the left side, the antenna connected to the associated RF-interface can be seen. As described in the functional specification [NXP08], the interface is used for modulation as well as demodulation of data. Additionally it is used as a rectifier for the alternating current that is produced by the reader's field, and as a clock regenerator as well as a voltage regulator. The digital control unit offers an anti-collision component, used to support the reader in the process of acquiring one single UID in multi-tag situations. And additional EEPROM[6] interface provides read and write access to the tag's internal memory. Most importantly a command interpreter is included on the tag that understands a certain amount of commands which are actually required by all algorithms of this thesis. The last block on the far right of Figure 4.3 represents the EEPROM memory itself.

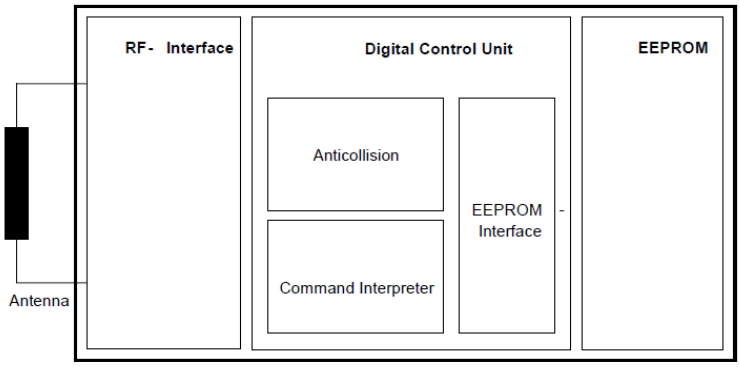


Figure 4.3: The block diagram of MIFARE Ultralight tags used in this thesis [NXP08]

The tags fortunately exhibit ten different UIDs, a fact that is not always guaranteed with tags produced by other companies. According to [NXP08], the UIDs were hard-coded

[6]Electrically Erasable Programmable Read-Only Memory

during the manufacturing process and can therefore never be overwritten anymore. Tags with unique UIDs are very important as this makes it easy to distinguish them, even if huge amounts of them are present in the near-field at the same time. Whenever algorithms are searching for a specific tag, they acquire the different UIDs, compare them to a known and wanted value, and only establish a full communication with the wanted tag.

All ten tags contain a few locked bytes of production specific data and additionally a lot of free space afterwards. For this thesis to demonstrate reading operations the white spaces do not present any problem as the actual data being read is of no interest. It is sufficient to only show the specified amount of data, 1 KB in this case, is indeed successfully read and forwarded to the proof of concept application running on the mobile device.

4.2.3 The Mobile Device

Given that this thesis especially aims at mobile usage of NFC scenarios, a portable device was selected to develop and test all algorithms on. Texas Instruments provides an evaluation module AM37X[7] that besides Windows and Linux supports different versions of the Android operating system and also offers USB ports required to connect it to the reader. This board was provided to the author to carry out further research. It represents mobile devices in general, like smart phones or tablets, and can be used as a proof of concept platform. It enables one to create different scenarios with multiple settings that approximate very well, how various commercial products would behave in such situation. Therefore, the evaluation board allows to test for multiple devices at once, a huge advantage that would not exist if only the final target product would have been used instead.

Pre-built or user modified and self-compiled images of Android systems can easily be deployed on the evaluation board by formatting a SD card[8] and booting the AM37X with it. For this thesis, during development a lot of different versions of Android were in use, pre-built images directly acquired from TI as well as versions totally modified and compiled by the author himself. This is discussed in much more detail in Section 4.3.2.

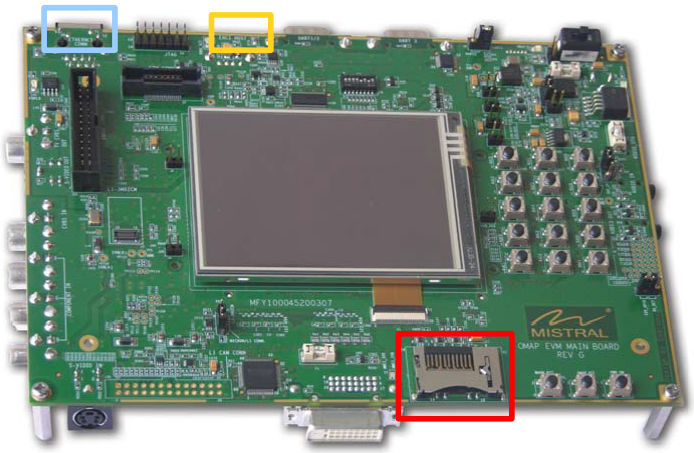


Figure 4.4: The evaluation board AM37X and its different connection possibilities [Ins10]

[7] http://www.ti.com [2012-09-19]
[8] Secure Digital Memory Card

Figure 4.4 demonstrates the three connection possibilities of the AM37X that were especially important for this thesis. The SD card slot, shown here in red, can be seen in the lower part of the picture. As discussed above, it was used to provide modified operating systems and boot the board. The USB bus on the other side was required to connect the NFC reader, although ultimately, Android's issues with USB prohibited its use. It is shown here in orange, slightly out of view. The Ethernet port which is required to connect the board to the Internet to reach the interface's web service used for measurements is shown here in light blue in the upper left corner of the picture.

4.3 Implementation Process

The whole process that lead to the final results is discussed in this section. Figure 4.5 shows the complete implementation flow, whereas the following sections explain what work has been done in each of the steps.

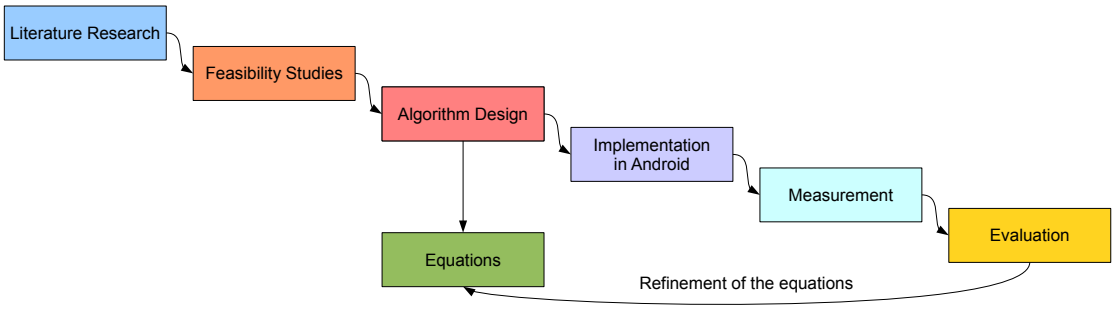


Figure 4.5: The steps during this thesis

4.3.1 Step One - Literature Research

Prior to any design, extensive literature research had been carried out. Interestingly, it showed how little work has already been done in the field of NFC multi-tag applications, especially in regard to power-awareness and energy optimization. Any ideas concerning how to realize the complex objectives of this thesis had to be developed from scratch, no preliminary literature was found for most of the topics covered in this document. Seemingly this field has been quite neglected so far, even though it offers a lot of great potential, even for commercial use.

The Android operating system is discussed in many different books and papers and additionally the software development kit is documented quite well. Nevertheless none of these sources cover anything close to what this thesis aims to achieve. The only fact very well documented is that Android does not provide any native support for multi-tag applications at all. As a result, any design had to be built up from scratch by the author himself and this very implementation provides such functionality for the first time ever in the Android operating system.

4.3.2 Step Two - Feasibility Studies

Given that a lot of the objectives of this thesis seemingly broke new ground, feasibility studies were carried out prior to any subsequent steps to figure out whether the goals could even possibly be reached. As discussed before, first tests were done in Windows with the help of the Java Smart Card I/O, which proved that the goals set for this thesis were indeed reachable, at least in Windows. Unfortunately, the actual target platform Android does not support the Smart Card I/O at all, and as a result, another approach had to be found. Different projects using proprietary NFC software stacks to communicate with the hardware as well as NXP's forum reference implementation (FRI), described in [NXP07a] were inspected, in order to find an interface that could possibly be ported to Android. The design and structure of the FRI and the documentation available however was considered to be far too complex for any further development to be based on it. The open source library libNFC[9] offered a very well documented software stack developed by a community over several years. It did in no way provide direct solutions to any of the objectives of course, however served as a quick and rather easy way to test first ideas directly in Linux which could later possibly be ported to Android. The original libNFC was heavily enhanced to provide advanced multi-tag support as well as field strength scaling functionality it did not include before. Multiple C compilers for Linux were used to port the library to Android's specific requirements.

Subsequently Android 2.3.4 was acquired in source code form[10] to be extended and compiled as will be explained in the following steps,

- given that building Android requires a Linux based host system, from the first step of acquiring its source on, everything was done in Ubuntu 10.04[11],
- a tool called *Repo*[12] was used to receive all Git[13] repositories which are containing Android's source code,
- the source code was checked out by issuing the following commands which initiate a download of every necessary source file,

  ```
  repo init -u https://android.googlesource.com/platform/manifest
  repo sync
  ```

- the download process requires several hours, as does the next step that actually compiles the source code for the first time,

  ```
  source build/envsetup.sh
  ```

- afterwards any library that needs to be added to Android's subsystem can be included by issuing the command *mm* inside the library's root folder. The Android tool-chain then only links this library without recompiling anything else. This of course shortens the compile time required after every change of the library's source code to a few minutes.

[9] http://www.libnfc.org [2012-09-19]
[10] http://source.android.com/source/downloading.html [2012-09-19]
[11] http://www.ubuntu.com/ [2012-09-19]
[12] http://source.android.com/source/version-control.html [2012-09-19]
[13] http://git-scm.com/ [2012-09-19]

During the course of the feasibility studies most of the functionality has already been integrated into the enhance libNFC that was included into a newly compiled Android system. Unfortunately, several issues with missing Linux driver prevented the author from using this approach any further. The manufacturer of the NFC reader used for this thesis did not supply the required driver, nor was any essential information on how to develop such a component for this specific hardware from scratch ever provided to the author, even though many respective discussions with the actual customer service took place several times. As a result, both libNFC and all the extended components that were integrated into Android had to be dropped altogether and a totally different approach had to be found. This fact is also shown in Figure 4.1 which presents the chronological usage of different software packages, and lists *Eclipse for Java* following the *Android toolchain* a second time. The implementation process returned to the functionality that was developed during the very first tests in Windows and used the *Java Smart Card I/O* as an interface to communicate with the NFC hardware.

4.3.3 Step Three - Algorithm Design

During algorithm design the requirements for applications to be able to communicate with multiple NFC tags in a power efficient way were analyzed. The limits of the technology at hand were inspected and different solutions were considered. Simple approximation equations were developed to compare the different approaches and to find the minimum amount of steps that lead to a certain goal.

The Evolution of the Algorithms

Right at the beginning of the algorithm design the standard approach of dealing with multiple tags inside the near-field, later called *CONST*, was implemented. Although the transmission module of this thesis' reader offers a large variety of valuable commands ready to be used [NXP11], no such thing as a multi-tag functionality per se exists. In order to read from a tag for example, its full UID has to be known and a communication has to be established first. The transmission module provides a command *InListPassiveTarget* that activates the next reachable tag, if any is present. The main goal of CONST of course is to step through all available tags and only read from the wanted one. Unfortunately, simply calling *InListPassiveTarget* multiple times is not the right way to go. If nothing else is being taken care of, the transmission module always returns the same tag, no matter how many others are actually present and could be reached instead.

Therefore another command *InDeselect* is required. It deactivates a tag that was found by *InListPassiveTarget*, and as a result the next such command establishes a communication to a different tag. Unfortunately, an additional issue has to be considered as well. Whenever the last active tag has been deactivated, the transmission module starts to reach all of the tags yet again. Therefore, after the last tag has been inspected, the search repeats itself with the first tag and carries on in that matter for ever. As a result, CONST has to include an additional list that stores which tags have already been inspected, so that whenever a tag would be inspected a second time, that communication is canceled beforehand.

If CONST or any other algorithm wants to have another go at searching for a certain tag, an additional issue occurs. Those tags that are currently deactivated have to be reset

by a command called *InRelease*. If neglecting to do so after each full pass of an algorithm or before it, further passes are not able to work with those tags anymore.

One can see that CONST actually has to use three different commands in a very specific order and include its own logic that knows when exactly to issues which command. A list also has to be maintained that stores already inspected tags, but needs to be emptied at certain points. All in all, it should be clear by now that the transmission module itself does not provide a simple command *CONST* that does all the work. CONST is indeed an algorithm that had to be implemented for this thesis.

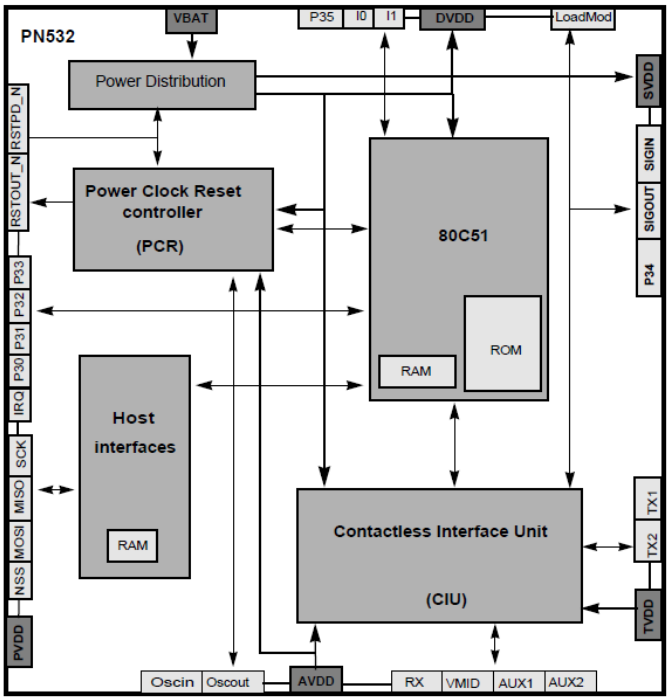


Figure 4.6: The block diagram of the PN532 NFC module used in this thesis [NXP11]

After CONST had been implemented, FSS was designed. It of course is very similar to CONST save for the fact that it is also able to dynamically scale the used field strength. Therefore, in depth research had to be carried out in order to find a way to set the transmission modules field strength. As can be seen in [NXP07c], two different registers can be set to alter the impedance of the n-driver and the p-driver, serial resistances to the antenna of the reader. Increasing said resistances leads to a drop of voltage and consequently to a lower power consumption. The command *RFConfiguration* provides access to many important settings of the module's current transmission mode and also includes parameters that influence the registers described above. Two values *GsPMos* as well as *GsNMos* that define the conductance of the NMOS and PMOS transmitter respectively are the most important parameters for this thesis. By manipulating them through their associated registers, the actual power available for the antenna, which is connected to *TVDD* shown in the lower right corner of Figure 4.6, can be varied and as a result the field strength generated by the reader as a whole can be set.

The additional algorithm BIN was only developed much later, when first tests on the

target hardware components had shown the great potential of the techniques already used. All three algorithms were already discussed and analyzed in depth before in this document. This section however additionally explains several important parts of the actual program flow of the algorithms, especially in regard to the respective APDU commands being issued to the reader.

The Required Commands

In the flow chart of Figure 4.7, the start and end points of the function are presented in green, decisions are shown in red, internal logical steps in gray and actual APDUs that have to be transfered to the reader are colored in blue.

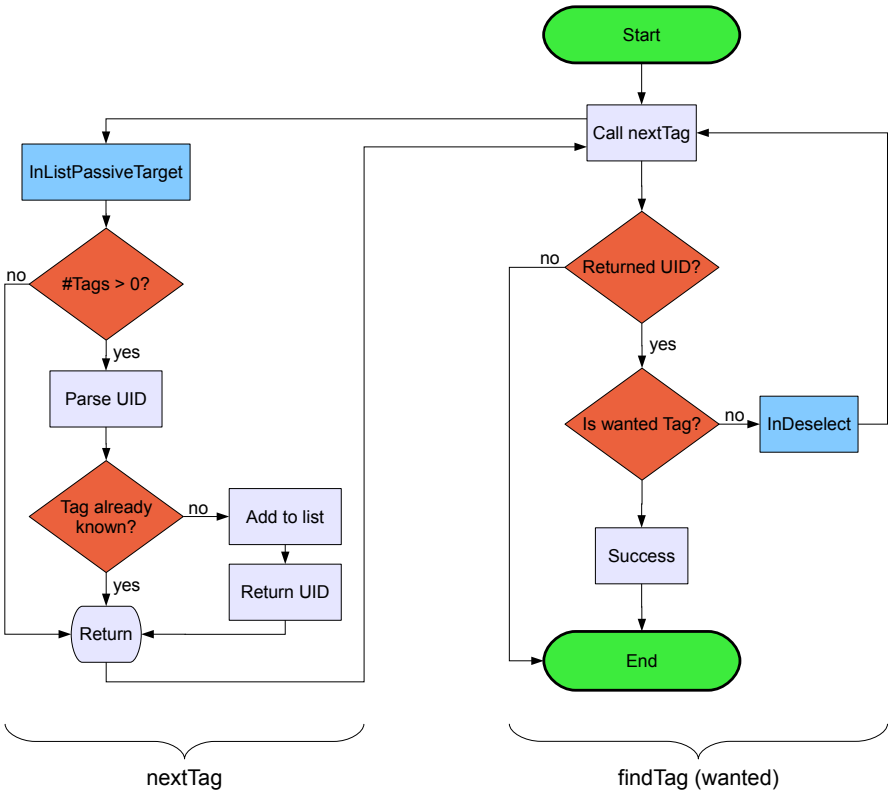


Figure 4.7: The program flow of findTag and its sub-function nextTag

Figure 4.7 shows the most basic method upon which all algorithms of this thesis are based on called *findTag*. In order to determine if a certain tag is reachable, *findTag* is executed which takes an UID as a parameter. Subsequently it uses its sub-function *nextTag* that issues an *InListPassiveTarget* command to list one tag reachable at that moment. If it had not been seen before, its UID is returned to the main function *findTag*, which tests if the UID in fact belongs to the wanted tag. At this point a communication has already been established with that tag and subsequently reading operations for example can be started with the *InDataExchange* command. It allows to read a maximum amount of 16 bytes of data from a certain start address, which is the most the used transmission

module of this thesis can handle at once. Therefore, the algorithms have to issue the read command 64 times in a row while also increasing the start address in order to acquire a total of 1 KB from the tag.

Besides the three APDU commands described above, FSS and BIN also use a forth command, *RFConfiguration*, to set the field strength. Finally, before or after each full pass of an algorithm the tags have to be reset by issuing the fifth command *InRelease*. All in all five different APDUs are required in this thesis to

- list the tags reachable at the moment,
- deselect a tag after communication with it has finished,
- reset the tags seen by the reader at the start of an algorithm,
- set the field strength used by the reader to a specific value,
- read a certain amount of data from the one tag currently selected.

Figure 4.8 demonstrates how one ACR122U specific command actually looks like, which as discussed before features a changed byte order compared to standard APDUs. In this example the request 'inListPassiveTarget' is discussed. Shown here in hexadecimal, the first byte is a simple header. This is followed by three bytes containing only zeros, shown here in blue, which is called the *preamble*. These values always have to be set to zero. Afterwards, shown in green here, is a byte defining how much other bytes are going to follow, in this case four. In red a byte is shown that defines the direction of the communication flow. Given that the application request data from the reader and not the other way around, *D4* is used. What follows in blue is the number of the internal routine the reader should executed, *4A* in this case stands for *InListPassiveTarget*. Finally, shown in black here, two parameters are passed, that tell the reader how often to try to reach one tag and which transfer mode to use.

```
FF00000004D44A0100
```

Figure 4.8: The ACR122U specific APDU for InListPassiveTargets

The four other APDU commands discussed above consists of the following bytes

deselect	FF00000003D44400
release	FF00000003D45200
setField	FF0000000ED4320A59F4xx114D856164266287
read	FF00000005D4400130yy

with xx being the actual value the field strength has to be set to and yy being the start address of where to read from in case the wanted tag was found and a communication was successfully established.

4.3.4 Step Four - Implementation in Android

Since Android 2.3 NFC support is part of its software development kit[14] (SDK). At the moment of writing, two main functionalities[15] of NFC are provided by the SDK, acquiring *NFC Data Exchange Format* (NDEF) data blocks off a target like a tag, and forwarding them to other Android devices via *Beam*. Unfortunately both features are very high level functionalities based on many layers without any direct access to hardware or even any of the lower layers. Android offers no multi-tag support at all and very limited access to external hardware like readers. Altering internal registers of the reader, which proved to be of high priority, is also impossible. As discussed before in Chapter 3, additional issues that occurred with Android and Linux drivers as well as USB ports, were by-passed by integrating the Java Smart Card I/O into the web service component. Therefore, the actual communication with the reader is done in Windows and Android's NFC stack is not used at all.

The logic of all algorithms is integrated into a Java based Android program, to be called *proof of concept application* (PoC), which is executed directly on an Android device. Additionally, the associated Android client of the remote control interface is included in the PoC as well in order to enable it to reach the web service. A new method called *sendAPDU* was created that takes a sequence of bytes, encoded as one String. The PoC can use the method to pass actual APDU commands, similar to the way it would return results or state information to the web service during measurements. The big difference however is the fact that the web service does not store these requests, to let them be picked up by another client of for example a Windows system, but rather directly forwards them to the reader and instantly returns any results of the reader back to the Android client. This provides the Android based PoC with full control over the NFC reader, although it is actually connected to a Windows PC. Figure 4.9 demonstrates how the PoC transmits an APDU to the web service which forwards it to the reader.

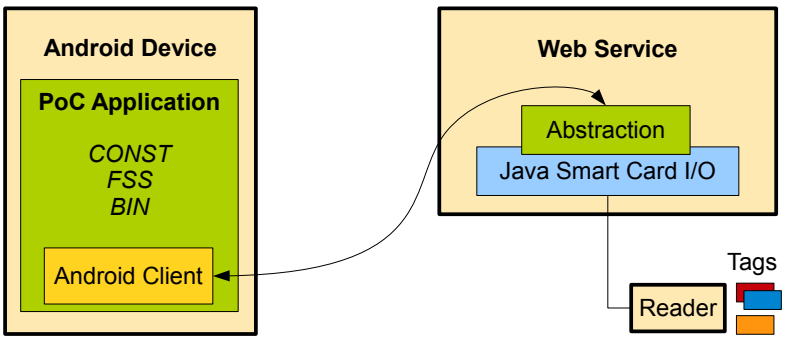


Figure 4.9: Forwarding commands to the reader

The Two Layers of the Algorithms

All three algorithms consist of two separate layers, the logic layer which is directly included in the actual PoC, as well as the communication layer which is a remote component

[14] http://developer.android.com/reference/android/nfc/package-summary.html [2012-09-19]
[15] http://developer.android.com/guide/topics/connectivity/nfc/nfc.html [2012-09-19]

executed on the web service. Every single step an algorithm has to take is implemented inside the logic layer, which also evaluates the respective results. Based upon these, decisions are made that influence the subsequent program flow. Each of them can be expressed as a sequence of APDU commands that have to be transmitted to the reader in order to change its behavior to reach a specific state. The logic layer prepares the bytes that have to be sent but does not actually forward them to the reader.

The transmission layer however, being part of the web service, is directly connected to the NFC reader via USB. It consists of two different parts that were already discussed before in Chapter 3, an abstraction layer that serves as an interface and the much more complex underlying Java Smart Card I/O. APDUs that were passed to the web service with the *sendAPDU* method are forwarded to the abstraction layer. It only provides one single encapsulated method of the Java Smart Card I/O to actually issue commands to the reader, *send*, as well as some additional initiation and clean up functionality required to establish communications and close them properly in the end. The underlying Java Smart Card I/O acquires access to the first available reader hardware and creates a card object by executing

```
CardTerminal terminal = terminals.get(0);
Card card = terminal.connect("*");
```

Commands are sent to the reader by establishing a basic channel

```
CardChannel channel = mCard.getBasicChannel();
ResponseAPDU r = channel.transmit(new CommandAPDU(apdu));
```

Therefore, the abstraction layer only has to offer one single method *sendAPDU* that takes the actual APDU command. The response of the reader is returned through the *sendAPDU* method as well. The actual implementation of all internal steps is encapsulated and hidden. Figure 4.10 demonstrates an overview of the two layers discussed in this section.

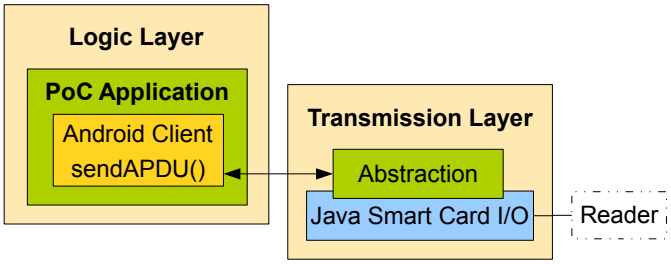


Figure 4.10: The two layers of the algorithms

The separation was done for a good reason, to realize a reader hardware independence. All three algorithms were designed in a very general way, the steps they have to take are the same no matter which NFC reader is actually used. They consist of decision-making logic only that is the same independently of the hardware. If for example future projects require a totally different reader, it is sufficient to adapt the transmission layer that is part of the web service, whereas the actual logic layer of the algorithms can indeed be reused. Additionally, the transmission layer and its underlying Java Smart Card I/O

are implemented in native Java, which can be executed on many different platforms. Therefore, the transmission layer can instantly be used on various other platforms besides Windows as well. It may also be separated from the web service altogether and directly be included into a mobile device. For commercial standalone products both layers would be implemented in a single device, without any of the indirection of the web services.

4.3.5 Step Five - Measurement of the Proof of Concept Application

The measurement setup for this thesis incorporates all components discussed before. As can be seen in Figure 4.11, on top of all the MATLAB based measurement suite is used to control all devices involved. It starts the PoC application including the algorithms and at the same time requests the measurement device to acquire the first current and voltage samples from the reader with a test prod, shown here in red.

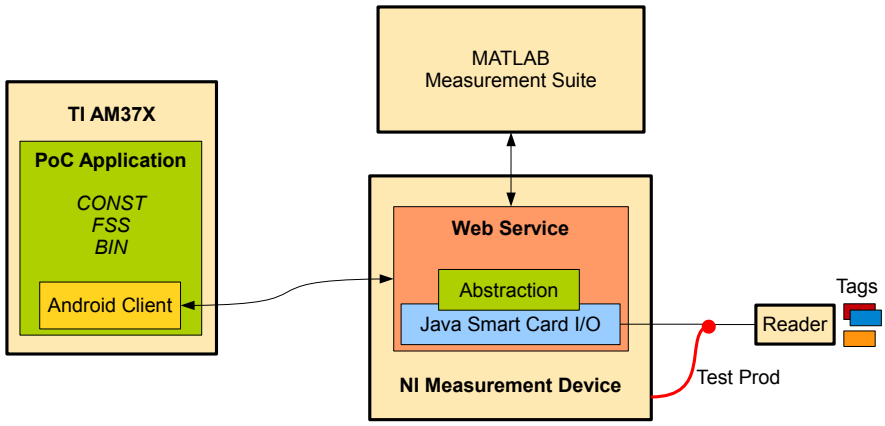


Figure 4.11: The measurement setup for this thesis

For actual measurements a setup was chosen where the web service is actually executed directly on the measurement device and the reader is connected to it as well. Requests regarding which algorithm to use and what tag to search for are sent from the measurement suite to the web service and forwarded to the evaluation board AM37X. The algorithms execute a certain logic and transmit APDU commands to the web service which itself forwards them to the reader. In the meantime the measurement device acquires the respective voltage and current samples.

This setup provides the huge advantage of being fully automated. No operator ever has to manually start or stop any of the devices independently. Therefore, during the course of this thesis various test cases were designed and interesting tag locations were considered, subsequently the measurement was started once and could be left alone afterwards. This approach actually worked extremely well and no issues with the remote control interface ever occurred.

Debugging During the Measurement Process

The evaluation board AM37X can be connected directly to a computer in order to debug the algorithms it executes during measurements. With the help of the so called *Android*

Debug Bridge (adb)[16] state information as well as error messages can be transmitted to a computer and be subsequently evaluated. The Android operating system however has to be configured beforehand in order to activate USB debugging. Additionally, the adb also allows to transfer new and improved versions of applications to the AM37X device whenever required.

4.3.6 Step Six - Evaluation of the Results

The superior META[:SEC:] project already provided a few evaluation and plotting scripts. The raw values acquired by the measurement device, both current and voltage samples, are multiplied in order to receive the power consumptions and these results are then delivered in the form of *mat* files, MATLAB specific container files that include all power values of one pass for each of the algorithms. This of course simplifies any subsequent comparison, and bundles data that is relevant for one test case in a single file, thus enabling a much better overview.

Nevertheless, a lot of additional MATLAB scripts still had to be developed in order to realize the extensive subsequent evaluation of the results, including methods that

- compare a varying amount of different measurements in diverse formatted plots,
- calculate the relations of the energy or power consumptions of the different algorithms in each use case,
- automatically predict the energy consumptions with the approximation equations developed in Chapter 3 and compare the results to the values actually measured,
- enable the user to interactively select certain samples for closer inspection,
- allow to color specific sections of a power profile in order to explain what happens at this moment,
- present the actual field strengths of each shell and compare them,
- demonstrate the best, average, and worst case behavior of all algorithm in one single plot for direct comparison and a better overview,
- smooth and filter power profiles during inspection in order to gain a more general view of the data at hand.

4.3.7 Step Seven - Refinement of the Energy Equations

The measurement phase consisted of multiple iterations that led to a broad range of results and new insights in each pass. Therefore, all algorithms could be incrementally improved whenever new relations not covered yet were detected. As a result, the approximation equations had to be adapted each time as well, in order to represent the actual energy consumptions in the most accurate way, while still remaining as abstract and general as possible. Over the course of this thesis three totally different approaches to develop these

[16] http://developer.android.com/tools/help/adb.html [2012-09-19]

equations were considered that covered entirely diverse aspects of the algorithms. Massive refinement resulted in the final form of the equations presented in Chapter 3.

Concerning their actual implementation, the equations were integrated into other MATLAB scripts that are automatically executed during evaluation and plotting. The results actually acquired from the measurement device are compared in relation to the predicted values of the equations. The next chapter will proof that they cover by far enough characteristics of the actual hardware used in this thesis in order to allow predicting the energy consumption very accurately beforehand. This of course allows developers to evaluate their ideas early on in the design process and to make important choices before actually implementing new algorithms.

Chapter 5

Results

In this chapter the extensive measurement results of this thesis are presented and further analyzed. On the one side, the possibilities as well as the limits of current NFC hardware like readers and tags are shown and answers are provided to explain certain behavior. On the other side, interesting use cases are delivered that simulate scenarios very likely to happen everyday in the near future. All use cases are either relevant to new developments or represent improvements of already existing applications.

5.1 Use Case 1 - 'Reading Ten Tags'

Ten NFC tags, directly stacked upon each other, are positioned on the surface of the reader. The goal of this setup is to show how a greater amount of tags interact, especially when being that close to each other. As opposed to RFID, NFC tags were not designed to allow many different targets inside a very small space [Fin10], and this section will demonstrate that ten tags represent quite a problem for the reader of this thesis when using the standard approach of CONST.

Figure 5.1: Setup for 'Reading Ten Tags', ten tags stacked upon each other

The tags, as shown in Figure 5.1, were labeled 1 to 10, small and thin stickers were put on the surfaces to mark them, but it was taken care not alter the tag's original height

as this would falsify the measured result. All algorithms developed for this thesis were tested against each other in ten iterations, changing the wanted tag from T_1 up to T_{10}, which led to the insights that

- CONST can only reach one tag T_9, no other tags are ever seen,
- FSS as well as BIN are able to reach the tags T_1 to T_4 and T_7 to T_{10}, the remaining two tags cannot be detected however.

Therefore, in order to fully demonstrate the possibilities and limits of the algorithms combined with the available hardware, three different scenarios are further inspected,

- one with the wanted tag being T_9, it shows how even though the standard approach can reach this tag, the optimized algorithms can operate much more energy efficient,
- one with T_2 being chosen, a situation is created where the standard approach totally fails to read from the tag,
- and finally one looking for T_6, which demonstrates the limits of the technology, as no algorithm successfully communicates with the tag.

5.1.1 Searching for a Tag all Algorithms Can Find - T_9

Given that CONST can only detect T_9, this tag had to be chosen in order to demonstrate a scenario where all algorithms are able to find and communicate with the wanted tag.

CONST Algorithm

CONST sets the field strength to the maximum value once at the beginning of its execution, which is shown in Figure 5.2 in red. It is able to discover the wanted tag T_9 right away, a single list operation, shown in green is sufficient, after which CONST successfully reads 1 KB from the tag. It however obviously wastes a lot of energy, especially during the reading operations, using the maximum field strength for the entire time range.

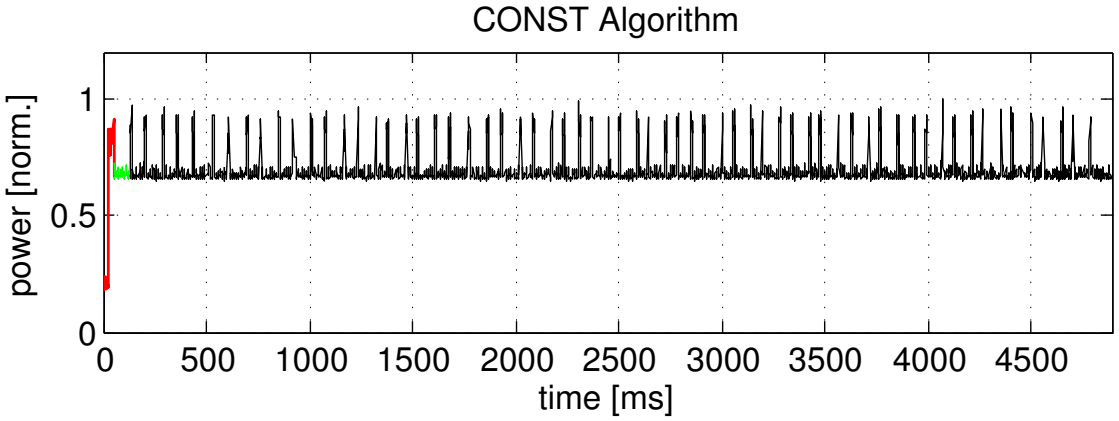


Figure 5.2: Power consumption of CONST when looking for tag T_9

FSS Algorithm

FSS starts by setting the field strength to the lowest possible amount with a power consumption of P_1. Similar to CONST, it reaches T_9 right at the beginning and exhibits the exact same general discovery behavior. One field strength setting operation and one list operation is carried out, each shown in red and green in Figure 5.3. The subsequent reading operations however clearly demonstrate the big advantage of the field strength scaling principle. FSS is able to execute all reading cycles, shown in Figure 5.3 as exactly 64 peaks, with a much lower field strength when compared to the standard approach of CONST.

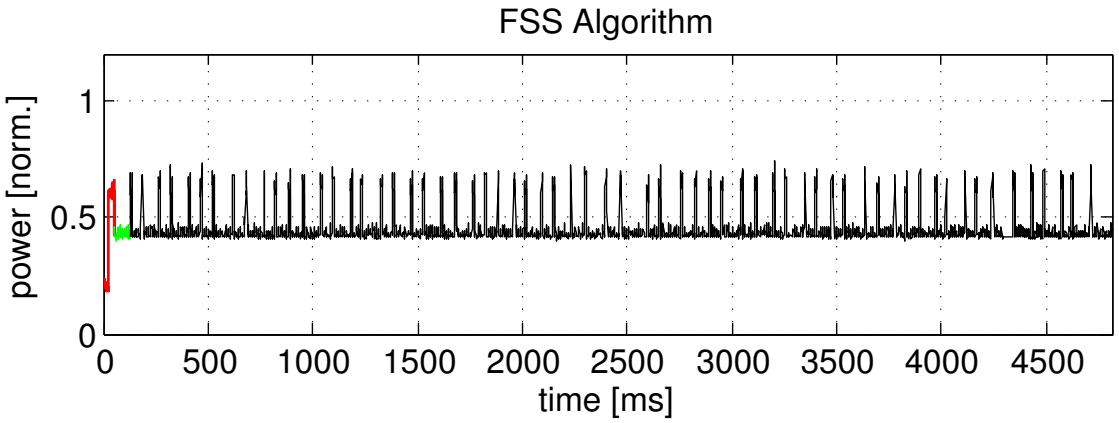


Figure 5.3: Power consumption of FSS when looking for tag T_9

BIN Algorithm

BIN exhibits two additional steps in its power profile, shown in Figure 5.4. It starts in shell 2 and executes one list operation that discovers the wanted tag T_9, each shown in red and green. BIN tries to reach the tag with an even lower field strength in shell 1 and subsequently repeats the two operations, once again shown in red and green.

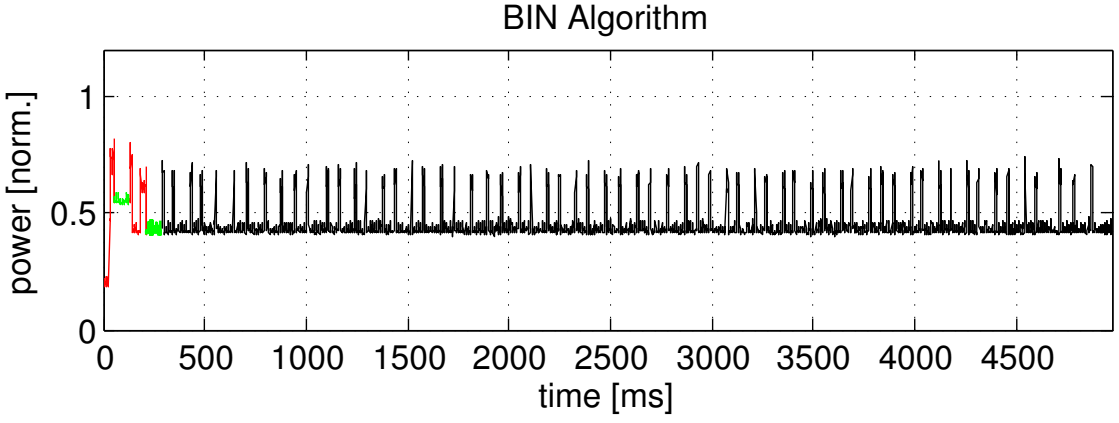


Figure 5.4: Power consumption of BIN when looking for tag T_9

Subsequently, 1 KB is being read with a power consumption of P_1, equivalent to FSS. Compared to the other algorithms, BIN requires twice as much time for its tag discovery process, as it executes two field setting and two listing operations.

Comparison of the Algorithms in the Case T_9

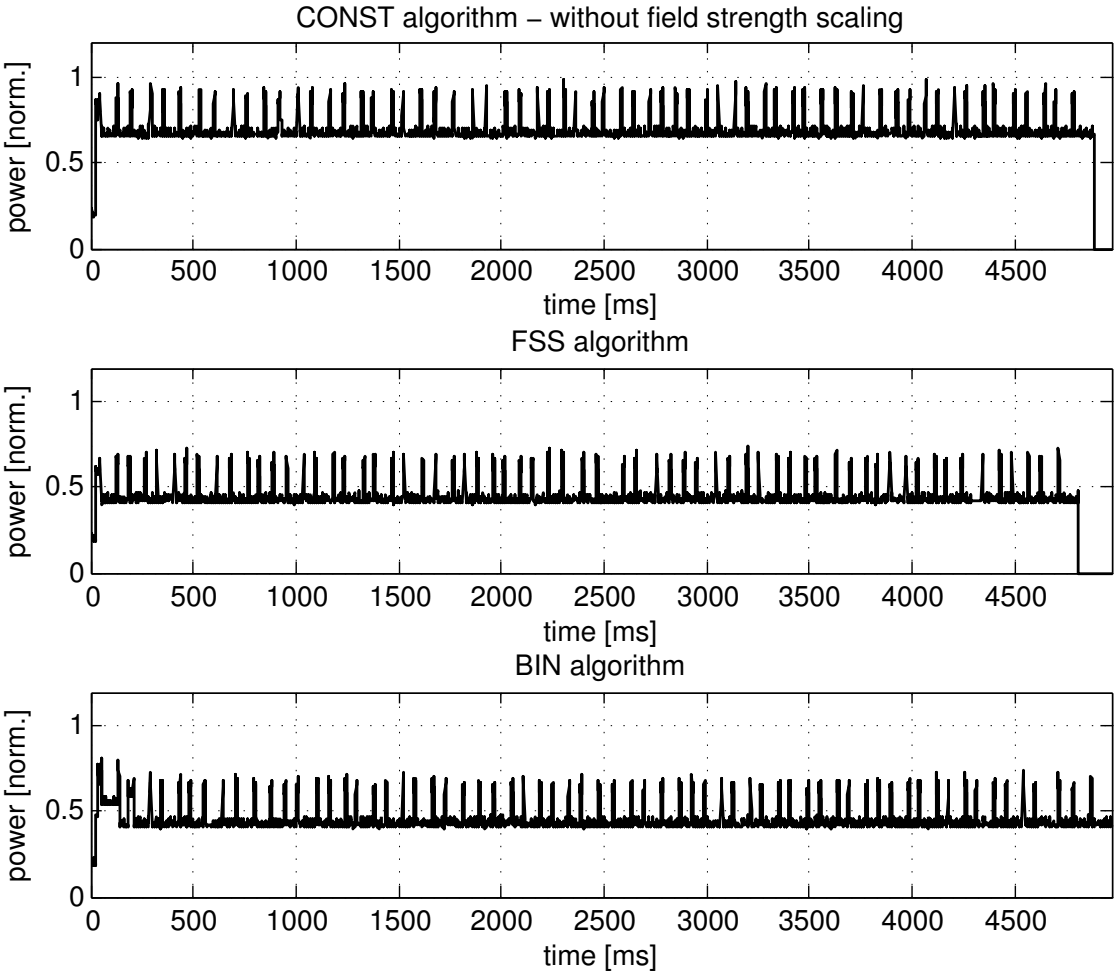


Figure 5.5: Power consumption of all algorithms when looking for tag T_9

Figure 5.5 shows a comparison of all algorithms when searching for the wanted tag T_9. In Table 5.1 the associated total energy consumptions are presented. It is clear to see, that FSS is far more energy efficient than the standard approach of CONST, closely followed by BIN. This scenario explains the great advantage of the field strength scaling principle, as both optimized algorithms are able to successfully read from the tag with the lowest field strength possible, whereas CONST has to operate with the maximum amount.

	Energy [%]
CONST	100
FSS	65.53
BIN	68.13

Table 5.1: Comparison of all algorithms when looking for T_9

Comparison of the Approximation Equations and the Measurement Results

In Chapter 3, equations that describe the energy consumptions of all algorithms were developed step by step. With the actual results shown above, these equations can now be reviewed.

CONST has to set the field strength once to the maximum amount and subsequently executes one listing operation, with $n_{tags-listed} = 1$ this leads to

$$t_{set} = 1 \cdot tu$$

$$t_{find} = 1.2 \cdot n_{tags-listed} \cdot tu = 1.2 \cdot tu$$

and finally 1KB is being read from the tag, which always takes

$$t_{read} = 80 \cdot tu$$

The total time required can therefore be written as

$$t = t_{set} + t_{find} + t_{read} = 82.2 \cdot tu$$

and with the maximum field strength consuming

$$P = P_{max} = 0.5\,\mathrm{W}$$

the resulting energy consumption is

$$E = P_{max} \cdot 82.2 \cdot tu = 0.5 \cdot 82.2 \cdot tu = 41.1 \cdot tu$$

FSS requires the exact same operations during tag discovery, as well as the reading process, however executes all of them with the lowest possible field strength. Therefore,

$$t = t_{set} + t_{find} + t_{read} = 82.2 \cdot tu$$

whereas the required power is only

$$P = P_1 = 0.663 \cdot P_{max} = 0.663 \cdot 0.5\,\mathrm{W}$$

and the resulting energy consumption is

$$E = 0.663 \cdot 0.5 \cdot 82.2 \cdot tu = 27.25 \cdot tu$$

BIN executes one field setting as well as one listing operation in shell 2, $P = P_2$, and afterwards in shell 1, $P = P_1$ which results in

$$t = t_{set} + t_{find} = 2.2 \cdot tu$$

each time. Subsequently BIN reads the tag in shell 1.

$$t_{read} = 80 \cdot tu$$

This leads to a total energy consumption of

$$E = P_2 \cdot 2.2 \cdot tu + P_1 \cdot 2.2 \cdot tu + P_1 \cdot 80 \cdot tu$$

$$P_1 = 0.663 \cdot P_{max} = 0.663 \cdot 0.5\,\mathrm{W}$$

$$P_2 = 0.823 \cdot P_{max} = 0.823 \cdot 0.5\,\mathrm{W}$$

$$E = 0.823 \cdot 0.5 \cdot 2.2 \cdot tu + 0.663 \cdot 0.5 \cdot 2.2 \cdot tu + 0.663 \cdot 0.5 \cdot 80 \cdot tu$$

which leads to a total energy consumption of BIN of

$$E = 28.15 \cdot tu$$

Table 5.2 demonstrates the the measured results compared to the approximated energy consumptions in abstract numerical form as well as in relation, and highlights how little the difference between these values actually is. Therefore, the approximation equations developed in this thesis allow to compare the different algorithms very well, given that the relation of their energy consumptions is reproduced quite accurately.

	$\mathbf{E}_{measured}$ [%]	$\mathbf{E}_{approx.}$ [**W·tu**]	$\mathbf{E}_{approx.}$ [%]
CONST	100	41.1	100
FSS	65.53	27.25	66.30
BIN	68.13	28.15	68.49

Table 5.2: Comparison of measured and computed values

5.1.2 Searching for a Tag the Standard Approach Cannot Find - T_2

This section analyzes the behavior of all algorithms when searching for T_2. This tag was chosen, because it cannot be found by CONST, whereas all the optimized algorithms are very well able to communicate with it. Therefore, a scenario is presented where the standard approach of CONST simply fails.

CONST Algorithm

Figure 5.6 shows a comparison of the behavior of all algorithms in this case. As can be clearly seen in CONST's power profile, the field is set to the maximum value and one listing operation is executed right at the beginning. From the last section it is known that CONST reaches one tag T_9 with this single operation, but obviously no other one can be found. Therefore, CONST is forced to stop its execution. One big advantage of this strict behavior however is the fact that once the wanted tag cannot be found, the algorithm instantly stops, preventing any superfluous energy waste. As will be seen in the following sections, no other algorithm can be that quick in deciding the wanted tag is out of reach.

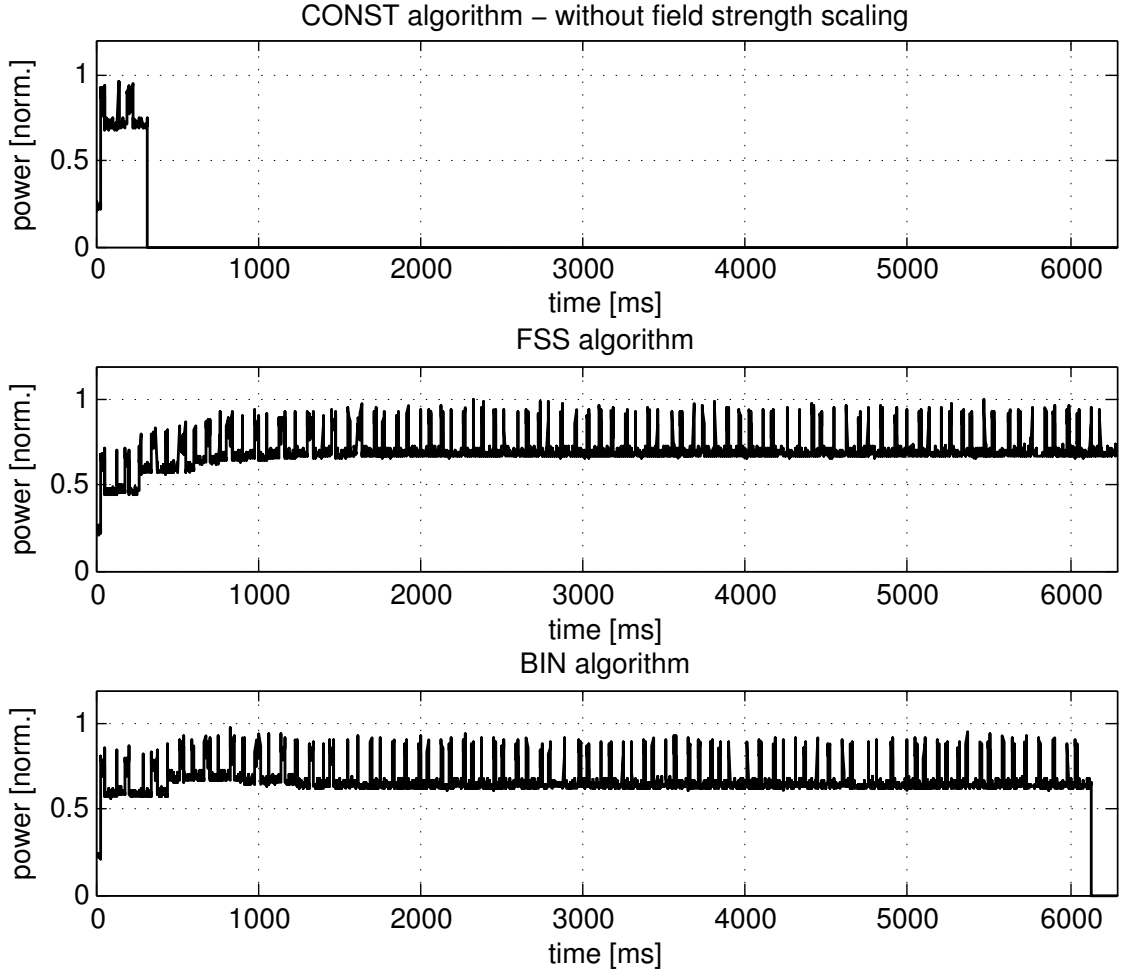


Figure 5.6: Power consumption of all algorithms when looking for tag T_2

FSS Algorithm

Figure 5.6 demonstrates that FSS cannot find the wanted tag T_2 as easily as it reached T_9 in the previous section. Quite a lot of steps have to be carried out during the discovery process. FSS has to scale up to shell 7 and execute just as many listing operations.

Therefore, in this scenario FSS wastes the longest time period of all algorithms during the discovery of the wanted tag. In each of the shells inspected by FSS a time of

$$t = t_{set} + t_{find} = 2.2 \cdot tu$$

is required in order to set the field once and to execute one listing operation. Additionally, when FSS reaches shell 7, it starts to read from the wanted tag, which requires the usual

$$t_{read} = 80 \cdot tu$$

Therefore, the total energy consumption can be computed as

$$E = \sum_{i=1}^{7} \Big(P_i \cdot 2.2 \cdot tu \Big) + P_7 \cdot 80 \cdot tu = 45.85 \cdot tu$$

BIN Algorithm

BIN, as shown in Figure 5.6, only requires four field setting and nine listing operations. While FSS has to read the wanted tag in shell 7, BIN due to its different discovery approach, is able to reach the same tag in shell 3. This scenario really highlights BIN's advantage over FSS in certain situations. It has to scale to fewer different shells, reaches and subsequently activates the tags in a totally different order and eventually realizes to read the wanted tag with a lot less field strength too. In its first step, BIN tries shell 2 and lists two tags without finding the wanted one,

$$P = P_2 = 0.823 \cdot P_{max}$$

$$t = t_{set} + t_{find} = 1 \cdot tu + 1.2 \cdot 2 \cdot tu = 3.4 \cdot tu$$

therefore, in the second step, BIN uses a higher field strength in shell 6 and discovers three tags, including the wanted one at the third position,

$$P = P_6 = 0.965 \cdot P_{max}$$

$$t = t_{set} + t_{find} = 1 \cdot tu + 1.2 \cdot 3 \cdot tu = 4.6 \cdot tu$$

in its third step, BIN tries shell 4 and discovers two tags, including the wanted one at the second position,

$$P = P_4 = 0.94 \cdot P_{max}$$

$$t = t_{set} + t_{find} = 1 \cdot tu + 1.2 \cdot 2 \cdot tu = 3.4 \cdot tu$$

BIN lowers the field strength, tries shell 3 and yet again discovers two tags, including the wanted one at the end,

$$P = P_3 = 0.91 \cdot P_{max}$$

$$t = t_{set} + t_{find} = 1 \cdot tu + 1.2 \cdot 2 \cdot tu = 3.4 \cdot tu$$

and finally the wanted tag is read in shell 3, with a power consumption of P_3 which takes the usual

$$t_{read} = 80 \cdot tu$$

The total energy consumption can be computed by adding all the partial equations which results in

$$E = 43.16 \cdot tu$$

Comparison of the Algorithms in the Case T_2

In Table 5.3 the associated total energy consumptions of this scenario are shown. A direct comparison with CONST of course would not make much sense, given that CONST quit its execution before even finding the wanted tag and therefore naturally consumes far less energy than all the other algorithms that were indeed able to read 1 KB of data. As a result, Table 5.3 only compares the energy consumptions of BIN and FSS. Once again it has been proven how accurate the approximation equations actually are when used to compare two algorithms in relation to each other.

	$\mathbf{E}_{measured}$ [%]	$\mathbf{E}_{approx.}$ [%]
FSS	100	100
BIN	93.96	94.13

Table 5.3: Comparison of measured and computed values

As described before, BIN despite its complex internal logic in this case is more efficient than FSS and has actually even finished its execution a bit earlier. This scenario however presents an additional and quite interesting fact. Much more tags can be reached by both optimized algorithms compared to the standard approach of CONST and therefore the possible transmission range is actually widened although the algorithms operate with lesser field. This scenario proves, that the field strength scaling principle, besides saving a lot of energy as was shown in the previous section, also enables more tags to be found with lesser field strength.

5.1.3 Searching for a Tag no Algorithm Can Find - T_6

Finally, this section covers a scenario where none of the algorithms developed for this thesis is able to find and communicate with the wanted tag. It is demonstrated that even optimized algorithms have certain limits, caused by the characteristics of the underlying NFC technology.

CONST Algorithm

Figure 5.7 shows a comparison of the behavior of all algorithms when looking for a tag they cannot find. CONST as always starts off by setting the field strength to the maximum amount possible, which is followed by one listing operation, that as mentioned in the previous sections, can only find tag T_9. Equivalent to what has been shown before, CONST stops its execution instantly. It has run for a very short time only, which of course saves a lot of energy. This section will demonstrate that CONST's behavior in this case is unmatched and by far the most efficient one energy wise.

FSS Algorithm

As demonstrated in Figure 5.7, FSS scales the field strength from shell 1 up to the last one shell 12. Although FSS is actually able to list new tags in most of the shells, the wanted one is never found. As can be seen in Figure 5.7, at the end of its execution FSS has reached the power consumption of CONST by scaling its field strength up to the maximum amount. This scenario shown here is the first time FSS is not the most efficient algorithm and even gets beaten by CONST, as it consumes ten times as much energy.

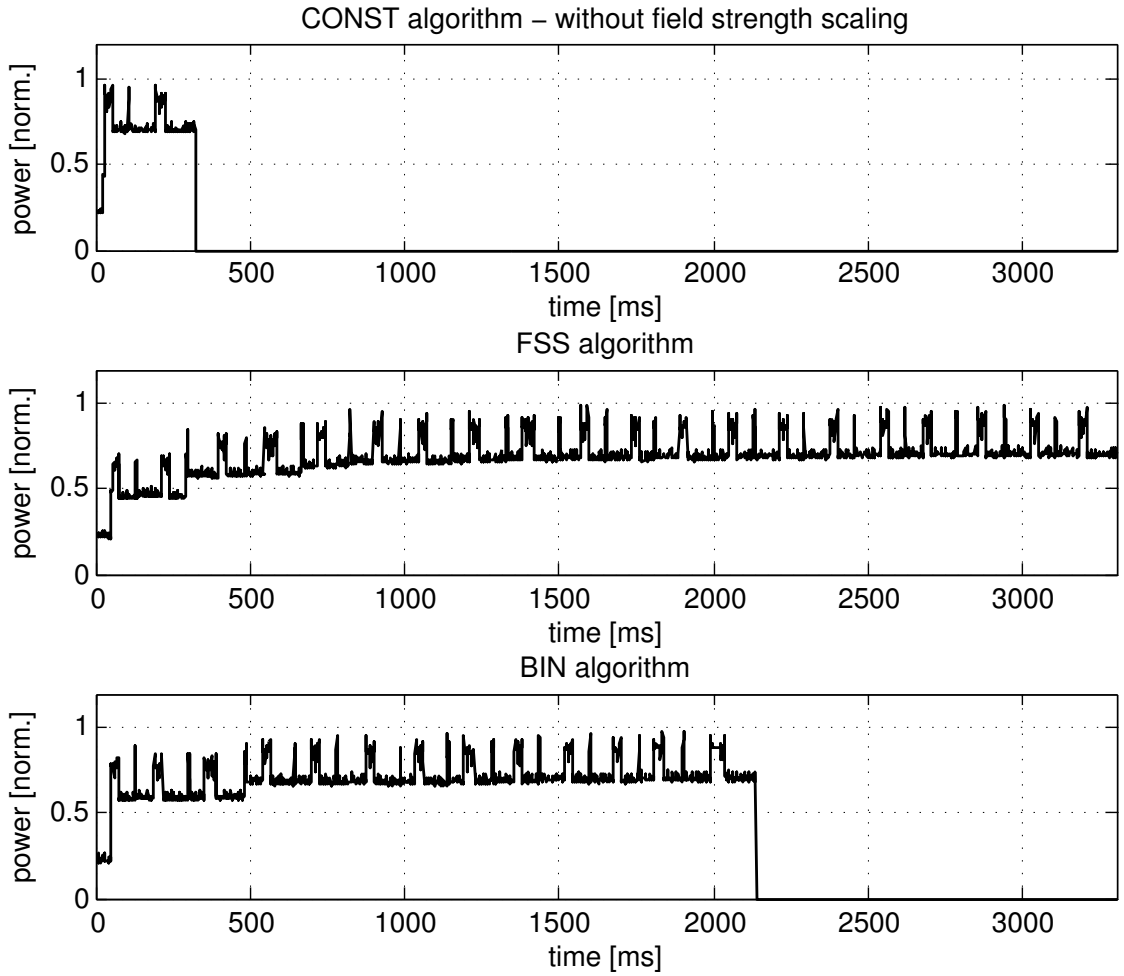


Figure 5.7: Power consumption of all algorithms when looking for tag T_6

BIN Algorithm

BIN is much more efficient than FSS in this scenario, although it still waste more energy than CONST. As shown in Figure 5.7, BIN requires five field strength setting operations and ten listing operations. At the end of its execution, BIN of course has reached the maximum field strength, but contrary to FSS less than half of the field setting operations were necessary and therefore it finishes its execution all lot faster.

Comparison of the Algorithms in the Case T_6

In Table 5.4 the associated total energy consumptions of this scenario are shown. A direct comparison with CONST is possible in this case, because all algorithms reach the same conclusion, that the wanted tag cannot be found. As discussed before, CONST by far is the most efficient one. This is the only case where CONST, even while using the full field strength, still consumes far less energy than any of the optimized algorithms. A difference of more than factor ten can be seen in Table 5.4. As discussed above, BIN is the second most efficient one in this case, followed by FSS. This scenario shows that the overhead introduced through field strength scaling operations and internal logic leads to a big disadvantages energy wise, whenever the subsequent 64 reading cycles cannot be executed. They of course represent the largest part of the execution time and executing them with less power consumption leads to a lot of energy savings. If however, as shown in this section, only the extensive tag discovery logic is compared to the simple one of CONST, the superfluous time and labor mainly dictates the energy consumption, regardless of how small the field strength and its power consumption could be kept.

	Energy [%]
CONST	100
FSS	1000.29
BIN	656.67

Table 5.4: Comparison of all algorithms when looking for T_6

5.2 Use Case 2 - 'NFC Wallet'

Various NFC applications are already in use today but the general spread of this new technology will especially increase over the next few years to come. One of the main sectors for NFC, as also discussed before in this thesis, is going to be mobile payment. Debit cards or bonus cards for specific shops of course are quite common nowadays, but they all lack an ease of use. Whenever customers want to pay for any goods, they need to search for the right card inside their wallets. Given that a lot of people buy articles in many different shops, this can be seen as a very time consuming and tedious thing to do. As will be shown in this section, NFC multi-tag support can remedy that shortcoming in an very elegant way.

The use case inspected in this section is called *NFC Wallet* as shown in Figure 5.8. It consists of five tags, being inserted in randomly chosen locations in a common wallet, which is then, as a whole, placed on the reader. The goal of course is to find a specific card inside that wallet without the need of interventions from the customer and to subsequently read 1 KB from it. Each shop uses a unique *store code* saved inside the first bytes, the remaining space could be filled with balance information and keys for encryption.

Figure 5.8: Setup for the use case 'NFC Wallet'

The last sections of this document have covered situations where great amounts of NFC tags in the near-field led to heavy detune and issues when trying to read from the tags, even though they were placed right on the surface of the reader. This section however tries to analyze a different aspect of multi-tag scenarios, the effects that occur whenever multiple tags, further away from each other but placed inside other objects like wallets, have to be read. Various tests and measurements have shown that up to three different NFC tags inside a common wallet, like the one shown in Figure 5.8, do not introduce any issues at all. Using a wallet with tags of three different shops that can be automatically charged at the cash-point however may not be enough in many cases. Therefore, the number of tags is raised to five and the associated results are presented in the next sections.

CONST Algorithm

CONST is able to reach four out of the five tags in this use case, as can be seen in the five power profiles shown in Figure 5.9, but this of course is still too little for any actual real life application. No customer would trust a system that only works in 80% of the times. As a result, the nowadays commonly used approach of a constant maximum field strength cannot be used to realize the *NFC Wallet* with five tags at all.

More generally speaking, one starts to wonder why this standard approach is still heavily used today, given that besides being very wasteful energy wise, in half of the scenarios discussed in this document up to this point it totally fails altogether.

FSS Algorithm

Figure 5.10 demonstrates the power profiles of FSS while searching for the tags T_1 to T_5 inside the wallet. It clearly shows the *NFC Wallet* use case is indeed realizable when using FSS instead of CONST. Every single tag can be found and FSS offers the overall most efficient way of all algorithms to achieve this. Table 5.5 demonstrates an overview of FSS' energy consumptions in this case.

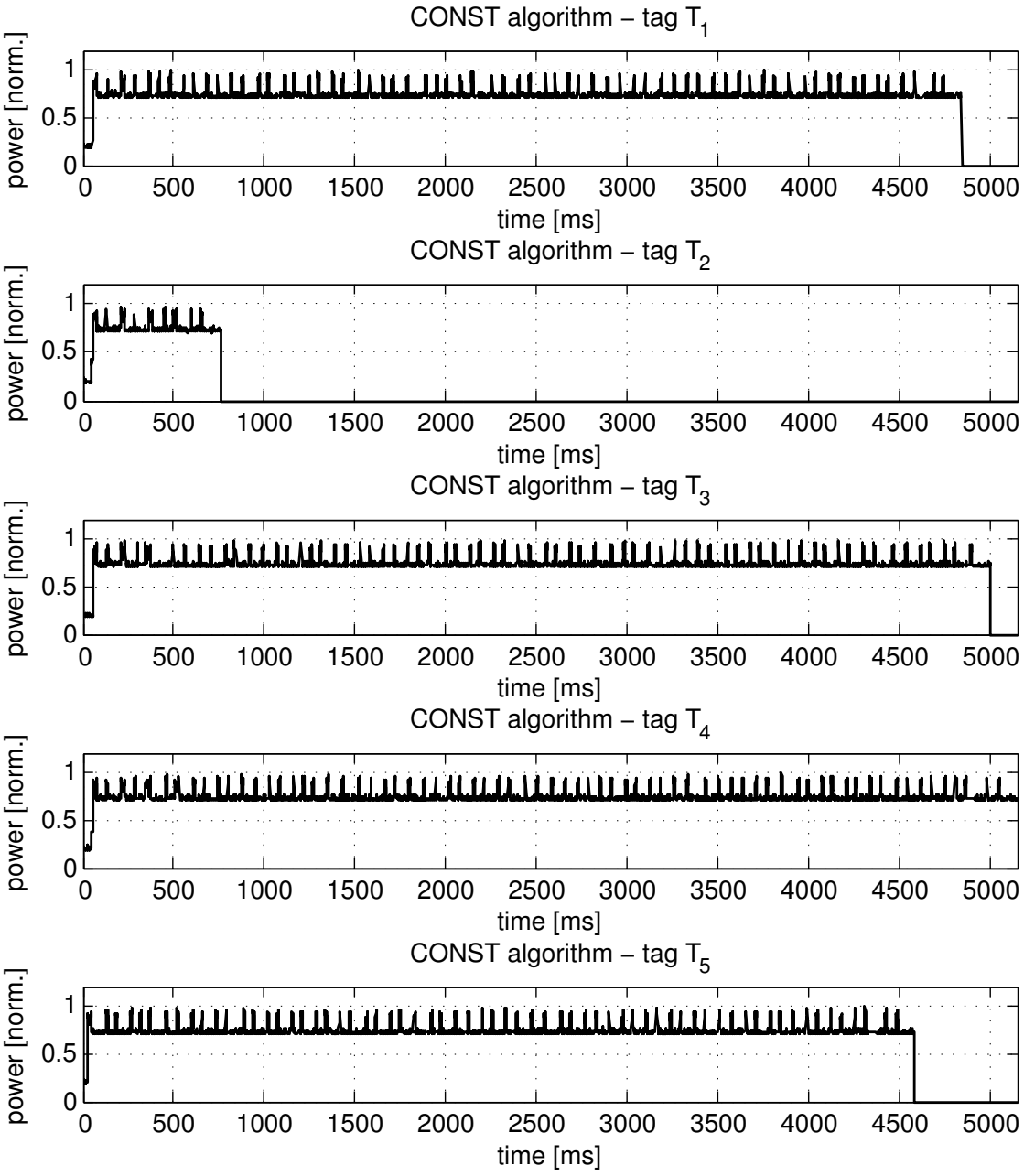


Figure 5.9: Power consumption of CONST when looking for all five tags inside the wallet

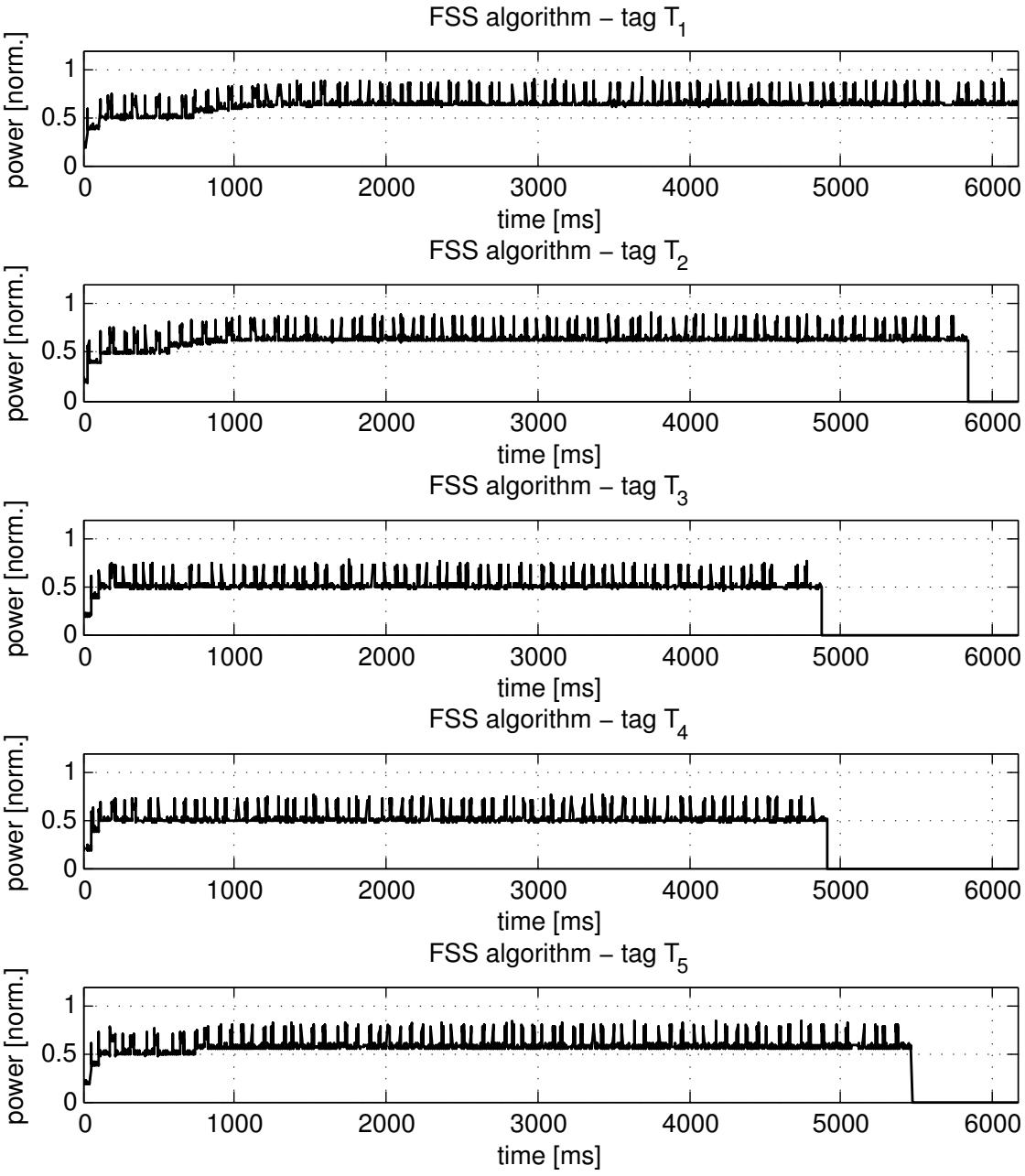


Figure 5.10: Power consumption of FSS when looking for all five tags inside the wallet

	Energy [%]
T_1	100
T_2	94.02
T_3	63.46
T_4	64.15
T_5	80.09

Table 5.5: Energy consumption of FSS when looking for all five tags inside the wallet

BIN algorithm

Figure 5.11 demonstrates that BIN is also able to find all five tags inside the *NFC Wallet*. Although BIN actually consumes more overall energy than FSS, it still offers one advantage. It exhibits very even power profiles, independently of the tag currently searched for. As can also be seen in Table 5.6, each tag can be found with almost the same effort and little variations in the energy consumptions occur. The average requirements an associated power supply of such an algorithm has to meet can therefore be predicted much easier.

	Energy [%]
T_1	97.14
T_2	99.64
T_3	96.79
T_4	100
T_5	91.79

Table 5.6: Energy consumption of BIN when looking for all five tags inside the wallet

The Delays of the Measurement Environment

For real life applications, one additional fact has to be discussed. Both optimized algorithms obviously take around six seconds in order to read 1 KB off the tag. These seemingly bad performances however are never actually caused by the algorithms themselves or the way they are implemented, but rather by the slow test environment in which the results presented here had to be measured. The power values discussed in the last few sections do indeed represent the actual requirements of all algorithms. The time and therefore the energy consumptions however are way too high to be representative of any

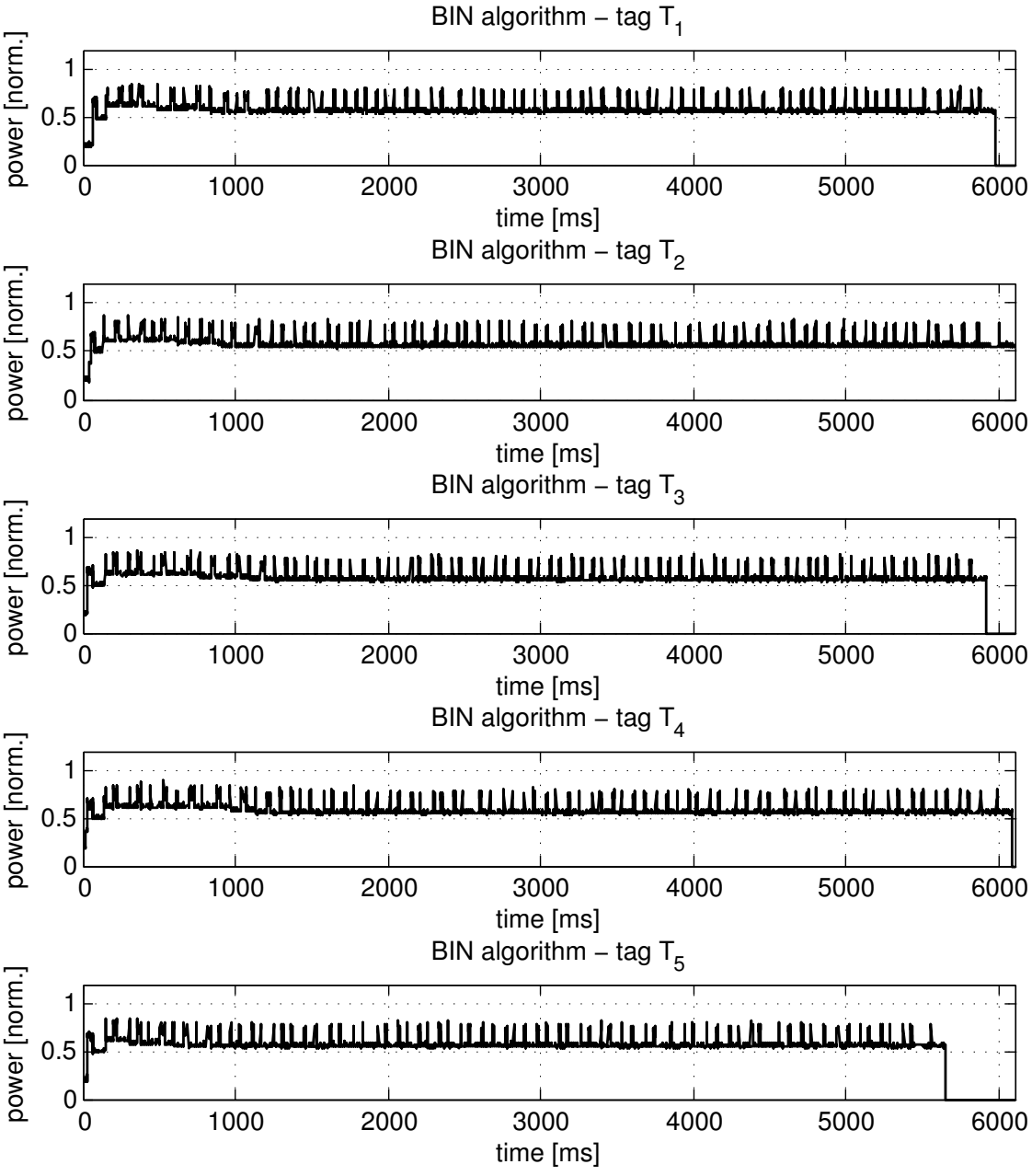


Figure 5.11: Power consumption of BIN when looking for all five tags inside the wallet

real life applications. Nevertheless they do serve their original purpose very well, allowing to compare all algorithms and to discuss advantages and issues. Therefore, energy consumptions have always been presented in relation to each other only, without the actual numerical values.

When directly executed on the target hardware without the delays introduced by the measurement components, all algorithms however can acquire the data of the wanted tag in at most one second, a time range short enough to be acceptable in every possible use case that may submerge in the next years, whereas delays of up to six seconds of course are not supportable in real life applications, especially when kept in mind that during these seconds, the actual location of the wanted tag should ideally not change at all. The long delays in this case very much demonstrate that the whole setup used for measurement, with all of its independent and remotely controlled components, is merely representative, concerning the time ranges of a real life application. The decreased execution speed is actually caused by three factors,

- a rather slow and old Windows XP operating system is being used on the measurement device,
- all the requests have to be sent to a web service, be loaded to an Android client, only to be forwarded yet again to the actual algorithms. This process of course takes much more time than simply executing the algorithms by themselves, as would be the case in a standalone application,
- Android's policy that any networking has to be done in background threads with lower priority and tedious overhead introduced by Android's tens of layers between software and the actual hardware that need to be passed.

Therefore, it has to be kept in mind that while the results shown here are indeed very useful for comparison, the actual algorithms perform far more efficient time-wise when used on a standalone device without all the measurement overhead.

5.3 The Three Algorithms in Single-Tag Scenarios

The last sections have demonstrated in detail how all algorithms perform in their intended environments. The question however remains how approaches specifically developed for multi-tag applications behave in more general situations, which of course may also include use cases that only require one single tag inside the communication range. Therefore, all measurements were repeated once more, with only one tag present inside the near-field this time.

The results do not surprise at all, the behavior does not change in any way and every algorithm developed for this thesis also successfully handles single tag use cases. This can be explained quite easily. It does not matter, whether only one tag can be reached because

- all other tags are too far away to be reached with the selected field strength,
- all other tags are detuned too much by mutual interference,
- there actually *is* only one tag present at the moment.

Therefore, if the algorithms can successfully handle one of these cases, all of them are supported. Nevertheless, additional versions of the algorithms were developed that drop the multi-tag functionality altogether. As shown in Figure 5.12, this does not affect the power profile at all. BIN in this case for example acts the same way as if multiple tags were present and only a single one could be reached.

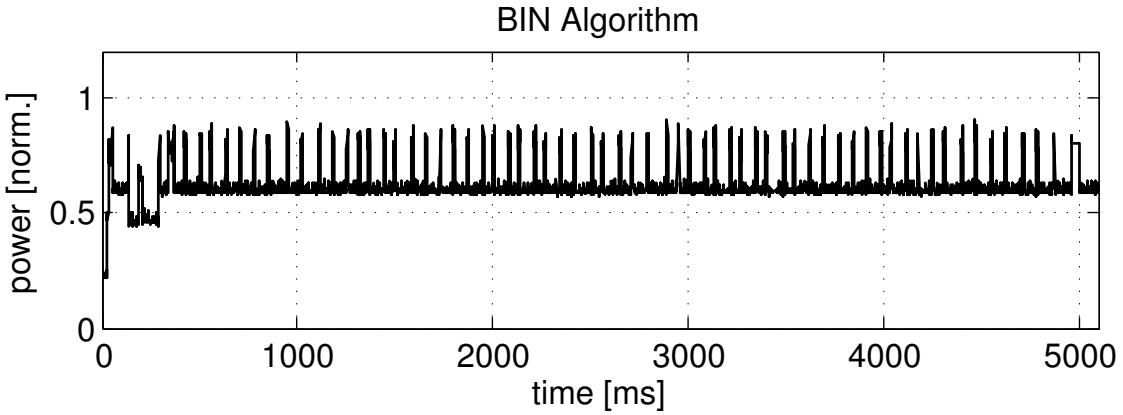


Figure 5.12: Power consumption of BIN without a multi-tag functionality

Even applications especially designed to only support single tag scenarios have to execute at least one listing operation to receive a full UID. This can clearly be seen in the example of Figure 5.12, where BIN executes one listing operation in shell 2, tries another one in shell 1 but cannot reach the single tag and therefore lists it again in shell 2. The exact same behavior would occur if BIN had included the multi-tag functionality as well.

The additional logic of the optimized algorithms does not aim at discovering a tag, but rather at finding the lowest possible field strength still sufficient to communicate with it. As a result, this power-aware approach is also applicable in single tag applications, as it is beneficial for saving energy, independently of number of tags actually present.

5.4 The Energy Consumption of a Complete Mobile System

The previous sections only discussed the potential energy savings of an *external NFC reader*, while omitting the power required by the *mobile device* it is connected to. The superior META[:SEC:] project however has already examined a contemporary mobile device with NFC capabilities, an Android *Nexus S* smart phone, and measured its power consumptions during two different operational states. On the one hand, while being in idle mode with NFC functionality turned off, and on the other hand, during an actual reading operation from a tag, executed by the smart phone's integrated NFC module. As shown in Figure 5.13, NFC operations lead to more than twice as much power being wasted by the smart phone, therefore definitely influencing the over-all energy consumptions of the entire system.

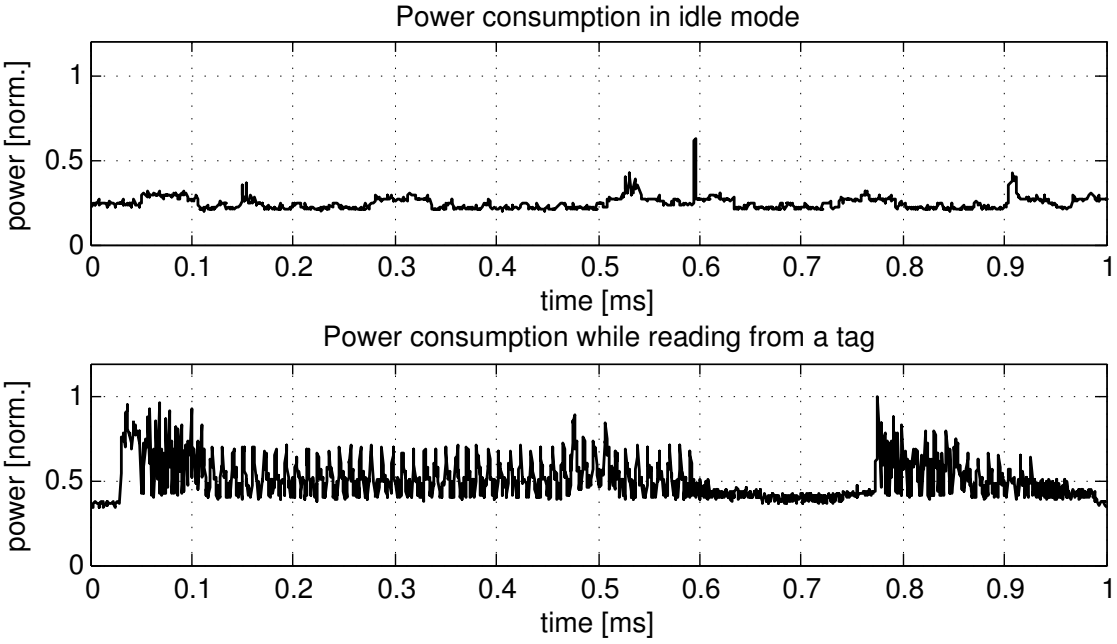


Figure 5.13: Power consumption of a smart phone

Figure 5.13 demonstrates that the amount of energy the two optimized algorithms FSS and BIN can actually save, by only improving NFC related operations, is indeed very relevant for a mobile system as a whole.

Chapter 6

Conclusion

NFC multi-tag field strength scaling scenarios offer a lot of unexplored potential for future use cases as well as great ideas for commercial products. Nevertheless, very little research has been carried out over the last few years and two basic issues were never really solved. Mutual interference of multiple tags inside a single field at a certain point renders several use cases impossible, while the high energy waste of systems being operated at full field strengths prevents multi-tag applications from being used energy efficiently on mobile devices.

This thesis aimed at finding solutions for both issues, and at the same time also realized an actual multi-tag support for the operating system Android. Given its widespread nature, a whole range of hand-held devices can benefit from this research. At first, a multi-tag functionality has been developed for Android that lists all reachable tags in sequence. However a communication with one specific tag with the least possible amount of field strength should be realized.

Two optimized algorithms that are based on the principle of field strength scaling were developed to handle that task efficiently and during the course of this thesis were compared with the standard approach of operating at full field strength. As opposed to a constant field, those algorithms scale the field strength during their search for the wanted tag in different ways. Detailed analyses inside this document have demonstrated the advantages as well as the problems of all algorithms and discussed how field strength scaling can be used as an elegant way to save a lot of valuable energy. It was shown that, dependent on the chosen algorithm, the actual energy consumption can be reduced by more than 34% and additionally scenarios where the commonly used approach of today totally fails to read data from the wanted tag, can be handled with the new algorithms. It was even demonstrated that paradoxically more tags can be found when operating with less field strength.

Various use cases were measured that on the one hand set out to explore the limits of the optimized algorithms, but on the other hand also introduced interesting new ideas to realize real life applications benefiting from the great advantages the algorithms offer. It was shown how the payment process of tomorrow can be simplified with the help of NFC and how issues that always occur when using the standard approach can simply be by-passed with the newly developed methods.

All functionality was successfully implemented in the widespread operating system of today's mobile devices, Android, therefore allowing as many people as possible to actually

benefit from it. The software's flexible design however even allows to switch the platform dependent layer and as a result use the optimized algorithms in any system environment desired in future works.

Accurate equations were developed, based upon the insights gained during the whole course of this thesis, which describe the behavior of a NFC reader in multi-tag scenarios. Predictions of energy consumptions are possible even without any access to the actual hardware used in this thesis, which allows new algorithms to be tested beforehand and different approaches to be compared in relation to each other. This thesis has

- demonstrated the great potential of multi-tag field strength scaling scenarios,
- shown how to save valuable energy without losing any functionality in many different use cases,
- allowed ten times as much tags to be discovered with far less field strength,
- enabled a real life payment use case, otherwise impossible with existing approaches,
- realized a direct support of all those features in the popular Android system,
- and even introduced accurate equations to predict the energy consumptions of the NFC reader used in this thesis.

6.1 Future Work

Power-Aware Computing

This thesis has discussed at length one possible approach at saving energy, known as field strength scaling, and based on this fundamental idea has developed two different algorithms that actually use the principle in multi-tag scenarios. Therefore, future work can be based on the insights gained in this document, and even single tag applications, as shown in this document, can benefit form the newly developed optimized algorithms.

'NFC Wallet' Use Case

Given that the field of NFC multi-tag scenarios offers so much unexplored potential, future work will definitely discover many new and amazing applications for everyday use. Especially ideas related to the use case *NFC wallet*, discussed in this document, provide great potential for every day applications.

Field Strength Scaling Algorithms

The optimized algorithms developed for this thesis have clearly shown how energy can be saved without compromising any of the original functionality. Future projects can extend these algorithms to adapt them to any new requirements, or even introduce additional and new algorithms altogether that may offer optimized performance for very specific situations. In their current state the algorithms only consider the least amount of field strength sufficient to detect and read from a tag. If however additional operations like cryptographic computations that consume more power may be necessary as well, issues

could occur. Therefore, future projects might want to enhance the algorithms to cover that aspect as well.

When combined with the Android operating system, common smart phones can also be used as portable NFC readers. Thanks to the new and power-aware algorithms of this thesis, these devices can operate for far longer time periods and successfully handle multiple tags at once. As soon as mobile payments with NFC or similar use cases become omnipresent, the great advantages of the algorithms will become even more apparent.

Energy equations

Besides the actual implementations this thesis also demonstrated a lot of new theoretical insights that can be used in future projects in this field. Especially the equations developed in this document can help to achieve quick and easy energy predictions and allow making important choices beforehand.

Bibliography

[CH07] V. Chawla and Dong Sam Ha. An overview of passive rfid. *Communications Magazine, IEEE*, 45(9):pages 11–17, September 2007.

[CKK+07] Jung-Hyun Cho, Jikon Kim, Jae-Whan Kim, Kyungil Lee, Kwang-Duk Aim, and Shiho Kim. An nfc transceiver with rf-powered rfid transponder mode. In *Solid-State Circuits Conference, 2007. ASSCC '07. IEEE Asian*, pages 172–175, November 2007.

[DMS+12] N. Druml, M. Menghin, C. Steger, R. Weiss, A. Genser, H. Bock, and Haid J. Adaptive field strength scaling - a power optimization technique for contactless reader / smart card systems. In *Euromicro Conference on Digital System Design*, pages 616–623, September 2012.

[ECM08] ECMA. Near field communication interface and protocol (nfcip-1) third edition. Technical report, ECMA International and International Standards Organization, 2008.

[Fin10] Klaus Finkenzeller. *Fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication.* Wiley, Chichester, West Sussex Hoboken, NJ, 2010.

[GZ11] Liu Guoyu and Wan Zhenkai. Summarize of rfid technology and typical application. In *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2011*, volume 2, pages 1032–1036, July 2011.

[IEA11] International Energy Agency OECD IEA. *World energy outlook 2011.* International Energy Agency OECD, Paris, 2011.

[Ins10] Texas Instruments. Sitara am37x evaluation module (evm) quick start guide. Technical report, Texas Instruments, 2010.

[Kip12] Bernhard Kipperer. Remote control interface for android devices with the use case safetycard. Technical report, Technical University of Graz, 2012.

[Lan05] J. Landt. The history of rfid. *IEEE Potentials*, 24(4):pages 8–11, October-November 2005.

[LMD+09] Bin Lin, A. Mallik, P. Dinda, G. Memik, and R. Dick. User- and process-driven dynamic voltage and frequency scaling. In *Performance Analysis of*

Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on, pages 11–22, April 2009.

[LRL10] Izabela Lacmanovic, Biljana Radulovic, and Dejan Lacmanovic. Contactless payment systems based on rfid technology. In *MIPRO, 2010 Proceedings of the 33rd International Convention*, pages 1114–1119, May 2010.

[MDS+12] M. Menghin, N. Druml, C. Steger, R. Weiss, H. Bock, and J. Haid. The ptf-determinator: A run-time method used to save energy in nfc-systems. In *Fourth International EURASIP Workshop on RFID Technology*, pages 92–98, 2012.

[MLKS08] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. Nfc devices: Security and privacy. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 642–647, March 2008.

[MSW08] Stephen B. Miles, Sanjay E. Sarma, and John R. Williams. *RFID technology and applications*. Cambridge University Press, Cambridge, UK New York, 2008.

[NXP07a] NXP. Nfc-fri software development kit 1.0. Technical report, NXP, 2007.

[NXP07b] NXP. Nfc is the double-click in the internet of the things. *RFID Systems and Technologies (RFID SysTech), 2007 3rd European Workshop on*, pages 1–5, June 2007.

[NXP07c] NXP. Pn532 user manual. Technical report, NXP Semiconductors, 2007.

[NXP08] NXP. Functional specification mifare ultralight. Technical report, NXP, 2008.

[NXP11] NXP. Pn532/c1 near field communication (nfc) controller. Technical report, NXP, 2011.

[PaGAP10] J.C. Paille ands, C. Gaber, V. Alimi, and M. Pasquet. Payment and privacy: A key for the development of nfc mobile. In *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*, pages 378–385, May 2010.

[PFW11] G.P. Perrucci, F.H.P. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1–6, May 2011.

[QC10] Xianming Qing and Zhi Ning Chen. Uhf near-field rfid antennas. In *Antenna Technology (iWAT), 2010 International Workshop on*, pages 1–4, March 2010.

[Res10] Florian Resatsch. *Ubiquitous Computing Developing and Evaluating Near Field Communication Applications*. Gabler Verlag / Springer Fachmedien Wiesbaden, Wiesbaden, Wiesbaden, 2010.

[SS03] M.R. Stan and K. Skadron. Power-aware computing. *Computer*, 36(12):pages 35–38, December 2003.

[Wan11] R. Want. Near field communication. *Pervasive Computing, IEEE*, 10(3):pages 4–7, July-September 2011.

[WW08] Harald Witschnig and Walter Winkler. *Characterisation of Detuning Effects in Multilabel Scenarios*. 2008.

[XGW+11] Xunteng Xu, Lin Gu, Jianping Wang, Guoliang Xing, and Shing-Chi Cheung. Read more with less: An adaptive approach to energy-efficient rfid systems. *IEEE J JSAC*, 29(8):pages 1684–1697, 2011.

[ZBSF05] Bo Zhai, D. Blaauw, D. Sylvester, and K. Flautner. The limit of dynamic voltage scaling and insomniac dynamic voltage scaling. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(11):pages 1239–1252, November 2005.