Masterarbeit

Measurement Based Evaluation Framework for Near Field Communication Systems

Gernot Johann Solic, BSc

Institut für Technische Informatik
Technische Universität Graz
Vorstand: Univ.-Prof. Dipl.-Inform. Dr. techn. Kay Römer

Begutachter:	Ass.-Prof. Dipl.-Ing. Dr. techn. Christian Steger
Betreuer:	Ass.-Prof. Dipl.-Ing. Dr. techn. Christian Steger
	Dipl.-Ing. Dr. techn. Manuel Menghin, BSc

Graz, im August 2014

Kurzfassung

Im Laufe der letzten Jahre hat sich NFC am Markt immer stärker durchgesetzt und hat sich auf mobilen Endgeräten als ein weiterer Kommunikationsstandard etabliert. Ein aktuelles Thema im Bereich von Datenerfassung und Fernsteuerung sind NFC-Bridges, die gewöhnliche Elektronikbauteile um eine NFC-Kommunikationsschnittstelle erweitern können und es somit ermöglichen, Smartphones als Ersatz für Benutzeroberflächen einzusetzen. Auf mobilen Endgeräten ist die Energieversorgung limitiert – daher sollten die integrierten Komponenten optimiert werden um eine möglichst lange Betriebszeit zu ermöglichen. Es müssen daher diese NFC-Systeme auf einen möglichst geringen Energieverbrauch optimiert werden. In der Praxis werden Werkzeuge zur Evaluierung des Energieverbrauches häufig an die zu entwickelnden NFC-Applikationen angepasst. In dieser Masterarbeit wird ein generisches Grundkonzept für NFC-Systeme diskutiert und wie anhand dieses Entwicklungswerkzeuges der Energieverbrauch gemessen und anschließend optimiert werden kann.

Das in dieser Masterarbeit konzipierte NFC-System dient als Grundlage für den Datentausch zwischen den kommunizierenden NFC-Komponenten. In der Implementation des Evaluierungssystems wird ein Smartphone mit NFC-Erweiterung als Benutzerschnittstelle und ein Einplatinencomputer mit NFC-Gateway Erweiterung als NFC-Bridge eingesetzt. Diese zwei Komponenten dienen als Basis für die Ausführungsplattform der zu evaluierenden Anwendungen. Anschließend wird eine vorhandene Messumgebung an die Ausführungsplattform angeschlossen, die zur Messung des Energieverbrauches dient. Nach der Messung werden die erfassten Daten ausgewertet und mit Hilfe dieser Datenanalyse können Optimierungen für den Energieverbrauch an der Applikation vorgenommen werden welche die Laufzeit des mobilen Endgerätes erhöhen. Die Stärken dieser Evaluierungsumgebung liegen im generischen Aufbau, in der Reduzierung des Entwicklungsaufwandes sowie der progressiven Optimierung der Applikation.

Zusätzlich zur Implementierung des Evaluierungssystems werden auch drei konkrete Anwendungsfälle auf diesem NFC-System evaluiert und mittels einfachen Softwareoptimierungen verbessert. Diese Anwendungsfälle sollen den Umgang mit dem messbasierten Evaluierungssystem erläutern und zeigen wie Applikationen optimiert werden können. Der Vergleich der erfassten Messdaten von den einzelnen Anwendungsfällen soll beweisen, dass dieses generische Konzept eines Evaluierungssystems für NFC-Brücken als Grundgerüst für die Entwicklung von Applikationen erfolgreich herangezogen werden kann.

Abstract

In the last few years NFC has become a mature communication standard for mobile devices and is suitable for a broad variety of application areas. A hot topic in terms of remote control and data acquisition are NFC bridges which are able to extend common electronic devices with a NFC interface. Furthermore, NFC enabled smartphones can be utilized as a replacement of the built-in user-interface for these electronic devices. NFC applications are different and therefore the evaluation of the power consumption has to be adapted to individual use-cases in order to increase the operating time of the system.

This thesis will introduce a generic measurement based evaluation framework for NFC systems which can be used as underlaying development and optimization platform. The evaluation framework is the foundation for the communication between the NFC components and is executed in a real hardware environment. The implementation uses a state-of-the art NFC enabled smartphone as the user-interface. Furthermore, a single-board computer with a NFC gateway extension as the desired NFC bridge is providing the run-time environment for the use-case applications. The analysis of the power consumption is done with an existing evaluation suite which will be appended to the run-time environment for measurement purpose. The gathered power consumption data is used to apply optimizations to the evaluated application and increase the operating time of the mobile device. The main advantages for such a evaluation framework is the generic architecture, the improvement of the development cycle and continuous progressive optimizations.

In addition to the evaluation framework, three use-case applications are evaluated to expose the current implementation and optimization potential with simple software techniques. These use-case applications demonstrate the functionality of the framework and are utilized in order to improve application development in terms of power consumption. Furthermore, the comparison of the collected measurement data from the unoptimized and optimized application shows how successful the evaluation framework works and serves as the proof of concept.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz am, ...

(Unterschrift)

Englische Fassung:

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)

Danksagung

Diese Diplomarbeit wurde im Studienjahr 2013/2014 am Institut für Technische Informatik im Rahmen des META[:SEC:] Projekts an der Technischen Universität Graz durchgeführt.

Danksagung an alle am Institut, die mich während der Recherche und Erarbeitung dieser Arbeit unterstützt und motiviert haben. Ein besonders ausdrücklicher Dank gilt speziell Dipl.-Ing. Manuel Menghin, der mir über die gesamte Dauer des Projektes mit wertvollen Tipps, Anregungen sowie Verbesserungsvorschlägen beistand und mich bei der Realisierung dieser Arbeit stets gefördert hat. Bedanken möchte ich mich auch bei Herrn Ass. Prof. Dipl.-Ing. Dr. techn. Christian Steger, der mich während der Anfertigung meiner Masterarbeit begleitet hat.

Herzlich bedanke ich mich außerdem bei meinen Eltern, die mir meine Ausbildung erst ermöglichten und mir immer eine wertvolle moralische Hilfe sind. Ein weiterer Dank gilt zusätzlich meiner Schwester und meinen Verwandten für die Unterstützung.

Bedanken möchte ich mich auch besonders bei meiner Freundin, die mir jederzeit mit vollstem Verständnis zur Seite stand. Desweiteren bedanke ich mich bei meinen langjährigen Freunden aus meinem Heimatort und auch bei den neu gewonnenen Freunden während des Studiums für die angenehme Zusammenarbeit und die spannende Zeit.

Herzlichen Dank für Eure Unterstützung!

Graz, 28. August 2014 Gernot Johann Solic, BSc

Contents

List of Figures

List of Tables

Chapter 1

Introduction

During the last years, radio frequency identification (RFID) technology expanded quickly and new fields of applications have been a hot topic of research for various technical domains. A relatively young specification architectures is near field communication (NFC) which is characterized by a short transmission range and secure communication between the two involved devices. NFC was designed to be backwards compatible to existing RFID infrastructures and should discover new application areas that are requiring the characterizing properties of this specification. NFC was integrated in different market sectors like payment environments or access authorization systems with a highly security relevant focus. The breakthrough that made NFC aware in general public was the integration of this communication technology in the new generation of mobile devices. The high market penetration of smartphones anticipate plenty new use-case scenarios for NFC systems. A current issue topic in terms of connecting mobile gadgets with common electronic devices are NFC bridges which are extending existing hardware with NFC interfaces. Thus common devices can be extended with an input unit (smartphone) that has a highly user-friendly design at a low-cost because no additional extensions are required beside the bridge hardware. [1] mentions three unique advantages of NFC enabled smartphones integrated into NFC system environments:

- *Interactivity*: User interface for intuitive client input options and visualizations.
- *Multi-application management*: Possibility of multiple application execution in real-time and remotely with help of the smartphones mobile network functionalities.
- *Remote User Management*: Users are able to authenticate over the NFC interface and can receive personal or customized information.

The main advantage is that NFC systems in combination with a sophisticated user interface have a valuable potential for innovative use-cases currently and in the near future. A limiting constraint of these innovative ecosystems is the power consumption of the NFC interfaces especially for smartphones as mobile devices are supplied by a battery and hence everything has to be power optimized.

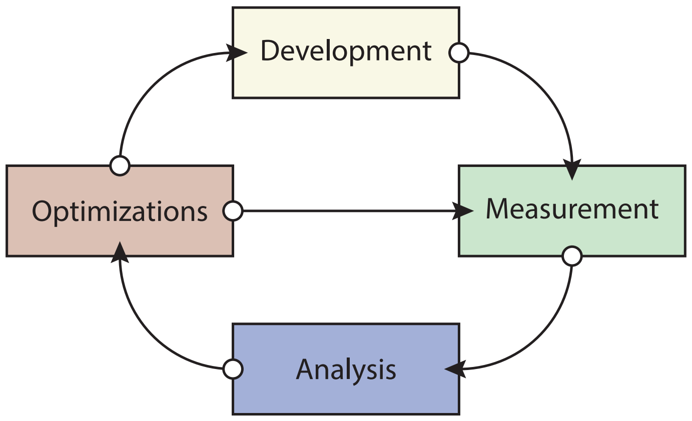


Figure 1.1: Evaluation and optimization process

1.1 Motivation

The limited power supply of the mobile device (smartphone) requires optimization techniques for the NFC connection in terms of power consumption. But how can researchers and developers determine the parts of the system with the highest impact to the power drain in order to increase time of operation? New development tools are required which can be utilized to discover power critical regions and are adaptable for a wide range of possible use-case applications. These tools gather the required information by measuring the power consumption on real hardware during application execution. The collected measurement data is analyzed afterwards and provides feedback for potential improvement adaptations (optimizations) to the present system. Thus the quality of NFC applications based on NFC bridges can be enhanced iteratively with continuous measurement. Figure 1.1 illustrates the optimization process through frequent measurement iterations. This concept is specifying four process tasks:

- *Development*: Implementation and realization of the main functionality and system logic in terms of application context.
- *Measurement*: Application execution and parallel power consumption measurement with convenient instruments.
- *Analysis*: Evaluation of measurement results and detection of optimization possibilities of the application.
- *Optimizations*: Improve the application with suitable power optimization techniques and methods.

Currently, evaluation tools are already existing but they are designed to measure specific applications rather than novel NFC system architectures. Thus the concept of a generic measurement framework can be utilized to evaluate and optimize various different use-case scenarios. Furthermore, this generic approach of the NFC framework could be the underlaying foundation for the general development of a NFC bridge architecture.

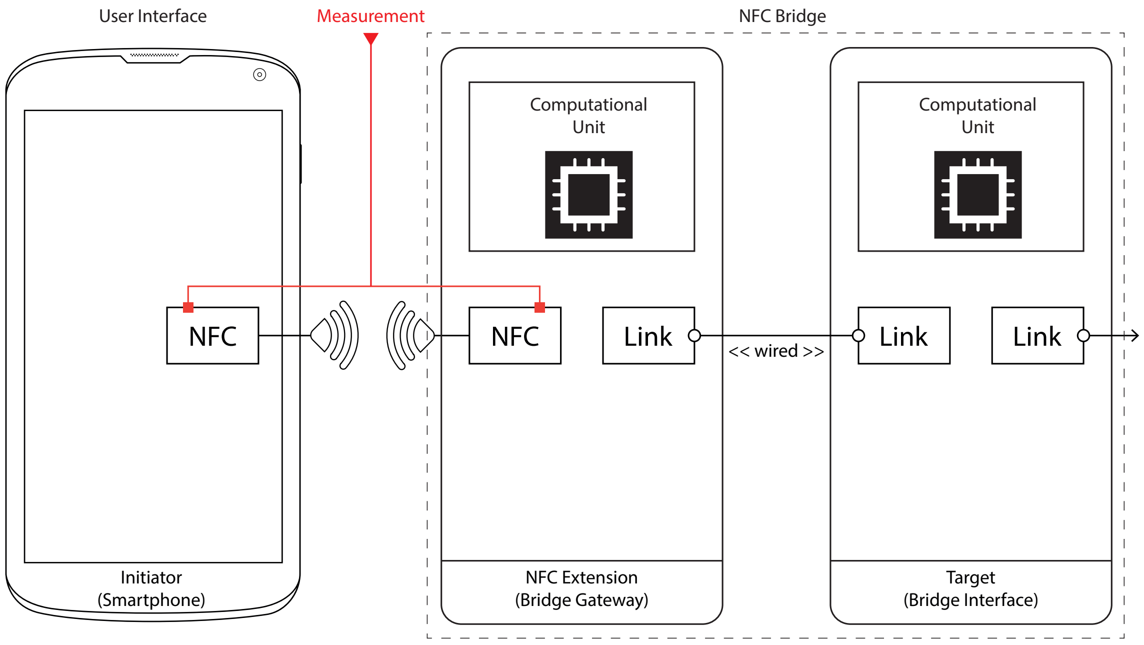


Figure 1.2: NFC bridge based on start-of-the-art hardware

1.2 Contribution and Goals

This master thesis is part of the META[:SEC:] project which is conducted by the Institute of Technical Informatics [1] on TUGraz. In this project, a measurement based evaluation framework for NFC bridges is designed and implemented. The main object of this thesis is the realization of an generic evaluation environment for researchers and developers which execute and measure various different use-case applications. The benefit is the classification of the required modules for such an environment and how this concept can be implemented with state-of-the-art hardware. Currently, NFC bridges consist of an initiator which is establishing the connection and a target which is responding to the requests from the initiator. This concept for NFC bridges is the central component which will be extended to an evaluation framework. The initiator and target should provide the run-time environment for the use-case application which will be evaluated on the system. A measurement suite is appended to the core execution framework and collects the power consumption of the devices. The concept of an evaluation system is illustrated in figure 1.2 in which the initiator consists of a user-interface (smartphone) and the target is a primitive computational unit (e.g. micro-controller) with a gateway as NFC extension. Furthermore, the target and the gateway extension can be integrated as NFC bridge in common devices (e.g. consumer electronics) with few effort and at a low-cost.

A concrete implementation with real hardware is developed during the practical work of this thesis. The main implementation part is the development of the run-time environment. An existing measurement component will be appended to the run-time environment. The

[1]`https://www.iti.tugraz.at/`

combination of the measurement component and the run-time environment is the targeted evaluation framework. Further, this basic system can be utilized to measure and optimize NFC use-case applications and serves as a template for more enhanced measurement based evaluation frameworks. A case study should demonstrate that the concept of such an environment is working and can be used to optimize NFC applications and services. In this case study three use-case scenarios are implemented on top of the evaluation framework and analyzed afterwards. This serves as the proof of concept for the implementation approach of the designed system. The following enumeration lists the general tasks for the research of this thesis and the practical implementation:

1. Analyze existing NFC evaluation systems and related work
2. Create a concept and design of the measurement based evaluation framework
3. Implementing of the run-time environment
4. Integration of the existing measurement suite from META[:SEC:] into the evaluation framework
5. Perform case-study analysis with use-case application examples as proof of concept

1.3 Content Structure

This thesis is structured in several different topics. This description gives a brief overview about all chapters and sections that are included in this work. From the design of the very first basic abstraction until the final evaluation framework with a prove of concept. Afterwards, results will be presented and upcoming tasks will be defined in the future work chapter.

Related Work 2 This chapter gives an overview of techniques, methodologies and concepts that are related to this thesis and have influenced the idea and realization of this work.

Design 3 This part discusses the classified components and composition for the targeted evaluation framework and also identifies the necessary requirements of the system.

Implementation 4 This chapter explains the creation and development process of the current implementation and describes how the hardware and software works.

Case Study 5 This part demonstrates the measurement extension and shows evaluation examples with different use-case applications.

Conclusion 6 This chapter gives a summary of the findings during the research of this thesis and exposes advantages and disadvantages of the system.

Future Work 7 This part describes possible enhancements for the current implementation which are researchable in future projects.

Chapter 2

Related Work

Related work describes reference systems for NFC environments and gives an overview of how to measure such systems and improve them with different strategies. They illustrate a generic architecture that is the anchor for further designs and concepts. In the sections power consumption and evaluation it is discussed how to automate the testing process in the system and how to use collected data in order to optimize the system and lower the power drain from the battery. Thereafter, the evaluation of hardware devices for the system is argued, this is the classification of possible devices that would fit in the environment. The last section in *Related Work* covers the possible operation systems that will be used for the basic practical implementation and the integration in the planned run-time environment.

2.1 Radio Frequency Identification

RFID is a wireless communication technology for data and energy transmission. The major idea behind RFID is the electrical or magnetic field which establishes the connection for the transfer. The NFC specification relies on a high frequency magnetic field, thus the NFC components of devices are inductively coupled systems. These systems are similar to the electrical transformer principle. The active device initiates a magnetic field which is used as the communication channel to the second device. [2, pg. 13ff]

2.2 Near Field Communication

A new specification of the RFID technology is near field communication (NFC) and very common for the current device generations. It represents a young set of communication standards compared to other personal area networks (PAN) systems (e.g. Bluetooth). NFC covers the radio frequency ISM band of 13.56 MHz and it is a short range (~10 cm) communication technology. The three operation modes for NFC are *peer-to-peer*, *reader-writer* and *card-emulation* mode. The major advantages for NFC over other wireless

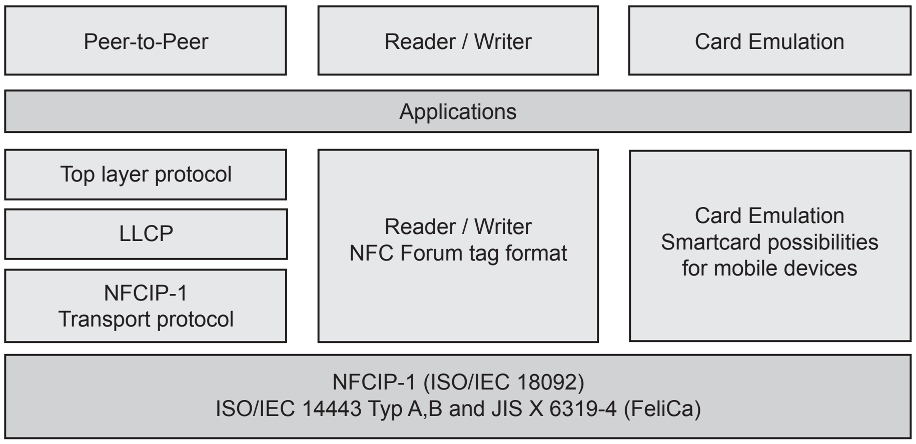


Figure 2.1: NFC forum architecture [2, pg. 90]

communication technologies are the energy transfer from the active to the passive NFC device and its pairing-less establishment of the connection. NFC is a combination as well as extension of the proprietary RFID systems *NXP MIFARE* (ISO/IEC 14443 Typ A) and *Sony FeliCa* and it is downward compatible to existing RFID systems based on these standards. [2, pg. 87]

2.2.1 Standards

There are several engineering standards for the NFC specification, the most important ones are *NFCIP-1* (ISO/IEC 18092) and *NFCIP-2* (ISO/IEC 21481). Table 2.1 gives an overview of the ECMA[1] NFC standards. [2, pg. 87]

Standard	Description
ECMA-340	Near Field Communication Interface and Protocol (NFCIP-1)
ECMA-352	Near Field Communication Interface and Protocol-2 (NFCIP-2)
ECMA-356	NFCIP-1 RF Interface Test methods
ECMA-362	NFCIP-1 Protocol Test Methods
ECMA-373	Near Field Communication Wired Interface (NFC-WI)
ECMA-385	NFC-SEC: NFCIP-1 Security Services and Protocol
ECMA-386	NFC-SEC-01: NFC-SEC Cryptography Standard using ECDH and AES

Table 2.1: Engineering standards for NFC defined by ECMA International [2, pg. 88]

The NFC forum is an association of organizations and companies with strong interest to push the NFC technology to the market. This forum is responsible for the definition of NFC specifications, compliance with the standards and information for customers about the application possibilities. The forum created a complete NFC architecture based on the engineering standards from ECMA International, see figure 2.1. [2, pg. 88]

[1]European Computer Manufacturers Association

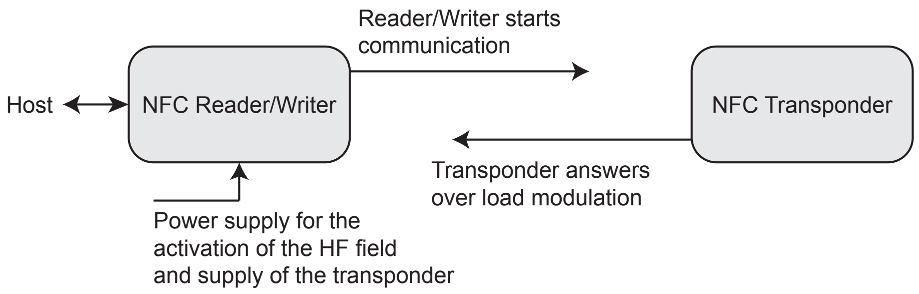

Reader/Writer starts communication
Host
NFC Reader/Writer
NFC Transponder
Transponder answers over load modulation
Power supply for the activation of the HF field and supply of the transponder

Figure 2.2: Reader-Writer mode [2, pg. 92]

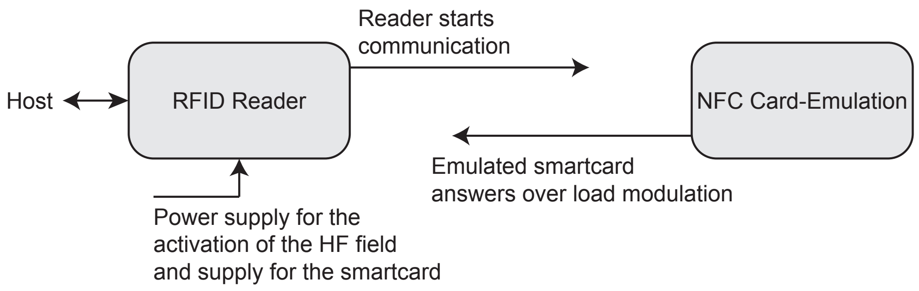

Reader starts communication
Host
RFID Reader
NFC Card-Emulation
Emulated smartcard answers over load modulation
Power supply for the activation of the HF field and supply for the smartcard

Figure 2.3: Card-Emulation Mode [2, pg. 100]

2.2.2 Operation modes

The three standard operation modes for NFC systems are reader-writer mode, card-emulation mode and per-to-peer mode. In this thesis reader-writer and card-emulation mode will be focused, peer-to-peer mode can be added in future projects if necessary.

Reader-Writer Mode

The reader-writer mode provides communication with passive RFID transponders (see figure 2.2). Every NFC enabled device must support the specified NFC forum tags. Based on NFCIP-1, the passive communication component notifies the active part if the peer-to-peer mode is supported, otherwise the reader-writer mode is used. The reader-write mode is the backward compatibility mode to existing RFID environments. [2, pg. 99]

Card-Emulation Mode

The card-emulation mode provides the interaction with RFID readers (see figure 2.3) and is backward compatible to smartcard environments. The NFC device emulates a contact-less smartcard like FeliCa, Mifare or even application protocol data unit (APDU) driven cards. NFC hardware provides a *secure element* that is similar to a real smartcard and communicates over the wireless NFC interface. A *secure element* is independent and operates autonomous even if the other hardware of the device is deactivated. [2, pg. 100]

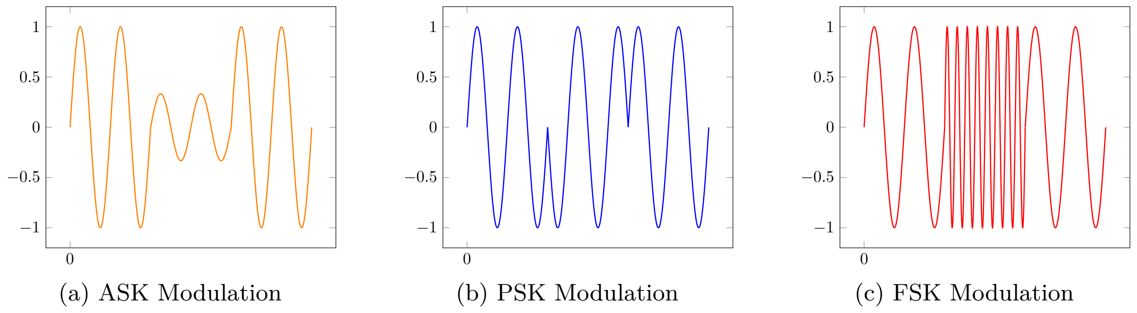


Figure 2.4: Modulation types for RFID with $n = 2$ symbols

The proposed system will be implemented in reader-writer mode with support for custom APDU operations (card-emulation mode). The target (transponder emulation) will be compatible to NFC forum type 1-3 tag specification [3, 4, 5] and can also be extended with customized APDU commands which are not part of the specification. A secure element is not planned but can be simply added to the software implementation of the target emulation.

2.2.3 Modulation

The modulation binds digital information to the emitted magnetic field in order to transport the data between NFC devices. There a three different modulation types for RFID technology: amplitude shift keying (ASK), phase shift keying (PSK) and frequency shift keying (FSK). For *NfcA* systems 100% ASK (initiator) and the on-off keying (OOK) (target) are the relevant modulation types. As the name suggests ASK encodes the data into the amplitude of the signal-wave. For binary signals 2-ASK is very common because this type only uses two states (0,1). Figure 2.4 illustrates these three different modulation types.

2.2.4 Coding

The coding in combination with the modulation is the layer for data transportation. Two types of coding are used in terms of *NfcA*: modified miller-coding on the initiators side and manchester-coding on the targets side. Figure 2.5 shows the information data and the signal for the specific coding.

2.2.5 Data and Command Flow

Data flow is based on a request-response sequence and actions are encapsulated into APDU packets which are transceived between the initiator and target. The initiator (active part) prepares request command and sends it to the target (passive or emulated transponder) which is responding accordingly to the request. The validation of transmitted data is

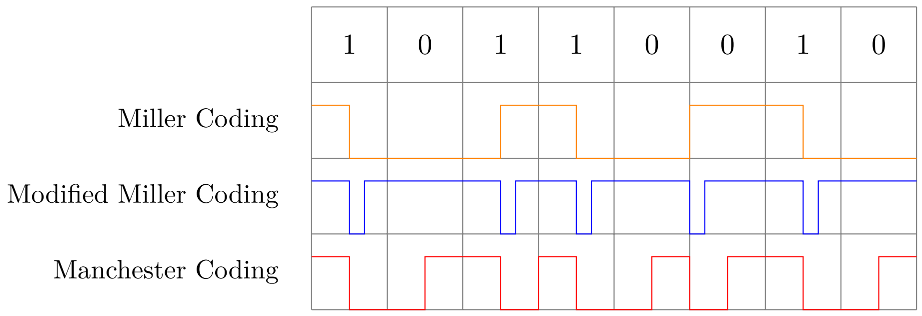


Figure 2.5: Coding signals [2, pg. 24]

performed with cyclic redundancy check (CRC) in particular CRC-16 (ISO/IEC 13239 standard) which is based on the polynomial equation $P(x) = x^{16} + x^{15} + x^2 + 1$ [6, pg. 1].

Depending on the operation mode there is an initialization sequence for target determination, activation and selection. After this initialization phase application dependent commands can be exchanged. For the *NfcA* technology which is supported and implemented in the proposed system the initialization phase is described in [7] and is executed with the following tasks:

1. Request targets (`ALL_REQ` and `SENS_REQ`)
2. Anticollision phase (NFCID1) (`SDD_REQ`)
3. Target selection (`SEL_REQ`)
4. Application requests (`READ`, `WRITE`, ...)
5. Target deselection (listen mode → sleep mode) (`SLP_REQ`)

2.3 Ubiquitous Systems for NFC

There are various prototypes for controlling and monitoring sensor devices over a wireless data link. [8] explains three NFC enabled sensors types.

1. Passive sensors
2. User controlled semi-passive sensors
3. Standalone semi-passive sensors for long-term monitoring

Suitable for this project are stand-alone semi-passive sensors for long-term monitoring which are activated by user or environment events. [8] describes such systems in detail. A possible realization of a stand-alone semi-passive sensor can be a simple temperature transmitter. The sensor is activated with a trigger in periodic intervals and records the measured temperature to the internal storage. The user can connect the smartphone with the sensor with help of the NFC interface and is able to evaluate the stored temperature values. The user is able to control the target device in order to activate the sensor and gets

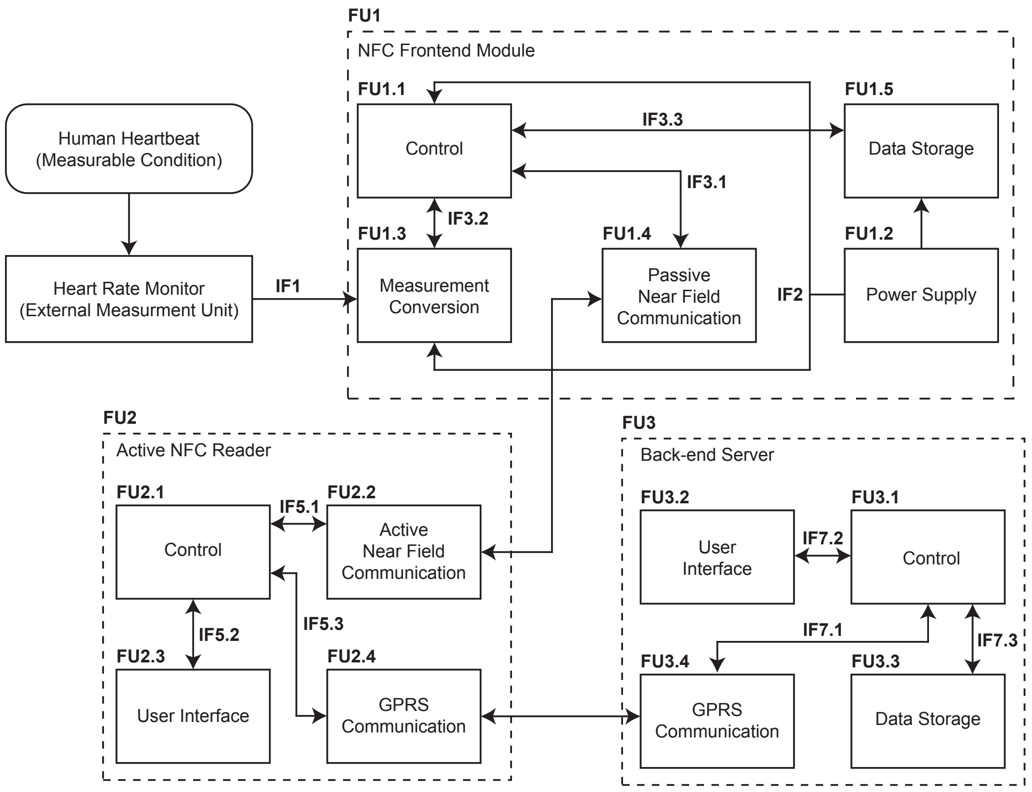


Figure 2.6: Block diagram of the remote monitoring system [9, pg. 74]

the current temperature directly. This approach is straight forward and a solid foundation for the targeted evaluation framework.

2.3.1 Monitoring System

A more detailed view on stand-alone semi-passive sensors is described by Opperman and Hancke [9, pg. 46]. The designed system measures the human heartbeat with an external measurement sensor. This sensor is connected to a passive NFC front-end module with an internal storage system. The active NFC reader is a smartphone and supplies the front-end module. The reader also establishes a general packet radio service (GPRS) connection to a back-end server where the heartbeat data is collected and stored. The created system is shown in figure 2.6.

The functional unit (FU) 1 is operating as the NFC gateway and any feasible sensor equipment is attachable to this device. Communication is established over a NFC connection to the FU 2. FU 1 is a passive component so it is activated when the FU 2 (smartphone) with its active NFC reader is in close proximity. The FU 2 in this configuration acts as a repeater. It receives data from the FU 1 and transmits the information over long range to the back-end server (FU 3) with help of GPRS technology. FU 3 has obtained control over the measurement equipment with a two-way wireless connection and is able to interact with FU 1. The detailed description and results for this example prototype is discussed in [9, pg. 46].

2.3.2 Wireless Sensor

Chan et al. [10, pg. 751] designed a wireless RFID power meter with an outage recording module, see figure 2.7. The central component is a RFID tag which provides access to the power meter with an outage recording module and the rear-end processing system. A data acquisition module (DAM) is used to collect and memorize the recorded data for voltage and current to the internal storage unit. This module is connected with the RFID reader via RS323[2] which can transmit the stored data to the RFID tag. The outage recording module operates the same way. The rear-end processing system is implemented as a personal digital assistant (PDA) RFID reader. The PDA can be used to extract the stored data periodically from the RFID tag and analyze it for further processing.

The two following caveats need to be observed in this particular prototype.

- **Concurrency:** It is important to manage the access times from the RFID tag. A flag can help to determine the access state and the reader which is just performing an operation on the tag. This simple solution keeps the system in a stable state. The flag solution will not scale well for larger and more complex systems with more sensor modules attached.
- **Memory Size:** The RFID tag has a limited memory size. The power meter prototype does not require a large storage capacity because the RFID tag is capable to store 64 pages each with 4 Bytes. The discussed system only needs 34 of the storage pages. On systems with a higher data throughput this is a bottle-neck for the RFID tag solution.

2.4 Power Optimization Techniques for NFC

Power consumption is an important issue on ubiquitous mobile-devices because available battery capacity is limited without connection to an electric supply network. Wireless data transmission drains a significant amount of power from the supply. Thus power saving principles must be applied to mobile-devices. Power optimization is a huge topic in general and therefore common approaches for NFC will be pointed out and explained in this thesis. NFC power-awareness is a key topic because in a typical environment there is an active component (*initiator*) and a passive component (*target*). The initiator energizes the target with the emitted magnetic field. Stand-by optimization, power management and appropriate hardware design are universal techniques for lowering the power drain of mobile-devices.

[2]serial communication interface

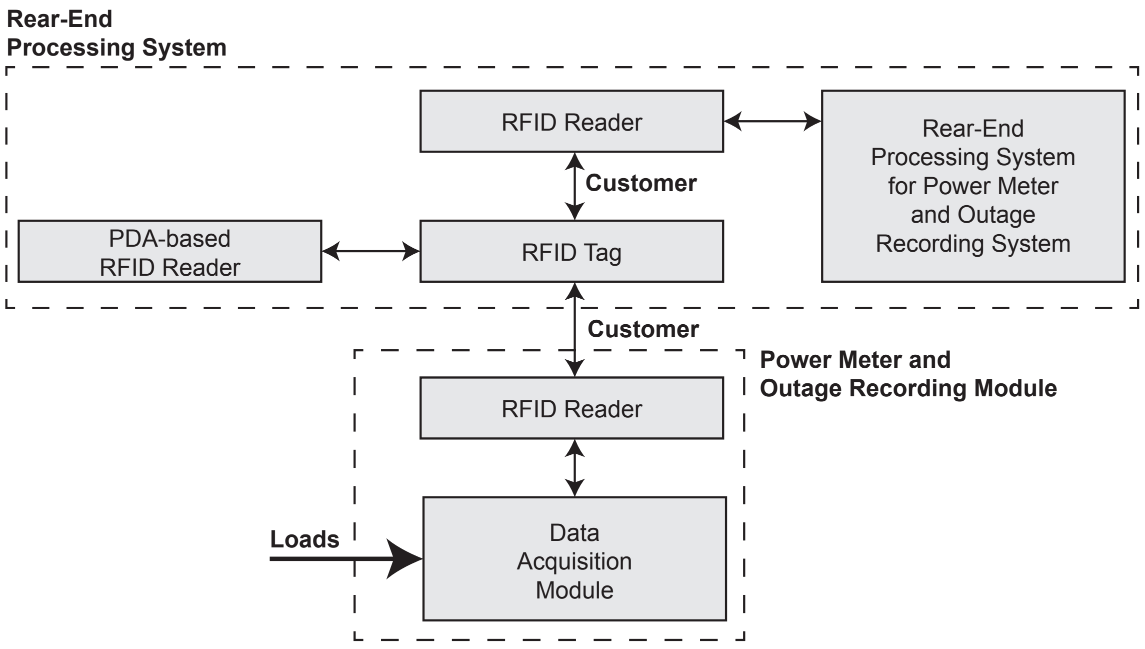

Rear-End Processing System
RFID Reader
Rear-End Processing System for Power Meter and Outage Recording System
Customer
PDA-based RFID Reader
RFID Tag
Customer
Power Meter and Outage Recording Module
RFID Reader
Loads
Data Acquisition Module

Figure 2.7: Simplified system architecture of the power meter [10, pg. 751]

2.4.1 Power Drain Reduction with Stand-by Mechanisms

Meier [11, pg. 1] remarks that a significant amount of power is wasted during idle time and power savings can be accomplished with state-of-the-art stand-by mechanisms. It is necessary to enable the magnetic field of the initiator on demand when there is data ready to be send. On Android operating system (OS) the magnetic field is switched on with the display of the mobile device. Therefore, user activity is required to initialize NFC communication. Another approach is periodically enabling the field and the determination of a responding transponder. Finding the best timings for the activation of the field is a complex task and is different for each application and based on the performed operations.

Another option is the use of the initiator's magnetic field to enable the targets internal power supply. Druml et al. [12, pg. 2] introduce an innovative near field communication interface enabling zero energy (NIZE). This methodology implements exactly this target activation with the initiators magnetic field. The analog front-end of the reader and target device behave like an over-the-air activation switch for the targets power supply, see figure 2.8. The reader establishes the magnetic field and the target collects the power and triggers the internal supply. After all enqueued communication messages are finished the target device turns off the internal power supply. This solution guarantees a lossless stand-by target. Devices which are waiting for user interaction can be extended simply and cost-efficiently.

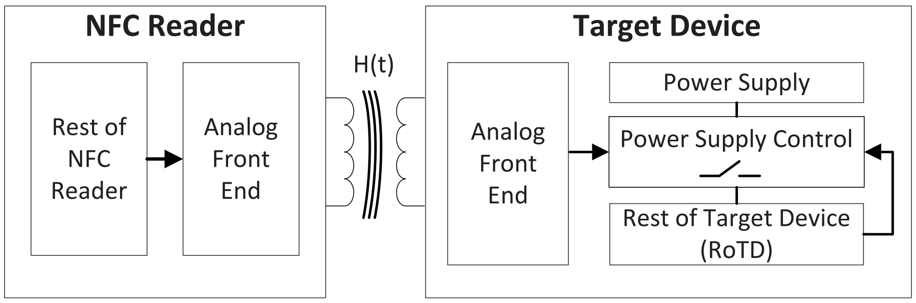


Figure 2.8: System architecture of NIZE [12, pg. 2]

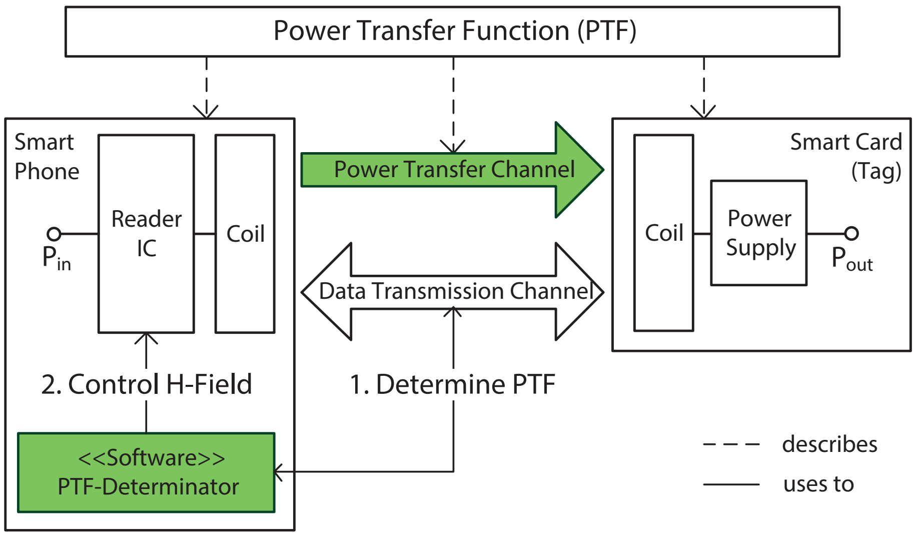


Figure 2.9: PTF-Determinator in an NFC environment [13, pg. 1]

2.4.2 Power Management

Another approach for power saving is efficient energy management. This methodology describes dynamically power scaling. Thus the system is analyzing the power consumption in every single state and based on the gathered information the power drain will be reduced to a minimum where the systems still operates correctly.

Menghin et al. [13, pg. 1] proposes the PTF-Determinator which is some kind of such a dynamically mechanism. During run-time of the system the PTF-Determinator identifies the power transfer function (PTF) and scales the consumed power to a sufficient minimum. The PTF-Determinator is adapted for a NFC environment and determines power consumption during run-time for the tag-detection. The acquired information is used to enhance the timings for the initialization phase. Menghin et al. mentions an average of 12% power saving for the PTF-Determinator.

The automatic power stepping (APS) is a similar procedure for power management. Xu et al. [14, pg. 1] mention that APS scales the transmission power level with the amount of used tags in the magnetic field. Extensive empirical studies has been performed on

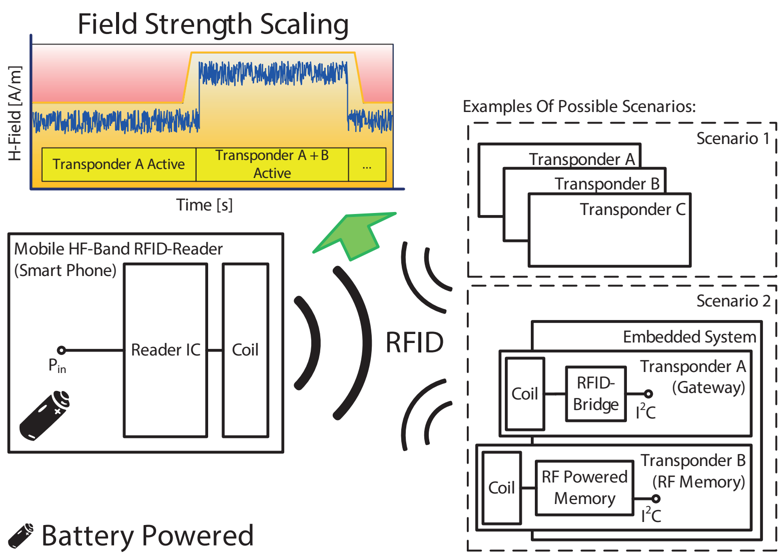


Figure 2.10: Adaptive field strength in a multiple-transponder system [16, pg. 1]

passive tags to establish an accurate APS which reduces the power consumption up to about 60%. This method is basically applicable for RFID systems and compatible with existing standards. It is not necessary to modify transponders for an APS extension.

Other counter-measure techniques focus on the magnetic field like adaptive field strength scaling (AFSS). Typically, many systems are operating on the maximum field strength but this is not necessary for some applications. Thus a better strategy would be an adjustment of the magnetic field to the transponders needs. The basic concept of AFSS is to increase or decrease the field strength accordingly to the smartcard power consumption [15]. There are two different approaches for this field scaling technique:

- **Request-based AFSS:** Software implementation of the scaling. For example, the smartcard exchanges the power consumption information with the reader (*initiator*) which adjusts the strength to a proper value.
- **Instantaneous power consumption-based AFSS:** Hardware modifications for the involved components.

Field strength can also be dynamically scaled for multiple-transponder systems thus the field strength is increased with the amount of transponders. Figure 2.10 shows a multi-transponder system and how the reader (*initiator*) behaves when new transponders are entering the proximity range. Energy reduction for AFSS is on average up to 54% and for multiple-transponder implementations it is about 34%. Therefore, the field strength scaling is a powerful optimization method [16].

2.4.3 Physical Parameter Analysis

Physical parameters have considerable impact on the power drain. For instance, in RFID and NFC systems the design and efficiency of antennas have a major influence on the operating time of different devices. A potential high power drain is avoidable with an evaluation of the systems specific physical parameters. For wireless environments the most common parameters are antenna design, distance between antennas and field strength.

Warnick et al. [17, pg. 1] developed a relationship between near field power transfer and radiation efficiency for dipole-type antennas in order to identify the optimal geometrical parameters for the desired antenna. This method can either be used for communication purpose or for power transfer between systems.

2.4.4 Use-Case Dependency

A different approach for power consumption reduction is the direct evaluation of the selected application. Previous discussed strategies are usually generic methods which can be applied to a broad range of systems. The following ideas and concepts are heavily depending on the use-case and have to be adapted to the actual scenario. This thesis is generally based on the mentioned methodologies below in order to test different kinds of NFC environments and reduce the power consumption with software techniques.

Compression and Packet Size

Data compression and accordingly packet size adjustment also increase battery supply time. Both types of optimization strongly depend on the system environment and have to be implemented correctly.

Data Compression: With data compression the communication transfers faster because less data is send and the data-channel can be closed earlier. In relation to NFC this means the magnetic field can be switched-off faster which results in a lower power drain. In general, there are two categories of data compression: lossless and lossy-compression. As their names suggest lossless compression transports the exact digital information from the sender to the receiver. The sender encodes data with the selected algorithm and the receiver decodes the data, so that the original data from the sender and the receivers decoded data are identical. With lossy-compression the information is lost between sender and receiver, but for the receiver the lost information is not important to recompose the essential data. A lot of efficient lossy and lossless compression algorithms exist. The proper selection of the right scheme depends on the information that is transceived. Another factor for choosing the appropriate compression method is the packaging ratio and in relation to that the encoding and decoding performance which increases execution time. Finding the right technique for the application can be researched in [18].

Message Size: The adjustment of the message size can also give a huge improvement in terms of power consumption. A simple approach could be clustering of a request-series. Therefore, it is possible to construct one cumulative packet for multiple requests. A trivial example for NFC systems: The initiator performs a custom command which requests a series of single Byte values from the target. For simplification, initialization and authentication are skipped. The initiator requests 10 values in row from the target. This results in $10 * 1 + 10 * 2 = 30 Bytes$ (2 Bytes for CRC16) which are responded in 10 messages. Clustering will reduce the size by $18 Bytes$ $(10 * 1 + 2 = 12 Bytes)$. So there is about 60% CRC16 overhead without clustering in this particular example which can be avoided with a sophisticated clustering algorithm. This approach will not work for chained APDU responses because they are already fully allocated.

Overhead Reduction

In terms of power optimization the elimination of specific application overhead could help to lower the power output. Depending on the system and the application context some generic solutions are:

- Skipping of request and selection commands
- No exchange of authentication data for none security-critical systems
- Reduce validation to a minimum (CRC)

Caching

A convenient caching strategy has a positive impact on the systems power consumption. The general question is the determination when data transfer is required and how long the data is valid. The developer has to keep in mind when information is outdated and needs to be refreshed. It is not necessary continuously pulling the actual data over the communication interface (NFC). Data can be stored in a buffer and is valid for a adjusted time period. On new data requests the buffer value will be returned as the current value. After the time-out the value can be requested again from the communication interface and is stored as buffer for the next time interval. Figure 2.11 illustrates a basic approach for a caching strategy in a NFC environment. The application is repeatedly pooling for a single value from the reader. The first request will be responded by the reader and the value will be stored in the cache. All following responses until the time-out are coming from the caches value buffer.

2.5 Power Evaluation Strategies

The last chapter demonstrate methods for power consumption optimization in energy-aware communication systems. The next step is to measure and capture power consumption in

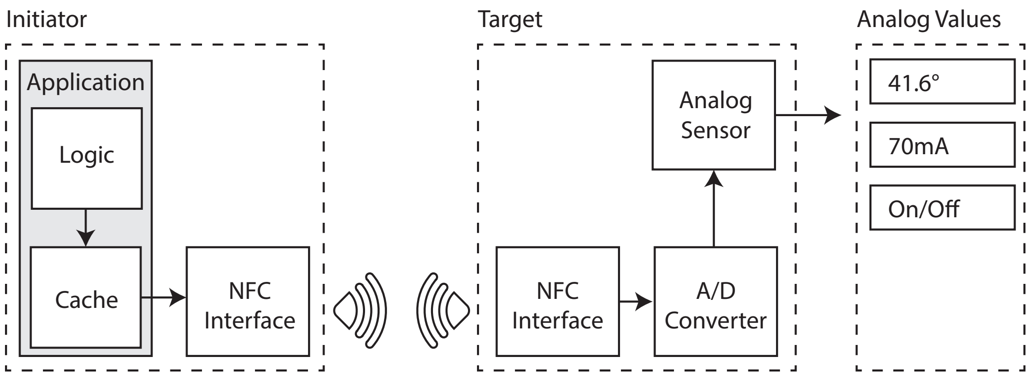


Figure 2.11: Caching strategy for primitive values in an NFC environment

order to find energy-critical implementation sections and improve them. Power evaluation closes the gap between classification of energy consumption and optimization techniques. Some challenging tasks in terms of power evaluation are fast and flexible adaptation of the system under test and also the separation of measurement suite and the system environment itself therefore the capturing systems should act like an observer.

2.5.1 Automation

Automation is the general foundation for power evaluation systems and thus it is necessary to implement automated behavior. Applications change over time and have to be tested repeatedly. With a highly automated environment it is possible to generate measurement data on demand while reducing testing-time. A concept is illustrated in figure 2.12 in which a controller manages measurement elements and stores the collected data for further use. Assigning measured data to application logic is essential for a conclusion. Therefore, the controller also observes the state of the executed applications. The controller is fully configurable and parameterizable to fulfill the requirements. Gathered results can either be evaluated by the controller unit or externally. A similar hardware in the loop approach is discussed in [19].

2.5.2 Hardware in the Loop

A hardware in the loop (HIL) as discussed in [20, 19] is a common and established process which facilitates the development and optimization tasks of hardware devices that are part of the systems environment. Main benefit of a HIL approach is testing various collaborating hardware components inclusively the program-logic in a surrounding simulation. In this thesis the targeted measurement framework will act as a HIL simulation where it is possible to replace hardware devices and software components effortless. The whole simulation performs in real-time and provides a perspective how the system behaves in real environments. This strategy is suitable for a wide range of technical systems like automotive industry, robotics and power engineering. Architectures and designs which can

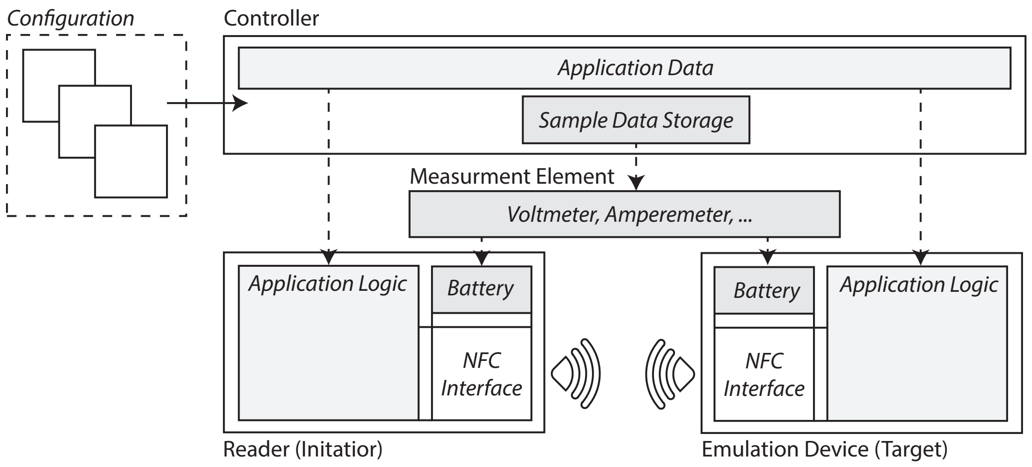


Figure 2.12: Controller automation for power evaluation systems

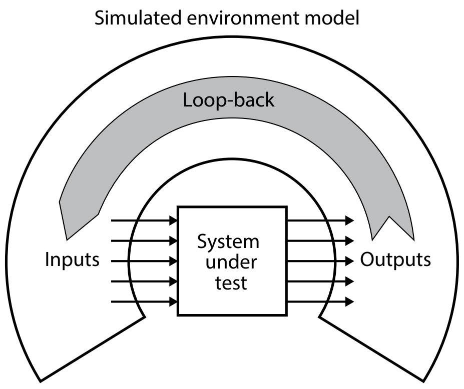


Figure 2.13: In the loop test method [25, pg. 2]

be applied to various systems are explained in [21, 22, 23, 24].

Another concept is in-the-loop testing for software components. Ryssel, Ploennigs, and Kabitzsch [25, pg. 2] mentions a system under test (SUT) in combination with a simulation model for in-the-loop testing, see figure 1. Furthermore, there are four in-the-loop procedures for the SUT:

- **Model in the loop (MIL)**: Creation of a model that describes the component under test. The model can be used to test whether the requirements are fulfilled.
- **Software in the loop (SIL)**: Software implementation (any kind of language is possible) or generation from the model.
- **Processor in the loop (PIL)**: Test of the previously developed implementation on the target processor. The underlying system architecture is irrelevant.
- **Hardware in the loop (HIL)**: Deployment of the software component to the target device itself with the predefined requirements.

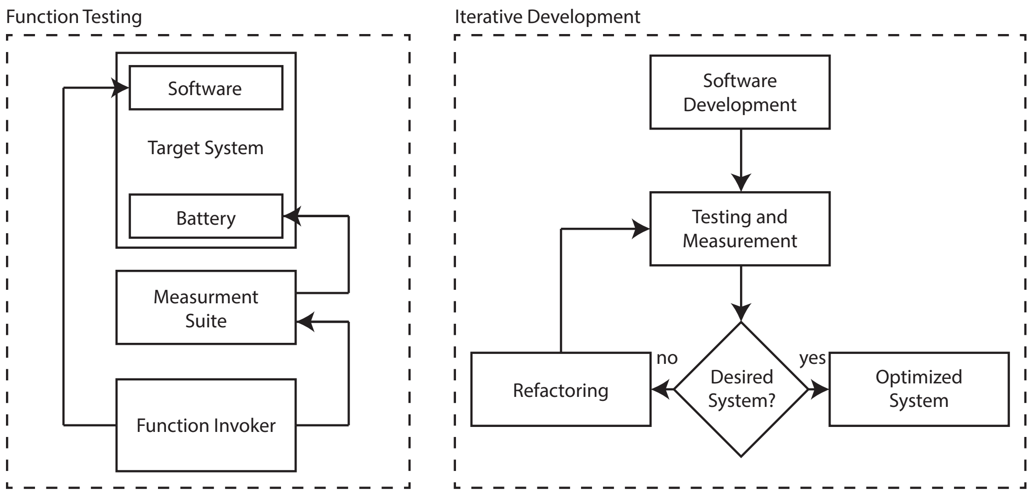


Figure 2.14: Function testing with iterative process

2.5.3 Run-time Measurement

The identification of critical code-segments (*software*) is a complex process. Run-time measurement is a proper approach for the identification of critical segments with a high power consumption. Power drain is measured while the system-software is running. The information is used afterwards to optimize the affected code-sections. There are different ways of implementing the run-time measurement.

Function testing is an option where an external or independent tool (*function-invoker*) is invoking functions or methods of the target-systems software application (*target-system*) which should be measured. The function-invoker can be any sort of software-component that is communicating with the target-system. This function-invoker executes a set of test instructions and gathers information about the power consumption of the target-system. The gathered data derives directly from the target-system itself or from some external attached measurement-suite. Thereafter it is possible to analyze the results and apply optimizations to the target-systems software application (e.g. rewrite/redesign critical code-sections, adapt time-out/wait operations). The whole process can be made iterative to improve the system progressively. The repeated steps should be included to the software development process to gain feedback of the current implementation. Figure 2.14 shows an abstract function testing environment and the iterative process with the described function-invoker, target-system and an external measurement suite.

Beside the functional measurement which is executed from external tools there is in-place section testing. The whole measurement process is directly included in the main software-application of the measured target-device. Thus there is no utility for external test initiation needed. The measurement starts with a special marker (*start-marker*) in the source-code and stops with another marker (*stop-marker*). The start-marker invokes the power measurement and the system is gathering the requested values until the application reaches the stop-marker. The collected measurement data can be stored and used to

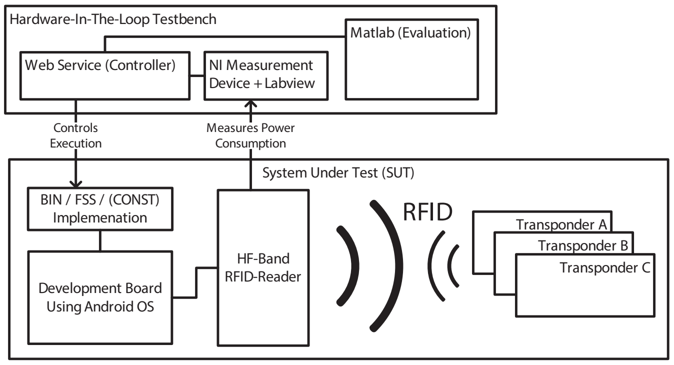

Hardware-In-The-Loop Testbench
Matlab (Evaluation)
Web Service (Controller)
NI Measurement Device + Labview
Controls Execution
Measures Power Consumption
System Under Test (SUT)
BIN / FSS / (CONST) Implemenation
Development Board Using Android OS
HF-Band RFID-Reader
RFID
Transponder A
Transponder B
Transponder C

Figure 2.15: Run-time hardware in the loop test-bench [16, pg. 6]

adapt the systems power consumption. An additional useful extension is labeling of the measurement sections with a tagging algorithm. The start and stop-marker will receive a tag to get a reference to each other. This tag-label can be anything (e.g. function name) or even a basic string identifier. The advantage over the function testing is the fast and comfortable implementation. Disadvantages are code-overhead (which can be removed after optimization) and the leak of encapsulation between application logic and measurement. Another disadvantage could be the accessibility of the measuring-component. Algorithm 1 shows an elemental pseudo-code of a measurement execution. After the measurement task the collected data is processed and critical sections or functions can be identified and optimized. Combination of both types can lead to a hybrid-system that is capable of performing external measurements execution with start-markers and end-markers.

Algorithm 1 Elemental in-place section test example for NFC

```
1: procedure TestProcedure
2:     initialisation
3:     id1 ← all identifier (tag)
4:     id2 ← request and authentication identifier (tag)
5:     startMeasurement(id1, id2)
6:     request(param, ...)
7:     authentication(param, ...)
8:     stopMeasurement(id2)
9:     read(param, ...)
10:    write(param, ...)
11:    ...
12:    stopMeasurement(id1)
13:    return
```

An actual system implementation in a HIL environment is discussed by Menghin and Druml [16, pg. 6]. Figure 2.15 illustrates a measurement set-up with an android development board as SUT and the web-service is the controller and invoker of the function testing.

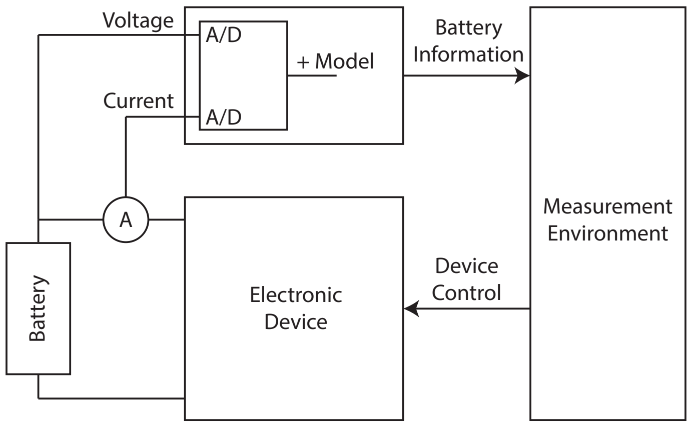


Figure 2.16: Voltage and current sensing with extended measurement environment [26, pg. 51]

2.5.4 Battery Sensing

Accordingly to [26] most cell-phone measurement environments are focusing on the main battery parameters like voltage, impedance and current. With state of charge (SOC) measurement it is possible to define a model for the power drain and charge state of a mobile device. This model can be used afterwards to optimize energy consumption and increase battery lifetime. Most SOC based measurement approaches are describing procedures to improve estimation accuracy. Generally the models can be separated in three generic categories as discussed in [27]:

- *Physical*: Description of physical measure
- *Empirical*: Extraction of information from testing and creation of an abstraction
- *Abstract*: Creation of a similar theoretical system
- *Mixed*: Combination of the other three described categories

[26] discusses some examples on how to determine SOC. Figure 2.16 illustrates a basically set-up for a battery evaluation and measurement system. Measured parameters are voltage and current, furthermore it is possible to calculate the electric power output ($P = U \cdot I$). The digitalization of the collected physical parameters is provided by analog-digital converter. This approach can be used to connect the measurement environment with the SUT.

2.6 Evaluation of Devices

2.6.1 Device Selection Criteria

The selection of appropriate NFC devices is a major object for the design and implementation process of this thesis. Various NFC devices are available nowadays and therefore it is necessary to determine which components are appropriate in order to cover the requirements for such a test and evaluation framework. Many use-cases are covering payment and ticketing systems but there are also other NFC applications available (e.g. *smartposters*).

Langer, Saminger, and Grünberger [28, pg. 2052] are mentioning three main types of testing NFC related devices and equipment. This testing types are specified by the NFC forum testing working group.

- Conformance testing
- Interoperability testing
- Performance testing

Langer, Saminger, and Grünberger [28, pg. 2052] explain that the antenna of NFC enabled smartphones highly influence the quality of the transmission. Thus the design of the antenna is important because of different device cases or the available space for the antenna. Langer, Saminger, and Grünberger are specifying a separation of the test cases in categories as seen in table 2.2.

Category	Description
Proximity range	Determine the minimal and maximal transfer and detection range for devices.
Communication hole	Measurement for communication holes in the proximity range.
Collision avoidance	Verification of the device is prone to collisions if there are multiple communication partners in proximity range.
Read operation	Check if there are malfunctions for read operation.
Write operation	Check if there are malfunctions for write operation.
Speed adaptation	Verification of the transmission speed after switching to a different speed rate.
Protocol	Testing the protocol reliability and correctness.

Table 2.2: Separation of test cases in categories

2.7 Generic Architecture

Opperman and Hancke [29, pg. 2] are describing a generic NFC system for monitoring or control functionality, where some sort of communication interface (reader) is placed between the monitoring and observed device. Opperman and Hancke are defining this interface as the gateway which is capable of short range data exchange. The transferred information can be control commands for a programmable logic controller (PLC) or stored sensor data. A combination of controlling and monitoring is also feasible. The recommendation for the utilized gateway is a NFC enabled smartphone. The interface is recording and storing sensor data and if the smartphone is in proximity range for data exchange the gateway and target device are connecting each other via NFC. Figure 2.17 represents the setup of a

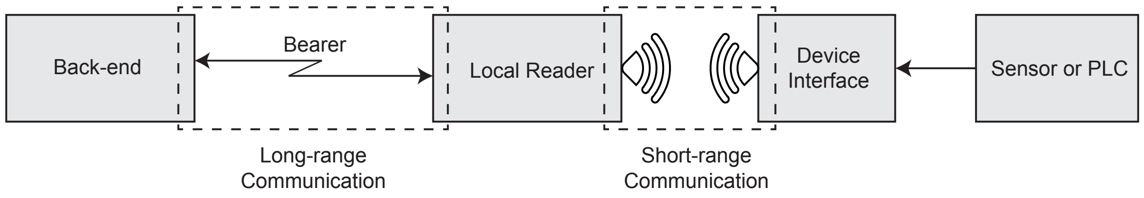


Figure 2.17: System architecture for a basic system [29, pg. 2]

simple gateway system. Opperman and Hancke want to avoid the use of power consuming hardware to enable wireless connection. So the target device can be assembled with NFC technology to create a short range communication channel. The received information from the target device on the smartphone can distributed to different electronic components with use of long range communication channels on the smartphone (e.g. *short message service (SMS)*, *wireless local area network (WLAN)* or *universal mobile telecommunications system (UMTS))*.

Opperman and Hancke point out that such systems are essential for low-power or even powerless devices. Thus it is possible to create a complete powerless sensor interface for various use-cases. Every time the smartphones magnetic field is in close proximity to the sensor it activates the interface and the sensor is able to measure, collect and transmit the data. All energy needed for this process comes from the magnetic field that is established by the smartphone. For the system that should be implemented and developed in this thesis the target device will have its own power supply and accordingly the NFC gateway interface, because the battery of the smartphone should be used as little as possible. [29, pg. 3]

Another reason for NFC enabled interfaces or gateways are the inexpensive assembly and implementation costs for this technology to add a wireless communication layer. Other technologies are more complex and expensive.

2.7.1 Open Platform

[30, 31, 32] clarifies open-source hardware and the potential of making hardware accessible for all individuals. Open-source hardware is similar to open-source software. Design and implementation are public available and everybody can assemble it. The advantages over proprietary hardware are collaborative public enhancements, cost reduction and comfortable extensibility. The planned measurement framework in this thesis is aimed to be open-source in order to make this solution public available for research and science.

As discussed earlier, it is essential to create an open environment including free and open-source software (FOSS). The projected system will use various devices thus it is necessary to establish stability and reliability in this eco-system. Therefore, proven firm- and software will be used to fulfill these objectives.

Linux Linux (Arch Linux) will be the operating system of choice because it is a stable, reliable and well documented system. It is expendable in terms of software and has a broad bandwidth of supported utilities and features. Linux will be the bottom layer of the software stack for the evaluation framework.

Android Android is a Linux based operation system from Google Inc. especially for mobile devices which is well established on the market. With approximately 71% market shares (2013) Android is the most used mobile operation system in Germany and still growing [33]. The applications on Android smartphones are implemented in Java[3] and executed in a Java environment called Dalvik Virtual Machine. The application development process is straight forward and extensively documented. The graphical user interface (GUI) can be defined with extensible mark-up language (XML) syntax which accelerates the development process significantly. Thus creation of user-interface prototypes is very rapid.

[3]object-oriented programming language

Chapter 3

Design

The chapter design gives an overview of the NFC evaluation frameworks concept. An approach with all involved parts and components will be illustrated to understand the fundamentals of that exposition. The design phase is the preparation of the concrete realization for the practical work. Therefore, the individual modules will be described in detail to channel the abstraction into real hardware later on. The mentioned modules can be certainly exchanged with other solutions that collaborate better with the targeted use-case application.

3.1 Synopsis

This chapter describes the design of the power evaluation framework for NFC systems. The related work that was discussed previously gives some fundamental ideas how to set-up such a framework. There are many different applications and use-cases for NFC architectures with the three communication specifications: reader-writer, card-emulation and peer-to-peer mode. This basic design will concentrate on the reader-writer and card-emulation mode, peer-to-peer communication can be extended later. These used modes require an initiator that is establishing the magnetic field to send request commands and a responding target. This serves as the bottom layer for the runtime-environment. In addition to that, the initiator and target can be observed in order to determine their energy consumption.

3.1.1 Basic Concept

The basic concept for the design is illustrated in figure 3.1 and explains the required modules for the framework. The initiator sends requests (Request A and Request B) to the target which is responding accordingly. This process is managed by the controller and the sequence is configured with help of the automation mechanism. Meanwhile the observable interface provides the observer with measurement data that is collected during information exchange between initiator and target. The measured data is tagged to create a relation to

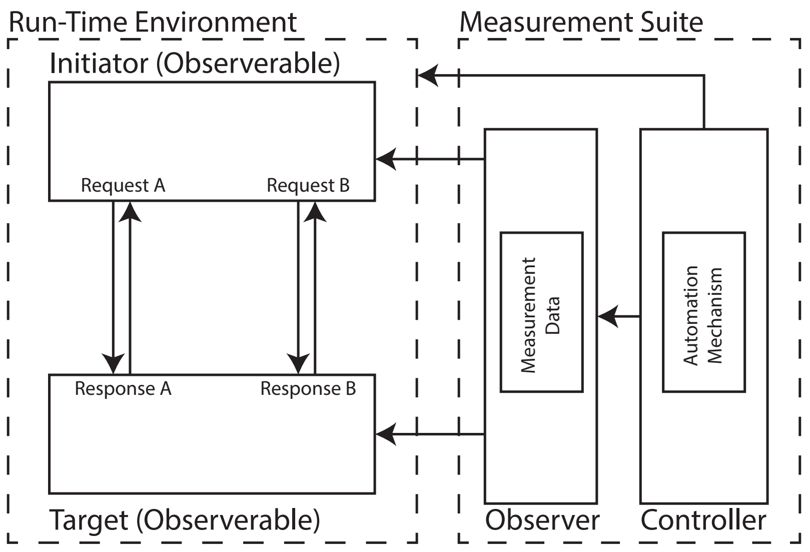


Figure 3.1: Basic concept

the executed commands (software) on initiator or target. The collected information from the observer is further processed and used to perform enhancements to the application that is executed on the system.

3.1.2 System Composition

The two basic modules are run-time environment and the attached measurement suite. These modules can be subdivided in various components as shown in table 3.1 for run-time environment and table 3.2 for measurement suite.

Composition	Description
Initiator	Initializer of the communication exchange (Requester)
Target	Component which is answering requests (Responder)
Observable	Interface that can be observed for measurement data

Table 3.1: Run-time module elemental components

Composition	Description
Observer	Connected to the observable and collects measurement data
Controller	Interface to control initiator, target and observer
Automation	Automatic testing mechanisms for the application with help of the controller

Table 3.2: Measurement suite module essential components

3.1.3 Strengths

The strengths of this platform in order to decrease the time for the optimization process are listed in table 3.3.

Feature	Description
Adaptability	Initiator, target and controller logic should be simply replaceable with other applications to ensure a rapid adaptable environment.
Integration	Measurement can be simply integrated in the application but will not influence the original application.
Feedback	Gathered consumption information is analyzed and provides feedback for energy optimizations.
Simulation	Application execution in an generic hardware environment for real-time systems.
Extension	Additional development of extensions which can be appended to the initial basic evaluation framework.

Table 3.3: Features of the proposed system

3.2 Requirements

The modules are outlined and approximately defined. The next step is the description of requirements for each component to curtail the module functionalities. Tables 3.4, 3.5, 3.6, 3.7, 3.8 and 3.9 specify the necessary requirements for each component and the priority (P) with values: high 🟢, medium 🟠 and low 🔴.

Identifier	P	Requirement
R-1.1	🟢	Possibilities to improve the applications with measurement techniques and optimizations based on the results.
R-1.2	🟢	Generic test environment and avoidance of specialized test environments for use-case applications.
R-1.3	🟠	Reusable software components for different use-case scenarios (modular system design).

Table 3.4: Requirements - general

3.3 Architecture

The basic concept is the first approach to an accurate design that is used to implement the evaluation framework. The basic modules are defined and specified in detail now. The following section gives an overview of the general design that is created and implemented.

Identifier	P	Requirement
R-2.1	🟢	User interface (GUI based) for input handling and feedback.
R-2.2	🟢	Establishment of the magnetic field and NFC connection to the target with internal or external module.
R-2.3	🟢	Recognition and identification of emulated transponders by the target module.
R-2.4	🟢	Possibility to attach measurement suite to the power consumption of the NFC module over the observable interface.
R-2.5	🟢	Protocol accordingly to ISO 14443 3-4 specification and custom enhancements (data packaging).
R-2.6	🟢	Exchange possibility for the NFC module so that external and internal interfaces are possible.
R-2.7	🟢	Generic interface for the underlaying NFC hardware so that fast reader-context switching is possible.
R-2.8	🟠	Fast development capabilities for the operating system platform (application execution environment and feature-rich software stack).
R-2.9	🟠	Open-source hardware and software for simple reproduction and documentation.
R-2.10	🟠	Support of discrete protocol definitions that are application-context depending.
R-2.11	🔴	Handle multiple (different) targets in the proximity range at the same time.
R-2.12	🔴	Internal power consumption measurement and data up-link bridge for the observer.

Table 3.5: Requirements - initiator (portable device)

Identifier	P	Requirement
R-3.1	🟢	Internal or external NFC modules are possible to answer the initiators requests.
R-3.2	🟢	Exchange possibility for the NFC module and the communication protocol/link.
R-3.3	🟢	Emulation capabilities of various transponder types based on ISO 14443 3-4.
R-3.4	🟢	Open-source hardware and software for simple reproduction and documentation.
R-3.5	🟠	Application logic (software) platform independent and executables for different operating system possible.
R-3.6	🟠	External accessories (e.g. sensors, indicators, actuators, etc.) can be attached and controlled by the target.
R-3.7	🟠	Hot-swapping of the emulation process to constantly provide a transponder to the initiator.

Table 3.6: Requirements - target

Identifier	P	Requirement
R-4.1	🟢	Measurement of various simple physical parameters.
R-4.2	🟢	Controllable with the attached observer.
R-4.3	🔴	Observable is able to store measurement data internal which is later fetched by the observer.

Table 3.7: Requirements - observable interface

Identifier	P	Requirement
R-5.1	🟢	Measurement of the observable interfaces with accurate timings for invocation and stop.
R-5.2	🟢	Gathering of measurement data from the attached observables.
R-5.3	🟢	Management by the measurement controller.
R-5.4	🟠	Saving of measurement data and related meta-data in internal storage and a data up-link to the controller.

Table 3.8: Requirements - observer

Identifier	P	Requirement
R-6.1	🟢	Initialization and monitoring of the whole measurement sequence.
R-6.2	🟢	Storage system for measured data.
R-6.3	🟢	Automation with application measurement configuration (predefined script for the testing process).
R-6.4	🟢	Possibility for measurement data export (analysis with external tools and software).
R-6.5	🟠	Internal computation and analysis of gathered values.
R-6.6	🔴	Automatic highlighting and indication of source-code segments with high power consumption.

Table 3.9: Requirements - measurement controller and automation mechanics

3.3.1 Fundamentals

The initiator that is included in the run-time environment can be a typically NFC reader. This reader establishes the communication channel to the target with a request-response protocol. The reader is controlled from user-input or from the software itself. Therefore, state-of-the-art portable devices with integrated NFC technology can act as reader which initiates the magnetic field and communicates with the target because they are able to handle user-interactions (e.g. touch-screen or buttons) to report feedback to the user (e.g. display). The NFC chip is not necessarily required because external readers can also be attached and the portable device is only used for computation (CPU) and user feedback. For this design the portable device has two main functions: handle user interaction and establish the NFC connection.

On the other hand, the target is responding to the initiator as a transponder. In this design the target emulates the NFC tag and contains a computational unit (CU) and the NFC

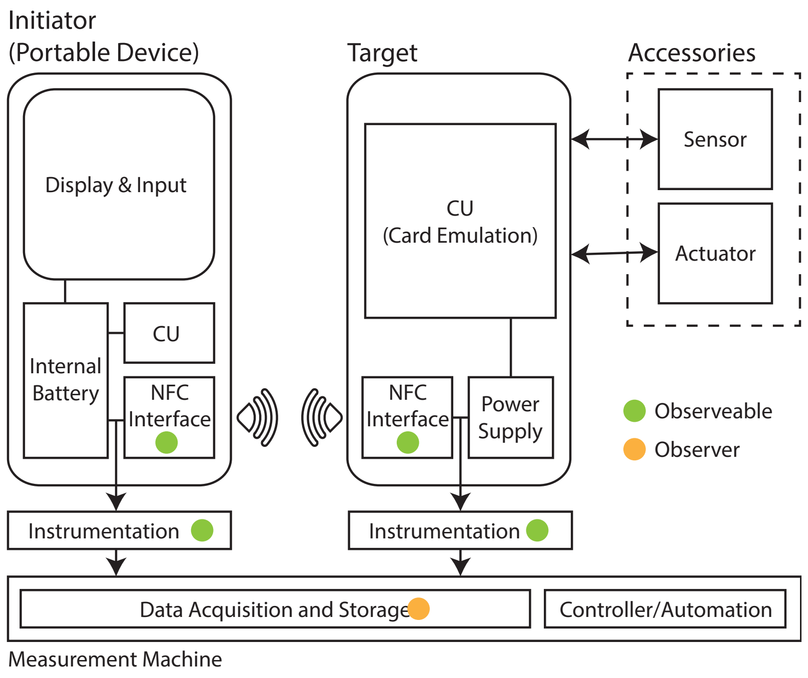


Figure 3.2: System architecture with essential components

element. The application logic is exchangeable on both sides (initiator and target) as this is important for the adaptability. The computational unit can be connected with various external accessories (e.g. sensors) which are required for the active application. This system is the definition of the basic run-time environment and in addition the measurement suite can be attached. Simple volt and ammeters can be utilized for the measurement process of the portable devices observable interface (external reader) and the targets NFC extension. The measurement instruments are connected to an independent computer for measurement data computation and storage.

3.3.2 Outline

Depending on the basic concept figure 3.2 shows a draft of the design with the fundamental modules.

Initiator As discussed previously, the initiator is a portable device with NFC interface and an own computational unit which is executing the initiator software of the application. The underlaying operating system provides a fast development and deployment process. The device handles and processes the user input. The power supply is an internal battery which is charged over the power network.

Target The target is an encapsulated machine with a standard operating system and the NFC interface is connected over wire. The target machine executes the opposite part of the software application. It can be combined with external accessories which can be controlled and prompted by the initiator over the NFC communication channel. For example, the target is connected to a temperature sensor so that the portable device is able to request the actual value from the target.

Observable Interfaces Initiator and target provide power supply wire branches from the NFC interface to measure the power consumption of the device. In this design the obervables are primitive electric circuits for current and voltage measurement with attached instruments. In an advanced design these observables can be replaced with independent and autonomous interfaces which can be modified and adapted to the application requirements.

Measurement Observer The observer is the module that is gathering and processing the measured values from the observable interfaces which can be used to analyze the executed software on the initiator or target. The measurement machine supervises the observer component and stores the collected data in the internal memory.

Measurement Controller and Automation Mechanics The controller and the integrated automation takes care of the whole measurement sequence and is responsible for the correct invocation of the software components on the initiator and target platform. The measurement process of the observer is timed within the executed application and calculates the results that are needed for system optimizations.

3.4 Design

General design is a more precisely definition of the included components and devices. The raw design showed a mostly abstract approach of the proposed system and in this section the framework is defined to make explicit decisions for the following implementation.

3.4.1 Component Classification

The previously design outline illustrates two basic components (run-time environment and measurement suite) and the subdivided parts (target, initiator, observable interfaces, measurement observer, controller and automation). The target is already defined as a portable device and for this design a state-of-the-art smartphone will be used because applications can be quickly developed and it is a proper interface for user input. Several smartphones have a NFC chip included which can be used as initiator, but also external NFC devices can be appended to the device. The target is a low-cost single-board computation platform with an over the wire attached and programmable NFC device called gateway. The measurement suite is a simple personal computer with a programmable measurement kit and a hardware interface for volt and ammeter. So the use-case application that is

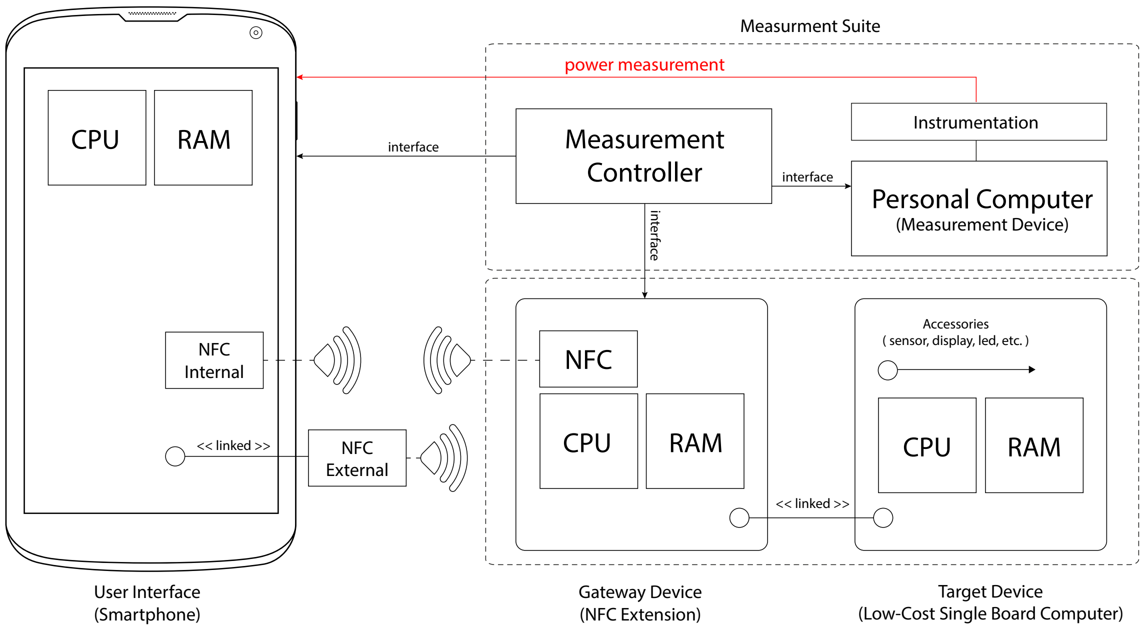


Figure 3.3: Evaluation framework design

developed with help of this framework is executed on the smartphone and the single-board computer. Measurement is invoked from the personal computer. Figure 3.3 shows the design of the evaluation framework.

3.4.2 Run-time Environment

The application under test is basically executed on the run-time environment (smartphone and single-board computer). The development for these devices is tied to the underlying system. Most smartphone operating systems provide a software development kit (SDK) for rapid software development. This SDK is used to implement the initiator application and the GUI is used to handle the user input. The target application can be implemented with any programming language that is applicable for the single-board computer.

3.4.3 Measurement Suite

The personal computer is connected to the smartphone's battery when the internal NFC chip is used or directly to the external NFC extension depending on the used interface. The suite can also be connected to the single-board NFC extension in order to measure the power drain. Measurement is based on voltage and current usage by the communication interfaces. The computer initiates the measurement with a custom application which can be configured for automation purpose.

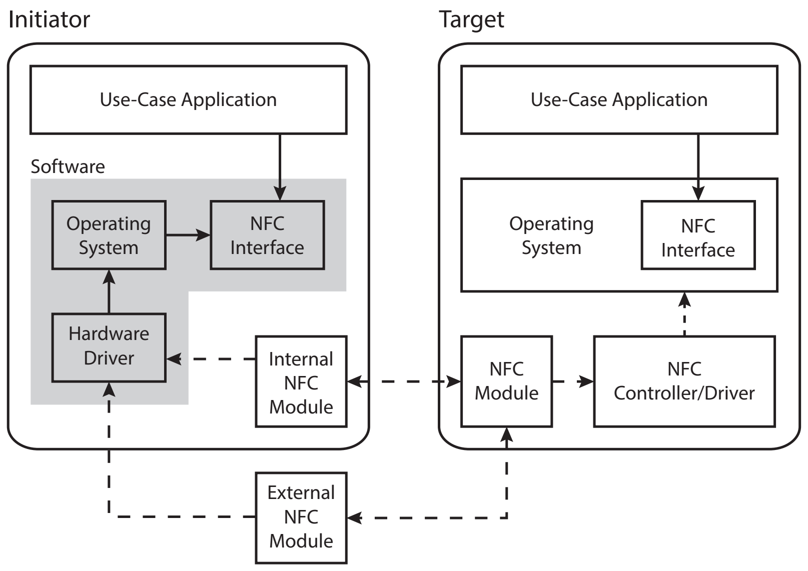


Figure 3.4: Simple masking of underlaying hardware

3.4.4 Generic Approach

The intention of the measurement framework is the generic approach which can be accomplished with exchangeable components. The executed application has no information about the underlaying framework and how it operates. The system provides interfaces for the software application on initiator and target so that the core functionalities are accessible. Figure 3.4 illustrates the generic approach with separation of NFC hardware and use-case application.

3.4.5 Communication Loop

This describes the cycle of information exchange between target and initiator. The transfer can be based on standards like ISO 14443 3-4 or other specifications. The evaluated use-case application consists of two major parts: the executed software on initiator and target. The software on the initiator side is sending commands to the target and is waiting for response. The necessary initialization steps are covered from the fundamental run-time environment (operating system). On the target side the requests are fetched and passed on to the software application which is computing the response. The data is depending on the application context.

Figure 3.5 shows a request-response challenge between initiator and target. For example, the portable device (initiator) is a smartmeter [1] which is able to read data from measurement cells (target). Thus the initiator is requesting a simple n-Byte value from the target (e.g. sensor value) therefore the ISO 14443-3 specification is used with a `read` request. The

[1]measurement device

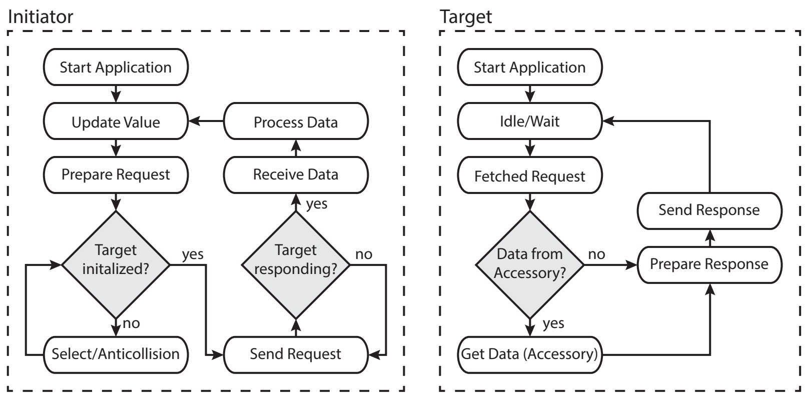


Figure 3.5: Request-response loop

target responds to the request with the measured value and the smartmeter can process the received value. Selection and anticollision are handled by the operating system.

Chapter 4

Implementation

After the design process the basic system implementation is discussed and utilities and tools for realization are defined. This section describes the final evaluation platform and also approaches the emerged challenges through the implementation process. Furthermore, the implemented software and extended firmware are extensively documented in order to explain how they can be utilized in other NFC applications or services with less effort.

4.1 Overview

The implementation is the practical part of this thesis in order to prove that the concept of this evaluation platform for NFC systems is working. In this implementation the initiator is the device under evaluation (measuring of the power consumption). This project will only implement software components for the environment. State-of-the-art hardware will be used to provide the basic computation platforms. The development stages are basically described in the following sections, and will give a brief overview of the development process. This will cover the implementation of the framework and the application scenario which will be used for the case study.

4.1.1 Brief Description

The development tasks are the foundation for a strict procedure to facilitate a successful implementation. Therefore, in the first phase the previously abstract design will be modeled to a real-hardware composition. Searching for appropriate devices that are fitting the requirements and the definition of the real-world concept is the initial task. Thereafter the collaboration of the selected devices is tested and analyzed to fabricate a elemental underlying hardware architecture for the software realization that is developed in the subsequent step.

After the completion of these tasks the basic run-time environment is ready to execute use-case applications which can be evaluated and optimized on this evaluation framework. These use-case examples will be implemented to give a proof of concept for the developed measurement based evaluation framework and therefore it is necessary to attach a measurement suite to the executing run-time environment. An existing implementation of such a measurement component will be appended and used for the evaluation. The utilized measurement suite is based on a previously project.

To get a brief overview of the implementation tasks, the definition of the general objectives that will be covered during the whole work progress in order to generate a reliable evaluation framework is summarized in the following list:

- Adapting the conceptual design from 3.4.1 to a simple real-world solution for replication purpose.
- Identification and selection of appropriate hardware that covers the predefined requirements from 3.2.
- Evaluation and testing of chosen hardware in terms of software compatibility and operating system fundament.
- Implementation and testing of the computation logic for all included modules to provide a basic run-time environment.
- Development and realization of application examples for the proof of concept that is discussed in 5.
- Integration of an existing measurement suite to the run-time environment, and configuration for the application examples.
- Documentation of the newly developed modules to provide a fundamental framework for future projects discussed in 7.

4.1.2 Schedule and Road-map

The implementation tasks is defined and described, but the progress of development is an iterative cycle. To realize and complete all required task a schedule for the implementation is shown in figure 4.1 to clarify the processing and execution of the work-tasks. Tasks can be closely interlinked to the next and previous neighbor and must be adapted iteratively to fit in the existing setup.

As an extension to the development schedule an associated road-map is illustrated in figure 4.2 that displays the chronology progress. The identification of objectives and stages in this road-map helps to give an outlook for each milestone in the project. This road-map is no indication for the working time that is spend for each milestone or objective.

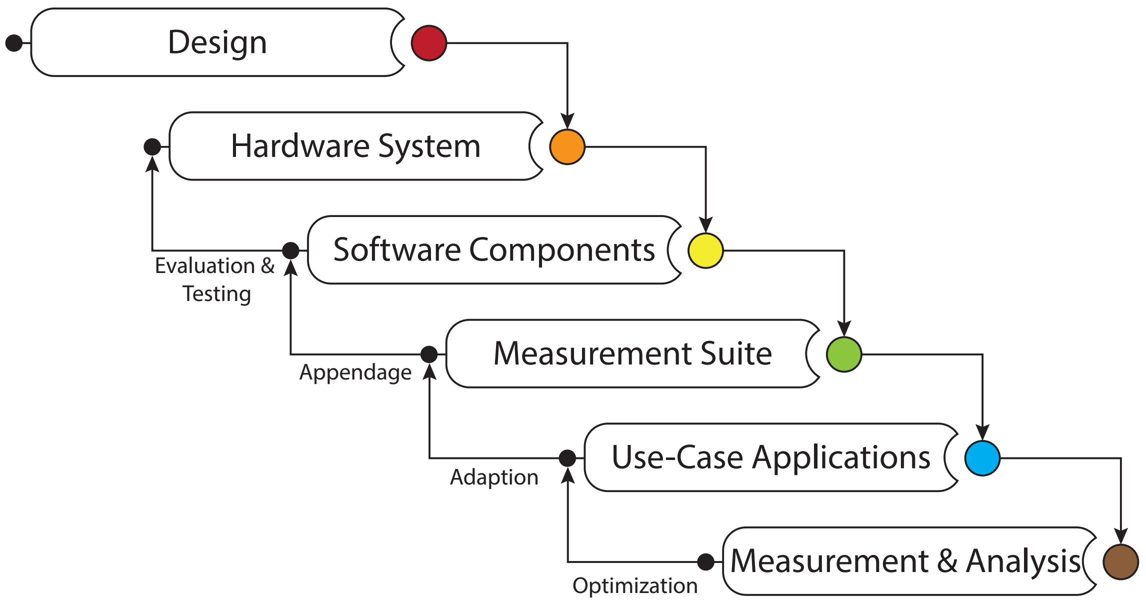


Figure 4.1: Development Timeline

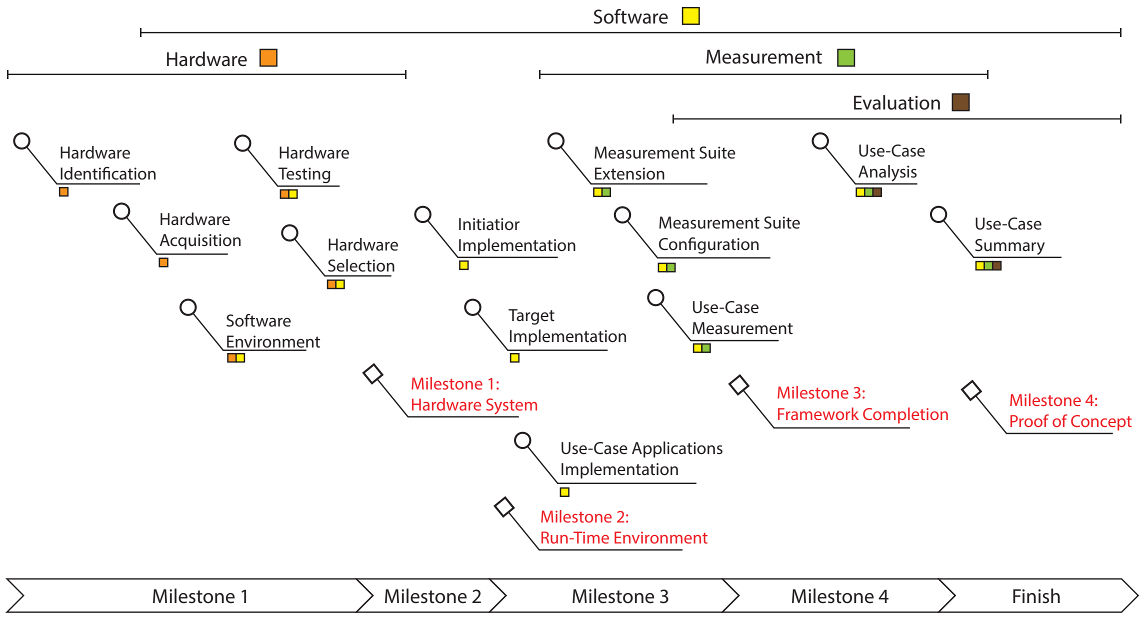


Figure 4.2: Development road-map

4.1.3 Design Realization

As discussed earlier the very first implementation task is the adaption of the existing design to a real-world solution. Therefore, all components have to be classified to specify a system with state-of-the-art hardware. The design pretends an user-interface (initiator) as smartphone and a simple single-boarded computation platform as the target. Android is the operating system of choice for this solution because it features a fast development process, software testing and GUI libraries. Android application execution environment is based on the Java programming language with dalvik virtual machine (DVM) as run-time environment and therefore the initiator software will be implemented accordingly.

The NFC module for the smartphone will be an external reader device which can be controlled programmatically. The advantage of this system is the fixed and reproducible NFC initialization. With an external device the user is able to control the phone interface without restrictions or limitations. Another advantage of this external device is the directly programming interface for the developer. Android handles the internal NFC chip autonomously so there is no direct connection to gain full control over the chip. With an external reader the developer directly controls the integrated NFC chip. Also the immediate measurement of power consumption is problematic because a wired interface is required for the process that means disassembling the smartphone. A further problem in terms of measurement is the assignation of energy consumption to the internal NFC module. The determination is a complex task which can be avoided with an external device because plain wire connections to the power supply of the reader can be used to measure the power consumption. The software of the NFC module implements the same interfaces like the Android system which provides them for the application execution environment. That guarantees the simple changeability of the reader device to the internal module. The communication between the smartphone and external reader is implemented wireless so that the phone can be freely moved.

The target will be a single-boarded ARM platform because of its energy efficiency and performance. The target device should be simply replaceable and therefore the software on this machine is also developed with Java to provide a platform-independent architecture. The target software can be executed on various different machines because Java is available for a wide range of computation platforms. Java is executed within an virtual environment called java run-time environment (JRE) which facilitate the platform independence. The target also has an external NFC component such as the smartphone. This NFC gateway module is capable to emulate tags within the ISO 14443 3-4 specification to establish a connection and can be programmed with no limitations (firmware). This module is communicating with two devices. On the first side to the smartphones external reader and on the other side to the connected single-board platform. The connection to the target is a wired interface because it is not necessary to make the target movable in the current stage. The wired link is used for data transmission and also supplies the module with power. A simple circuit attachment is possible to integrate this module in the measurement suites evaluation process.

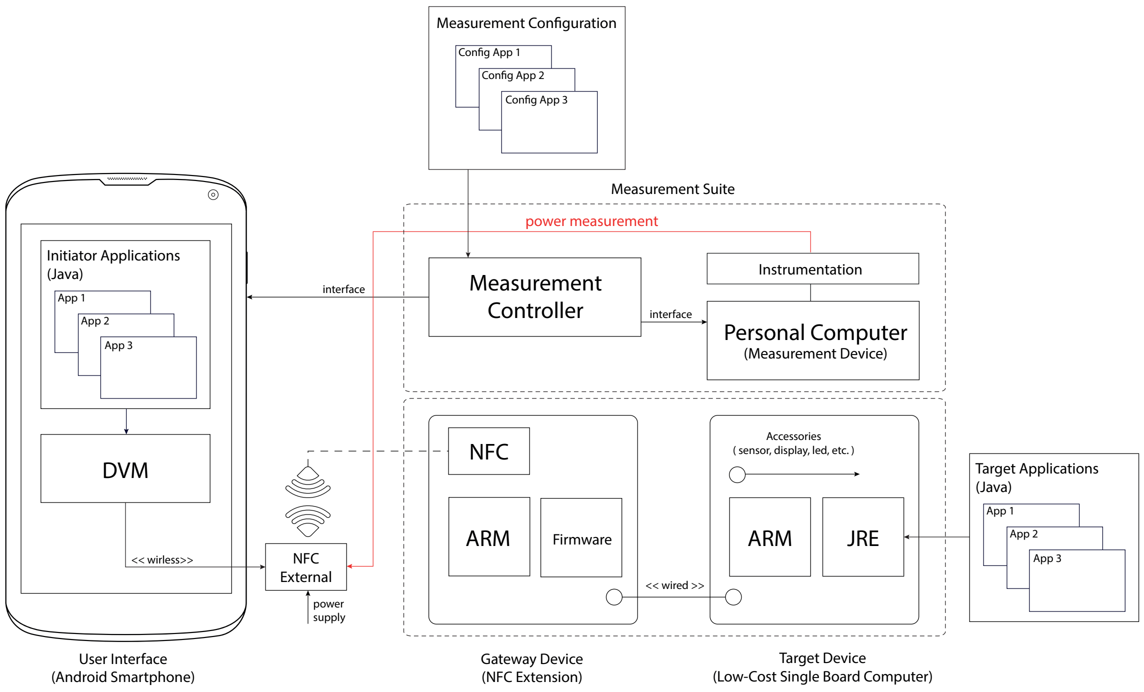

Measurement Configuration
Config App 1
Config App 2
Config App 3
Measurement Suite
power measurement
Initiator Applications (Java)
App 1
App 2
App 3
interface
Measurement Controller
Instrumentation
interface
Personal Computer (Measurement Device)
DVM
NFC
ARM
Firmware
Accessories (sensor, display, led, etc.)
ARM
JRE
Target Applications (Java)
App 1
App 2
App 3
<< wirless>>
NFC External
power supply
<< wired >>
User Interface (Android Smartphone)
Gateway Device (NFC Extension)
Target Device (Low-Cost Single Board Computer)

Figure 4.3: Design realization with real hardware

As earlier described, the attached measurement suite is available from a previously project and provides interfaces for voltage and current measurement. The suite software is parameterized with a configuration that contains the necessary measurement sequence that is responsible for the correct invocation and timing of the whole process.

Figure 4.3 shows the composition of the above described devices.

4.2 Hardware Components

In this section the used hardware is described and specified in detail in order to reproduce the whole evaluation framework. The devices are selected because they are compatible and simple to obtain except of the Proxmark3 that can be assembled with the open-source blueprints or from a reseller [1]. The mentioned hardware can be equipped with custom software to cover the requirements and provide the underlaying run-time and evaluation stack for the use-case application measurement.

4.2.1 Initiator User-Interfaces

The state-of-the-art smartphones that are used for this implementation are the common mentioned development phones from Google (Nexus series). The mobile devices are shipped

[1] http://www.proxmark3.com/

Specification	Description
CPU	Qualcomm Snapdragon^{TM} S4 Pro CPU (Quadcore 1.5 GHz Krait)
GPU	Adreno 320
Chipset	Qualcomm Snapdragon^{TM} APQ8064
RAM	2GB
Battery	Non-removable Li-Po 2100 mAh battery
Sensors	Accelerometer, gyro, proximity, compass, barometer
Display	True HD IPS Plus capacitive touchscreen, 16M colors (768 x 1280 pixels, 4.7 inches (~318 ppi pixel density))
Storage	Internal 8/16GB
WLAN	Wi-Fi 802.11 a/b/g/n, dual-band, DLNA, Wi-Fi hotspot
Bluetooth	v4.0 with A2DP
NFC	Broadcom Controller BCM20793
USB	microUSB (SlimPort) v2.0
OS	Android OS v4.4.2 (KitKat)

Table 4.1: Technical Specification - LG E960 (Nexus 4) [34, 35]

with the plain and standard Android user interface that is part of the original source-code. Other manufacturers are assembling customized user-interfaces on top of Googles Android implementation. Google also introduced the Nexus series as the reference hardware platform for application development on Android operating systems. Another reason for choosing phones out of the Nexus series is the internal support of NFC with an integrated chip. The applications that run on the target are not developed for a specific type of hardware or mobile device and therefore should be deployable and executable on several Android platforms.

LG E960 (Nexus 4)

As the name suggests the Nexus 4 is the fourth smartphone generation of the Nexus series introduced by Google. This mobile device was selected because it was the latest available reference hardware that is mentioned for Android development. As of now the Nexus 4 was replaced [2] by the new generation of the series. The Nexus 4 features an internal NFC module from Broadcom which is also a selection choice for this phone. A major disadvantage of this phone is the encapsulated battery so the phone must be disassembled if the measurement should be attached to the phones internal battery to record the power consumption of the NFC chip. But this disadvantage does not affect the actual implementation because an external NFC module is used for the measurement and evaluation setup. Table 4.1 lists all relevant technical specifications for the Nexus 4 smartphone that are relatable to this implementation. Figure 4.4a shows the Nexus 4 smartphone.

[2]Nexus 5 introduction, October 2013

Specification	Description
CPU	1.0 GHz ARM Cortex-A8
GPU	PowerVR SGX540
Chipset	Hummingbird
RAM	512MB
Battery	Li-Ion 1500 mAh battery
Sensors	Accelerometer, gyro, proximity, compass, barometer
Display	S-LCD capacitive touchscreen, 16M colors (480 x 800 pixels, 4.0 inches (~233 ppi pixel density))
Storage	Internal 16GB
WLAN	Wi-Fi 802.11 b/g/n, DLNA, Wi-Fi hotspot
Bluetooth	v2.1 with A2DP, EDR
NFC	NXP PN65N
USB	microUSB v2.0
OS	Android OS, v4.1.1 (Jelly Bean)

Table 4.2: Technical Specification - GT-I9023 (Nexus S) [38, 39]

GT-I9023 (Nexus S)

As a second user-interface the Nexus S is used which is the second generation of the Nexus series. This portable phone was introduced to demonstrate the exchangeability of the user-interface with less effort. The chassis of the Nexus S can be simply opened on the backside. Thus the measurement suite can be attached to the internal battery of this phone. This is just a possibility for the measurement setup and is not actually used in the current implementation which uses a common generic interface for the NFC communication over the external reader as mentioned earlier. The internal proprietary NFC module is a mature and proven chip from NXP Semiconductors which has a high potential to be integrated in the evaluation framework [36, 37]. Table 4.2 lists all relevant technical specifications for the Nexus S smartphone that are relatable to this implementation. Figure 4.4d shows the Nexus S smartphone.

Android Device Emulator

The development is driven by the Android emulator that is shipped with the SDK because no real-hardware is necessary to develop the user-interface for the initiator. The emulator is a visualization of an Android hardware device and has the equal execution environment for the Java applications. With the ARM hardware emulation the performance of such a virtual device is limited because the emulation itself requires numerous resources in addition to the JRE which is the second layer of virtualization. Therefore the SDK provides a hardware accelerated execution manager (HAXM) created by Intel which speeds up the emulators execution. HAXM is suitable for CPUs that support VT-x [3], EM64T [4], and Execute

[3]Intel virtualization on x86

[4]extended memory 64 technology

Disable (XD) Bit. In addition with an Android x86 [5] operating system HAXM can be utilized to accelerate the software execution on a standard x86 development computer [40]. The emulator supports various different android virtual device (AVD) images like the Nexus 4 and S, so that applications can be developed within the virtualization and deployed on real hardware. The emulation can be initiated with the following command which `avd_name` is the image of the emulated device (Nexus 4 or S):

```
$ emulator -avd <avd_name> [-<option> [<value>]] ... [-<qemu args>]
```

4.2.2 Initiator NFC Modules

There are two types of NFC elements in the system: internal and external ones. Integrated modules in the smartphone are universal communication chips with a variety of supported technologies (e.g. Bluetooth, WLAN). The interface of this chips is provided by the Android operation system and is hardware independent. The attached external NFC modules for the smartphones are state-of-the-art reader devices. They can be programmed over personal computer/smart card (PC/SC) or proprietary custom software application programming interface (API) (vendor mode). They also provide the standard NFC commands as a facade and they manage the protocol communication with the soldered internal chip. In this implementation they are not directly connected to the smartphone, a standard personal computer (PC) acts as a bridge and provides the reader over a socket connection to the network.

Internal Modules

The internal NFC modules assembled on the smartphone were evaluated and tested in this practical work but they were not utilized. The reasons for this decision is the API that is provided by the external reader and the circumstantial measurement setup for internal modules. Also several problems occurred during device evaluation which are discussed and explained in section 4.8. The integration of the internal modules can be extended in future work 7 until the Android API features more low-level commands for the NFC interface like the introduction of host-based card-emulation [41] in Android 4.4 (KitKat) [42].

DUALi DE-620 USB NFC Reader

The DUALi DE-620 USB NFC Reader (short: DE620) is the primary and fully featured external NFC extension for the initiator (smartphone). The development will focus on this device and implements the same NFC interface as the standard Android API. The major advantage of this reader is the proprietary vendor mode for ISO 14443-3 specified communication. The actual system implements the DE620 in vendor mode and skips the PC/SC mode. During software implementation the reader is connected to the development

[5] CPU instruction set architecture

Specification	Description
CPU	ARM 32-bit Cortex-M3 (48MHz)
RAM	20KB SRAM
Storage	128KB Flash
Power Supply	Voltage: 5V DC, Current: 150mA
USB	2.0 Full speed interface (12Mbps)
NFC Standard	ISO/IEC 18092 NFC, ISO 14443 Type A/B, NXP Semiconductors Mifare, Sony FeliCa
Antenna Frequency	13.56 MHz
Antenna Distance	20-50mm
Speed	106-424 kbps
Peripherals	2x LED
API	PC/SC and Proprietary (Vendor Mode)

Table 4.3: Technical Specification - DUALi DE-620 USB NFC Reader [43]

Specification	Description
Power Supply	Voltage: Regulated 5V DC, Current: 50-200mA
USB	2.0 Full speed interface (12Mbps)
NFC Standard	ISO/IEC 18092 NFC, ISO 14443 Type A/B, NXP Semiconductors Mifare, Sony FeliCa
Antenna Frequency	13.56 MHz
Antenna Distance	about 50mm
Speed	106-424 kbps
Peripherals	Dual color LED (green/red), monotone buzzer
API	PC/SC and CT-API (PC/SC Wrapper)

Table 4.4: Technical Specification - ACR122U USB NFC Reader [44]

machine and in the final setup the reader is able to connect with the measurement machine or another PC. Table 4.3 lists all relevant technical specifications for the DE620 reader that are relatable to this implementation. Figure 4.4b shows the DE620 reader.

ACR122U USB NFC Reader

The ACR122U USB NFC reader (short: ACR122U) is the secondary extension that is used for parallel testing and evaluation purpose. The reader is implemented with standardized PC/SC mode to support a simple plug-and-play solution. The ACR122U features a basic functionality for the NFC communication and the interface for the smartphone. The ACR122U can be purchased with an extra SDK but that is not necessarily required because Java provides a generic solution (Java Smartcard IO) for the integration in other software. Table 4.3 lists all relevant technical specifications for the ACR122U reader, that are relatable to this implementation. Figure 4.4e shows the ACR122U reader.

4.2.3 Target Platforms

The target platform is the foundation for the target application execution and is a common computation machine. The target software is designed to be independent and can be executed in a JRE. This platform is emulating a transponder tag which is activated by the initiator and responds to request commands. This emulation logic has the possibility to integrate additional accessories from the target platform and further on this machine acts like a bridge to another device which is controlled by the user (smartphone).

Development Machine

During the implementation a development machine is the central component for the whole evaluation framework. The development machine emulates the Android smartphone with the execution of the initiator application, provides access to the external NFC reader and also operates as the target with the associated software stack and application. The development environment is described in section 4.4 to illustrate the detailed implementation progress of the evaluation framework. The wired connection to the devices are powered by universal serial bus (USB) therefore the development machine requires at least three unused ports. This platform guarantees a simple and comfortable environment for development and makes rapid deployment and testing possible.

Raspberry Pi Model B (rev. 2)

Later on the development machine is replaced with a Raspberry Pi single-board computer to operate in an own environment. This ARM computation platform takes control over the tag emulation and the connection to peripheral devices (e.g. sensors, actuators, etc.). Linux (ArchLinux [45]) is the operation system on this computer because it is lightweight and management is possible over secure shell (SSH). The system works out-of-the-box and is installed on a common standard high capacity (SDHC) card. Some modifications are necessary in order to work with the attached NFC module and other connected utilities. A major advantage of the Raspberry Pi platform is the high connectivity with electric circuits over the integrated general purpose input/output (GPIO) interface which allows implementation of various different use-cases with simple hardware elements [46]. Table 4.5 lists all relevant technical specifications for the Raspberry Pi single-board computer that are relatable to this implementation. Figure 4.4c shows the Raspberry Pi Model B hardware.

4.2.4 Target Gateway

The target gateway device is the extension of the Raspberry Pi with an interface for NFC. The gateway takes control of the hardware communication and pass-trough the received data to the target software on the Raspberry Pi. During implementation progress the

Specification	Description
CPU	ARM1176JZFS v6 32Bit Single Core + (VPU) and DSP, 700MHz
GPU	Videocore IV, Dual Core, 128 KB L2-Cache, 250 MHz, HDMI 1.3a
Chipset	Broadcom BCM2835 SoC
RAM	512 MB RAM, 400 MHz
Storage	SD Memory Card Slot (SDHC), Class 4 & 6 Cards
LAN	RJ45 10/100 MBit/s Ethernet, internal over USB-Controller
Indicators	5x LEDs (Power, SD-Card Access, LAN 10/100 MBit, LAN Full-Duplex, LAN Link/Access)
NFC	NXP PN65N
USB	2x USB 2.0 internal Hub (5V Supply max. 3,5W (700mA))
GPIO	26 Pin Port @5V, 3,3V, GND and 17 3,3V GPIO Pins (SPI, I2C, UART) with 2-16mA

Table 4.5: Technical Specification - Raspberry Pi Model B (rev. 2) [48, 47]

gateway device is directly connected to the development machine in order to rapidly deploy software changes.

Proxmark3

The Proxmark3 is a small hardware device for research RFID and NFC technology [49]. It features a large spectrum of supported modes like snooping, listening, emulating in the two frequency ranges: high frequency (3-30 MHz) (HF) and low frequency (30-300 kHz) (LF), only HF is used in this context because of the NFC specification. The Proxmark3 is based on an open-hardware design and can be self assembled or purchased also the whole software is freely available [50] and can be modified under GNU General Public License. This software has basically two major components: firmware and client. The firmware is the operating system of the Proxmark3 and the client software is the application which is the interface to control the Proxmark3 from the host-machine. This implies that this tool is connected to another computer. Windows and Linux are supported operating systems for the host-platform and the connection is established over USB. The device is powered over the USB connection but also runs without the host-machine and firmware modifications. A detailed description of the used protocol and control system between the firmware and the client of the host-machine is available in section 4.7.3. The bulk of the software implementation is written in plain C [6] so some parts have to be adapted to Java for the evaluation framework. Table 4.6 lists all relevant technical specifications for the Proxmark3 extension that are relatable to this implementation. Figure 4.4f shows the Proxmark3 hardware.

[6]imperative programming languages

Specification	Description
CPU	Atmel AT91SAM7S256 55 MHz
RAM	64kB
Storage	256Kb
FPGA	Xilinx Spartan-II
RF	Independent circuits for HF and LF
USB	mini-USB 2.0
Interface	1x key button, 4x LED
Antenna	External with USB 2.0 mini 4-Pin Hirose

Table 4.6: Technical Specification - Proxmark3 [52, 51, 53]

Specification	Description
CPU	Cortex A8
RAM	256 MB LPDDR
Storage	256 MB NAND Flash
LAN	10/100Mbps

Table 4.7: Technical Specification - OMAP35x Evaluation Module

4.2.5 Measurement Suite

The measurement suite is the expansion for the run-time environment and is used for the evaluation of the use-case application under test. The initiator software that is executed on the development machine (Android emulator) is deployed on the Android development board and on the measurement machine. Furthermore, the development machine is now only used to observe the Android development board with the android debug bridge (ADB).

Android Development Board (OMAP35x)

The measurement for the case-study described in chapter 5 was done with an Android development board, because of the direct local area network (LAN) connection to the measurement machine. This reduces the delay between the measurement suite and the application under test. The development board is a OMAP35x Evaluation Module (EVM) from Texas Instruments [7] with Android 2.2 (Froyo) as operating system. Table 4.7 lists all relevant technical specifications for the OMAP35x that are relatable to this implementation. Figure 4.4g shows the Proxmark3 hardware.

Measurement Machine

The measurement machine is the computer that executes the measurement software and measures the power consumption of the external NFC reader. The socket connection to

[7]http://www.ti.com/tool/tmdsevm3530

the smartphone invokes different methods on the initiator software and the measurement machine collects voltage and current (2-channels) from the connected reader. The measurement machine is a NI PXI-1042Q from National Instruments and the reader is wired with a SCB-68 I/O connector block that is connected to the PXI-1042Q. META[:SEC:] Evaluator 4 is the software that is used to measure the power consumption and call the methods from the configuration. The setup and explanation for the measurement suite is described in chapter 5.

4.3 Hardware Composition

This section describes the development setup for the evaluation framework with the earlier introduced hardware. The linking of the integrated hardware devices is either accomplished with a serial interface or over a plain socket connection depending on the requirements. The first step is to assemble the fundamental run-time environment and the second step is the extension with the measurement suite. The development is done on a PC and therefore the machine is the central device for the realization process. The development environment is replaced later with the measurement machine and the single-board computer.

4.3.1 Development Setup

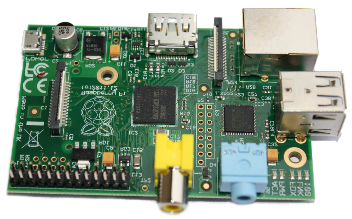
Foremost task is the creation of the run-time environment consisting of the initiator (user-interface and reader) and the responding platform (target and attached gateway). Figure 4.5 illustrates the setup of the basic utilities of the framework in general over USB and table 4.8 describes the utilized connections between the included devices and shows required or optional linking in order to provide a runnable environment.

Id	Type	Req.	Description
Link A	Socket	●	TCP/IP over LAN/WLAN connection for the Android smartphone to the development machine in order to control the external reader.
Link B	Serial		USB debugging and application deployment connection for the ADB (*Link A* also can be used for this purpose).
Link C	Serial	●	USB connection between the development machine and the external NFC reader.
Link D	Serial		USB connection between the development machine and the Proxmark3 for firmware flashing and control mechanics.
Link E	Analog	●	Connection from the Proxmark3 to the attached antenna with USB cords. Only for power supply and analog signal transfer.

Table 4.8: Connection Description - Development setup

(a) LG E960 (Nexus 4)

(b) DUALi DE-620


(c) Raspberry Pi Model B

(d) GT-I9023 (Nexus S)

(e) ACR122U

(f) Proxmark3

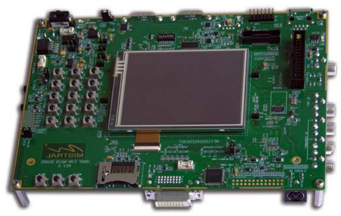

(g) OMAP35x

Figure 4.4: Overview of the utilized hardware equipment

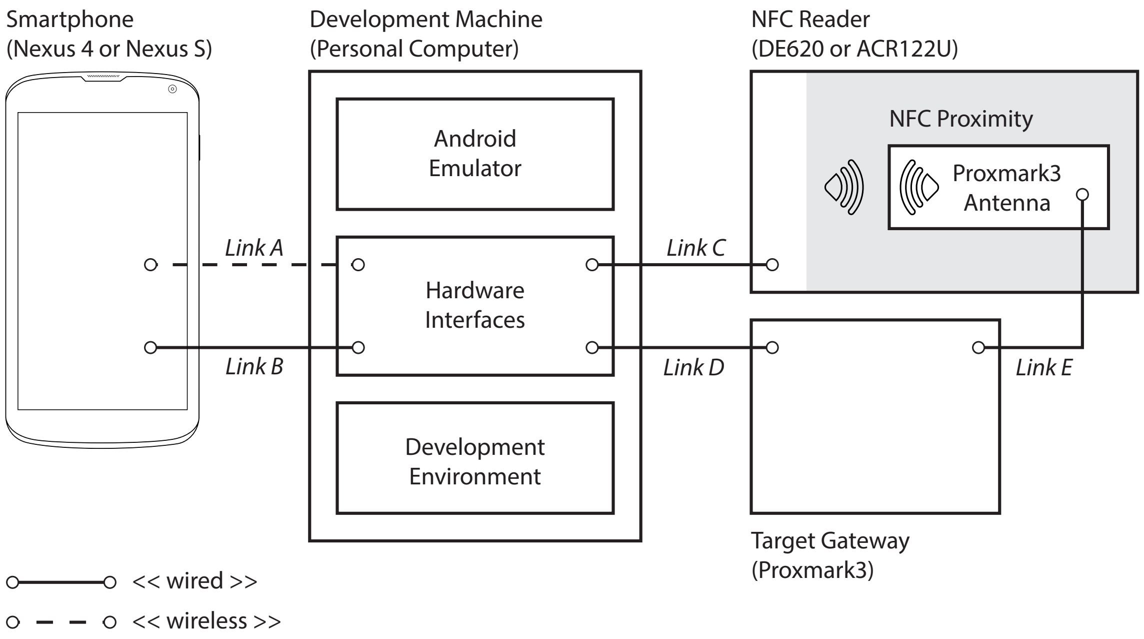


Figure 4.5: Development hardware setup

4.3.2 Evaluation Setup

The difference between the previous setup is the reduction of the development machine and relocation of the executed software on this computer. The development machine administrates the programming logic for the user-interface (when the emulator is used), the external reader and the gateway of the target. So the development machine is the all-in-one device for the implementation process and the Raspberry Pi as target is not even needed. For the productive setup with the inclusion of the measurement suite the software is distributed to the corresponding gadgets.

User-interface can be either deployed on the hardware smartphone, the Android development board or the emulator which is connected to the development machine (can be also executed on the measurement machine). The control software for the Proxmark3 (gateway) is now executed directly on the Raspberry Pi (target). Modifications to the firmware of the Proxmark3 are not possible at the current stage of implementation but can be extended in future work. Initiator and target software can be modified in the productive setup. The client application for the external reader is also executed on the measurement machine because the USB connection is also required to determine power consumption of the device. Figure 4.6 demonstrates the evaluation setup of the utilized hardware in this project including the measurement suite with the Android development board as the initiator platform without the requirement of the development machine. Table 4.9 describes all used connection between the devices.

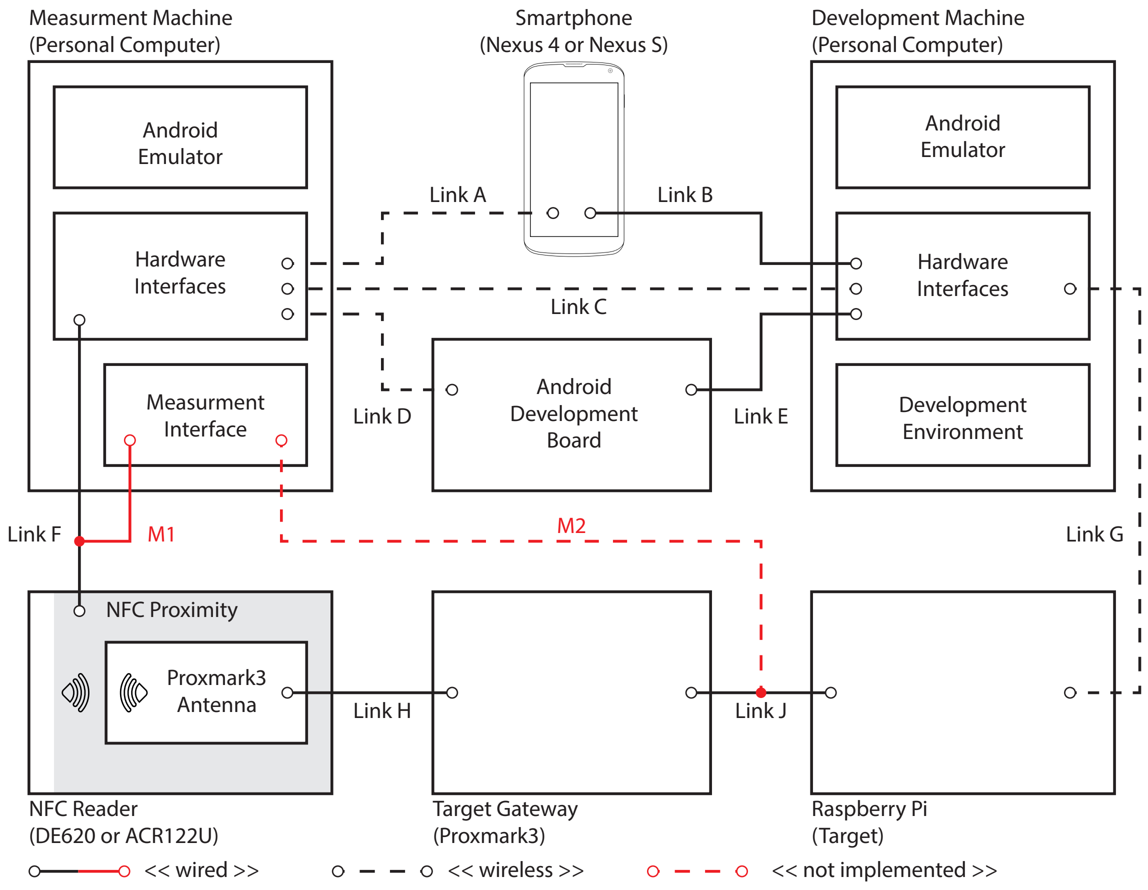


Figure 4.6: Evaluation hardware setup

Id	Type	Req.	Description
Link A	Socket		TCP/IP over LAN/WLAN connection for the Android smartphone to the measurement machine in order to control the external reader.
Link B	Serial		USB debugging and application deployment connection for the Android smartphone using ADB (also possible over LAN/WLAN).
Link C	Socket		TCP/IP over LAN/WLAN connection for the Android emulator (development machine) to the measurement machine in order to control the external reader.
Link D	Socket	●	TCP/IP over LAN/WLAN connection for the Android development board to the measurement machine in order to control the external reade.r
Link E	Serial		USB debugging and application deployment connection for the Android development board using ADB (also possible over LAN/WLAN).
Link F	Serial	●	USB connection between the measurement machine and the external NFC reader (data and power transmission). Software for the reader is executed on the measurement machine.
Link G	Socket		SSH over LAN/WLAN connection to manage the Raspberry Pi remotely (deployment and control).
Link H	Analog	●	Connection from the Proxmark3 to the attached antenna with USB cords. Only for power supply and analog signal transfer.
Link J	Serial	●	USB connection between the Raspberry Pi (target) and the Proxmark3 (gateway) for control mechanics (flashing not implemented).
M 1	Analog	●	Managed power measurement of the attached external reader (DE620 or ACR122U) with an analog interface for current and voltage.
M 2	Analog		Managed power measurement of the Proxmark3 (gateway) with an analog interface for current and voltage (not implemented in the actual stage but can be extended in future work).

Table 4.9: Connection Description - Productive setup

4.4 Development Environment

The section development environment explains how the software components were created and which common programming practices were applied. As mentioned earlier Java is the development language of choice (except the Proxmark3 firmware which is C based) because Android applications are mainly based on Java and the JRE is available for a wide range of various operating systems and platforms. Implemented software-code can be reused especially for protocol and communication purpose. For the implementation common software patterns [54] were used to guarantee a modular, maintainable and extensible software design.

4.4.1 Tools

Java: The applications for initiator (smartphone), target (Raspberry Pi) and the external reader (DE620 and ACR122U) are implemented in Java (Version 1.7) and can be executed on the standard JRE except the external reader implementation which must be executed in a 32bit JRE. This is necessary because the Duali DE620 is shipped with a proprietary interface (dynamic link library (DLL) based) for Windows operating system. The DLL is mapped with java native access (JNA) [55] to provide the low-level reader interface to the Java application (details see 4.7.2).

Android SDK: The development of Android applications is driven by the Android SDK. This development environment is shipped with various tools which are helpful during implementation process. With the integrated download manager the SDK can be updated comfortably and additional extras might be installed like the earlier described HAXM or support libraries. The manager also administrates the different API versions from early Android 1.5 (API 3) until the latests version Android 4.4 (API 19). The Android applications implemented in the practical part require a minimum API level of 8 and an actually target level of 18.

C: The Proxmark3 open-source software is based on C therefore the modifications are directly made in the firmware implementation. The Proxmark3 sources can be compiled with the common GNU compiler collection (GCC) (version >= 4.7) and optionally a Subversion client and Perl interpreter (for inclusion of a version string in the firmware). A tutorial for the setup of the compilation and flashing process can be found in the appendix B.

Eclipse: Eclipse [56] is the integrated development environment (IDE) which is used to develop, compile and execute the software components in Java and C. Which is not a requirement but rather a personal choice. The assembled Java software will be distributed as JAR files to the target platform (Raspberry Pi) and the measurement machine. The compilation of the Proxmark3 software will be performed with Makefiles that can be executed with the Eclipse IDE. The major advantage of this development environment is the inclusion of all software components in an all-in-one solution for Android, Java and C. Furthermore, the sharing of common used source-code is possible which avoids duplicated code segments.

There are many alternatives for compilation and management like Ant [57] or Maven [58] which are more lightweight and can be configured faster than a whole Eclipse setup. These tools will also properly work with the implemented Java components.

4.4.2 Testing and Debugging

In order to ensure an appropriate software quality the implementation is developed with suitable testing tools and based on test-driven development (TDD) approaches. The Java software components are tested with JUnit (Version 4) [8] and the Android implementations with the internal unit testing tools [9]. The firmware implementation of the Proxmark3 could be also tested with CUnit [10] or other unit testing frameworks for C. In this project this part was skipped because the firmware is only slightly modified and tunnels the received commands directly to the target (Raspberry Pi). Only very few calculation logic is implemented on Proxmark3 side. Debugging for the Java components and the Android applications was done with internal Eclipse debug tools.

Debug Messaging

The official Proxmark3 implementation supports a custom debug messaging system in which the Proxmark3 is sending message packets over the serial connection to the client software which can be further processed or printed. This messaging system provides a simple and fast remote debugging mechanism for developers with program execution on the Proxmark3. The debugging is based on the communication protocol for the Proxmark3 and the client software. A detailed protocol description can be found in section 4.7.3.

Logging

The debug messaging of the Proxmark3 firmware is not a properly debugging technique but rather a logging mechanism for the client software so this system works like the LogCat [11] for the Android environment. A logging subsystem was appended to the developed Java software components to get feedback of the executed application. The logging is based on simple logging facade for Java (SLF4J) [12] with Log4J [13] as back-end framework. The back-end framework can be exchanged with different logging implementations if required.

4.5 Software Components

In the following sections the utilized software components of the evaluation framework will be introduced. Some of this applications are depending on already existing interfaces or protocols and were implemented to cover the necessary requirements which will be

[8]`http://junit.org/`

[9]`https://developer.android.com/tools/testing/index.html`

[10]`http://cunit.sourceforge.net/`

[11]`http://developer.android.com/tools/help/logcat.html`

[12]`http://www.slf4j.org/`

[13]`http://logging.apache.org/log4j/2.x/`

explained accordingly. An extensive description as well as implementation details of the actual software applications will be given in section 4.7.

4.5.1 Reader Bridge

The Android reader bridge implementation is the connection to the external executed NFC reader software. This is basically a shared library that has to be imported in every use-case application in order to use the DE620 or ACR122U as the NFC communication module which is required to gather power consumption of the use-case application. The current implementation is based on Mifare Ultralight and therefore the `android.nfc.tech.NfcA` class from the Android SDK is the reference for the external reader communication interface. Mifare Ultralight is used because it is compatible to ISO 14443-3 and the Nexus 4 internal NFC chip is not supporting the common Mifare Classic transponder types. This was a design decision for the current realization an advanced implementation can be developed with any type of transponder hardware and emulation. Use-case applications NFC context can be switched between the external and internal reader with a simple module exchange. The use-case application has no knowledge how the bridge is communicating with the reader and which protocol is used for this purpose. This guarantees a simple changeability for the utilized NFC element on the initiator side.

4.5.2 Reader Interface

The reader interface software provides the external NFC reader to the Android bridge. This software is based on plain Java and has to be executed in a 32bit JRE because the DE620 is controlled with a DLL. The ACR122U is connected with the PC/SC interface. The reader manager also provides a simulation mode and therefore no real hardware is needed and the responses are generated by the software itself. The simulator can be freely implemented by the developer to fulfill the use-case application requirements and assists during the Android application development.

4.5.3 Proxmark3 Firmware

The Proxmark3 as the gateway device is based on three general software components:

- The bootrom (bootloader) which initializes the hardware, provides the flashing function over USB and transfers execution to the operating system. The bootrom is the backbone of the device and should only be flashed if really necessary (in terms of compatibility).
- The field programmable gate array (FPGA) image converts the analogue signals from the external attached antenna to digital information that can be computed with the ARM processor.

- The operating system is the software that is executed after hardware initialization (bootrom). It can be modified in any way and flashed to the Proxmark3. This implementation uses a extended version of this operating system with a customized emulation loop to ensure a gateway mode for the Proxmark3 (pass-through).

The operating system is booting and set in an idle mode waiting for a client software connection. The Proxmark3 source-code includes a native client which is implemented in C. The Proxmark3 firmware and the client are using a serial connection and a customized protocol for communication. The existing client (also implemented on the operating system) features a broad bandwidth of RFID tools for LF an HF operations.

The standard firmware is computing the emulation of the NFC tag directly on the Proxmark3 and the client only initializes the emulation. This does not cover the requirements because the operating system has to be flashed for modifications of the use-case application and no peripherals (e.g. sensors) can be attached to the Proxmark3 platform. So it is important that the Proxmark3 is only acting as a gateway and forwards all received commands to the target (Raspberry Pi). The target is computing the response and sending it back to the Proxmark3 which responds to the requested command from the NFC reader.

4.5.4 Proxmark3 Client

The existing Proxmark3 client is not needed for the actual system because a custom client software was developed for the tag emulation. This implementation is written in Java and executed on the target (Raspberry Pi). The Java Proxmark3 client implements the communication protocol from the original client and initializes the gateway mode on the Proxmark3. This client internally computes the responses for the use-case under test and therefore it can be loaded with customized response configurations during start-up.

4.5.5 Use-Case Applications

Use-case applications in this system are separated into two general parts which might be tested and evaluated with the framework:

- **Initiator:** Android application with the reader bridge implementation for the NFC communication.
- **Target:** Response configuration that is executed within the Proxmark3 client and additional software from the peripherals which is exchanging data with the Proxmark3 client.

4.6 Software Composition

This section describes the data flow between initiator and target with the introduced software components. The Android application initiates the communication with the reader bridge and interface over the socket connection. The interface implements the basic methods from the `android.nfc.tech.NfcA` class and maps this object to the readers native functions. The protocol that is used with the socket connection is a request-response challenge and data is encoded in JavaScript object notation (JSON) for readability. The request from the Android application transports an identifier and optional function parameters to the reader software. The application is blocked until the response data is received.

The reader interface software executes the given method with the additional parameters so that the reader hardware performs the data exchange. Afterwards the reader waits for the response of the gateway or continues execution after a predefined timeout.

The gateway device (Proxmark3) receives the emitted command from the reader, passes through it to the target and waits for the response from the target. The Proxmark3 has a customized protocol for the serial connection to the target. The data exchange is also based on a request-response protocol but within plain Byte format.

The target receives the command (APDU Frame) from the gateway over the serial connection, dispatches it and computes the response accordingly to the received command. This response is send back to the gateway which encodes the command for the NFC connection. The response data is received by the reader and forwarded to the initiator through the controller and the interface. Figure 4.7 illustrates the software composition and how they are connected and interacting.

4.7 Software Implementation

4.7.1 Reader Bridge

The reader bridge class `NfcSim` implements `android.nfc.tech.NfcA` from the Android SDK. The constructor of the `NfcSim` class needs a host parameter (`readerHost`) with internet protocol (IP) and port number for the connection to the reader interface. Thus it is possible to include multiple readers if necessary. The connection to the NFC reader has to be initialized with the `connect()` method. The `close()` method is disconnecting the reader interface. After connection establishment basic functionalities for `read()` and `write()` accordingly to `NfcA` (Mifare Ultralight) is available. The more advanced method `transceive()` can be used to send custom APDU commands to the reader interface.

The connection is established with a socket and data is encapsulated in the class `Packet` with a message code and a payload. The socket connection is partitioned in incoming and outgoing data so therefore the application is listening for incoming data and sending data when required. The message code is an identifier for the bridge and the reader interface to determine which command should be executed. This `Packet` is encoded with help of the `Gson` library [14] into JSON format for readability. Currently the protocol supports the

[14] `https://code.google.com/p/google-gson/`

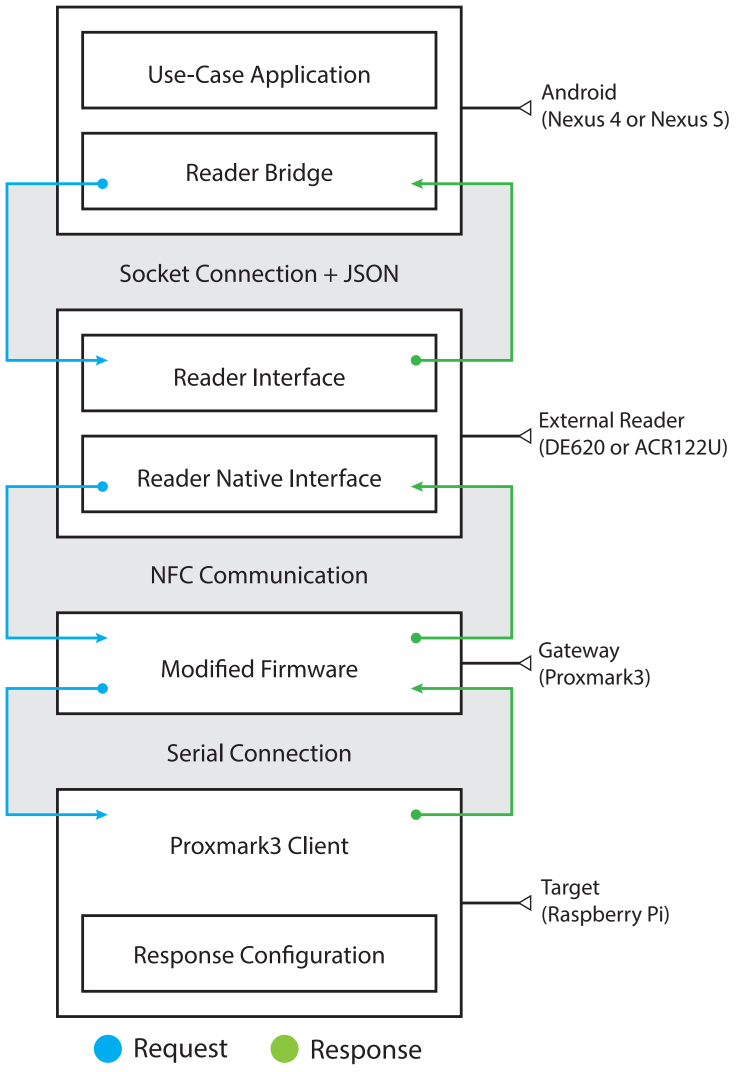


Figure 4.7: Software composition

basic request-response commands (`READ` , `WRITE` , `TRANSCEIVE`). If required, additional commands can be added within the `Packet` class. Because of the socket connection every Android application needs to provide network communication with the reader interface therefore it is necessary to enable the permission `INTERNET` in the `AndroidManifest` .

Listing 1 shows the basic usage of the `transceive()` function of the reader bridge with a simple `read` request.

Listing 1 Example of a custom Android use-case application

```
public class ExampleApplication extends Activity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {

    // New NfcSim object with the IP of the reader interface
    NfcSim tag = new NfcSim("129.27.146.29");

    // Send request
    byte[] result = tag.transceive(new byte[] { 0x30, 0x01 });

    // Process the result ...
  }
}
```

4.7.2 Reader Interface

The reader interface is the counter-part for the reader bridge described above. The `CardReaderServer` class is the entrance point for the execution and can be called with different command line parameters:

- `-r <reader>` Indicates which reader should be loaded. Default reader will be DE620. Possible arguments are `de620` , `acr122` and `simulator` .
- `-h` Prints the help for the application.

After startup the reader interface initializes the socket and waits for the bridge to connect. When the bridge has successfully connected the reader interface dispatches all incoming commands to the initialized hardware reader or simulator. The simulator is a virtual device that behaves like a real hardware. The simulator receives commands and replies to it. This is helpful for the development process because the whole target back-end is not required.

DE620

This is the primary reader module which is used later for the use-case study and the measurement setup. The reader is shipped with a 32 bit DLL (Windows operating system) file which grants access to the reader hardware. So therefore it is required that Java application can access this DLL outside of the virtual execution environment. The JNA library [15] is included to map the methods Java internally to the accordingly native DLL methods. The `DE620DLLInterface` class is the connection to the native code execution.

The connection and data exchange with the hardware reader is executed as follows:

1. `DE_InitPort()` Initialization of the reader with a port for the serial identifier.
2. `DE_FindCard()` Detect transponders in the magnetic field of the reader.
3. Data Exchange with the Proxmark3 gateway device and interface for the reader bridge.
    - `DE_Transparent()` $\rightarrow$ reader bridge `transceive()` command.
    - `DEA_READ()` $\rightarrow$ reader bridge `read()` command.
    - `DEA_UltraM_Write()` $\rightarrow$ reader bridge `write()` command.

The `Select` and `AntiCollision` mechanics for the transponder are accomplished with `DE_FindCard()` but can be directly implemented if this is required for the concrete use-case application.

ACR122U

The ACR122U is hooked over PC/SC and has to be initialized with `connect()`. After the initialization the reader can be accessed over the provided methods or either natively with APDU commands. The methods `read()`, `write()` and `transceive()` are mapped directly to the reader bridge.

Simulator

The simulator is a tool for development process to simulate a real hardware reader. The simulator can be simple extended to fit the requirements of the use-case application. The idea behind this simulator is that the `Simulator` class is the template for the response configuration from the target executable. So for the development process no real NFC hardware is required. In future work the simulator and the target response configuration should share the same implementation for reusability.

Figure 4.8 illustrates the communication between the reader bridge and interface with the JSON `Packet` protocol.

It is recommended to create a java archive (JAR) for the execution of the reader interface software. Thus the application (with the DE620 as reader) can be started with:

```
$ java -jar reader.jar -r de620
```

[15] `https://github.com/twall/jna`

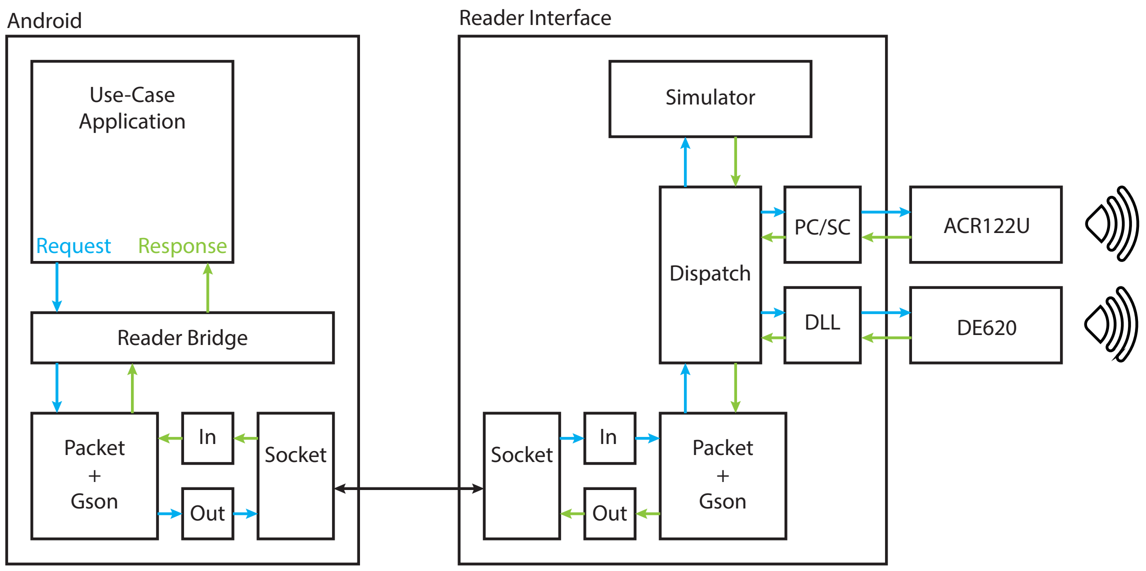


Figure 4.8: Communication between the reader bridge and interface

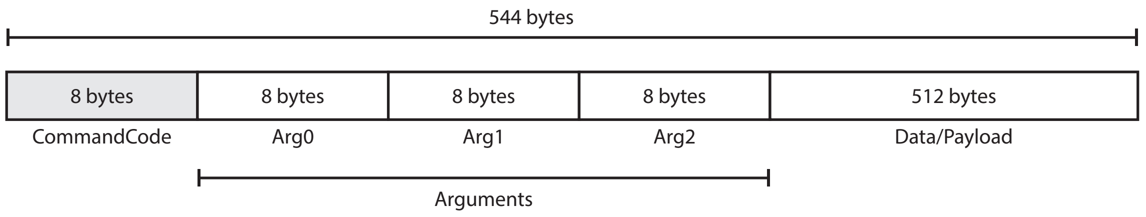


Figure 4.9: Proxmark3 protocol packet

4.7.3 Proxmark3 Firmware

The Proxmark3 firmware is open-source and available from the public online repository [16]. This firmware can be compiled and flashed to the device with the tutorial in appendix B.

The Proxmark3 starts when it gets connected over USB. After the bootrom initialization and firmware loading the Proxmark3 is idle and waits for an incoming command from the serial connection. For this implementation a additional functionality called gateway mode was added. In this mode the Proxmark3 forwards all incoming commands to the client and replies the computed response from the target (Proxmark3 client) to the reader.

So the Proxmark3 serial protocol is extended to make data communication possible in this gateway mode. A message packet of the protocol for the command/data exchange is shown in figure 4.9. The `CommandCode` indicates which command has to be executed on the receiving side. The arguments can be used to handle parameters that are important for the command itself. The data field is used for the payload that is processed during command execution.

[16] `https://github.com/Proxmark/proxmark3`

The main simulation loop that was introduced for the gateway mode can be found in `armsrc/iso14443a.c` within `SimulateIso14443aTagOnClient()`. When the emulation is initiated from the client software the Proxmark3 is listening for NFC communication from the reader device. After a command was successfully received the Proxmark3 wraps the command in a message package, sends it to the Proxmark3 client and waits for a message packet. When the response packet has arrived, the Proxmark3 extracts the `CommandCode` and performs the corresponding action. The existing `CommandCodes` are listed in `include/usb_cmd.h`. The additionally introduced `CommandCodes` for the message protocol:

- `CMD_SIMULATE_TAG_ISO_14443a_ON_CLIENT (0x0800)` which starts the gateway mode on the Proxmark3.
- `CMD_SIM_REQUEST (0x0801)` is the request command from the Proxmark3 to the client.
- `CMD_SIM_RESPONSE (0x0802)` is the response from the client to the Proxmark3.
- `CMD_SIM_RESPONSE_CORRECTION (0x0803)` is equal to `CMD_SIM_RESPONSE` but with parity correction.
- `CMD_SIM_RESPONSE_4BIT (0x0804)` responses for a `write` command with only 4 bit response.
- `CMD_SIM_CONTINUE (0x0805)` continues execution without sending a response to the reader.
- `CMD_SIM_STOP (0x0806)` stops the emulation (this is currently not supported).

4.7.4 Proxmark3 Client

The Proxmark3 client is the software that is executed on the target (Raspberry Pi) and mainly communicates with the Proxmark3 firmware over the specified protocol. The client software establishes the connection to the Proxmark3 with the `jSSC` library [17] for serial interfaces. Furthermore, the client loads the responder module which is a implementation of the `Command` Java interface and is the counterpart for the use-case application on the smartphone. After the connection initialization, the client automatically invokes the gateway mode on the Proxmark3. From that moment on all received messages from the reader device are handed-over to the client through the Proxmark3. The client software is polling for new messages from the serial interface and the `CommandProcessor` dispatches the commands and sends it to the loaded responder. This module computes the response and the client encapsulates it in a packet for the serial protocol and sends it back to the gateway.

Listing 2 shows how to implement a custom responder for a specific use-case application accordingly to the Android application.

[17] `https://code.google.com/p/java-simple-serial-connector`

Listing 2 Example of a custom responder class

```java
public class ExampleResponder implements Command {

  @Override
  public void execute(CommunicationCommand cmd) {

    byte[] data = cmd.getData();

    if (data[0] == 0x30) {
      // Compute read request and respond
      int blocknumber = data[1];
    }

  }
}
```

The Proxmark3 client provides some start parameters which can be used to start the client with different modules and automates the execution:

- `-r <responder>` Indicates which responder should be loaded for the dispatcher. An example might be `-r com.example.Responder` and the parameter needs the full name of the class including the Java package.
- `-i` Starts the interactive mode with a command line interface. This parameter skips the automatically starting of the gateway mode on the Proxmark3. The user has to start the mode manually with `sim`. With `version` the Proxmark3 bootrom, firmware and FPGA versions are printed to the screen. `quit` leaves the interactive mode and exits the client software.
- `-h` Prints the help for the application.

As mentioned earlier the sources should be packed into a Java JAR file for comfortable execution also the deployment process on the Raspberry Pi is faster with a executable JAR. With the following command the client can be started:

```
$ java -jar client.jar -r com.example.Responder
```

In figure 4.10 the communication between the Proxmark3 firmware and the client software for the target is illustrated.

4.7.5 Utilities

The utilities are a Java library that is shared among all other Java software components and contains routines that are commonly used. These tools provide formatting, conversion and manipulation of different basic Java types.

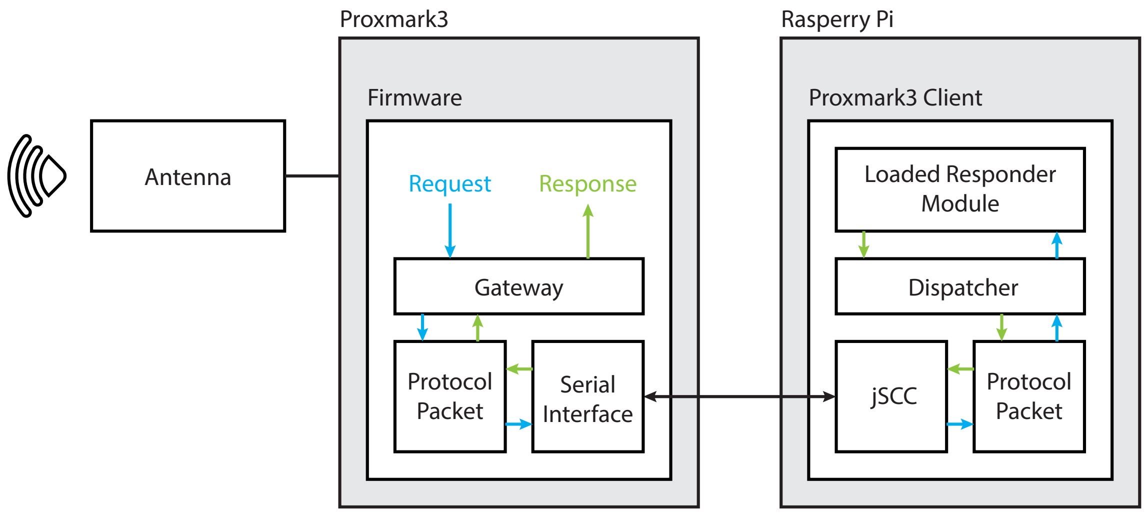


Figure 4.10: Communication between the Proxmark3 firmware and the client

4.8 Alternative Approaches

This section gives an overview of other implementation approaches with different hardware that was evaluated during this thesis and points out advantages and disadvantages for each system. Furthermore, it will be explained why these systems were not used and which problems occurred during testing and evaluation.

4.8.1 Internal Reader

The internal NFC chip from smartphones can be used to act as the initiator or target. This system architecture was evaluated with the Nexus 4 as the initiator part and the Proxmark3 and Raspberry Pi as the back-end which is currently used in the evaluation framework. The intention was to have an initiator which is combining the user interface and the NFC communication in one device and with the Android API it would be runnable on various smartphones. Thus such an environment can be copied effortless and has a potential outcome in terms of power optimizations for NFC use-case applications.

So the first step to connect the Nexus 4 as the initiator with the Proxmark3 gateway. The NFC connection between these two devices worked rarely and the emulated tag from the Proxmark3 was continuously dropped (`HALT` command) by the Nexus 4. Also other tests with the Nexus S as the initiator failed with similar results. It may has to do with the Proxmark3 antenna which is not optimal designed for target mode (transponder emulation) but rather than initiator mode. Also the Android operating system handles the tag internally and if a standard tag technology is connected the system is reading the tag. So the Proxmark3 has to determine if the request is coming from the operating system itself or from the running application.

The measurement of power drain was not simple possible with the Nexus 4 because the

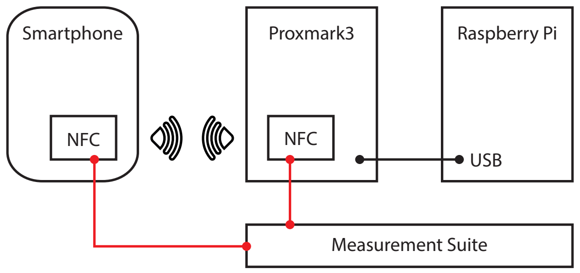


Figure 4.11: Approach with the smartphones internal NFC reader

chassis has to be opened to measure the power consumption of the NFC module from the internal battery. Furthermore, the measurement would not be accurate because all other internal hardware components of the smartphone are connected to the battery and would falsify the measurement results. The real power drain from the NFC module has to be estimated from the measured data. Only a full disassemble and a direct measurement of the NFC chip may have leaded to helpful consumption measurement. Figure 4.11 shows an example setup for a basic evaluation system with the internal reader.

Advantages:

- *Cost efficiency*: No extra reader hardware for the initiator is required for the NFC communication.
- *Android API*: The use-case applications can be implemented without any dependencies (no reader bridge).
- *Various devices*: The initiator can be any kind of Android hardware that has an internal NFC chip. (e.g. tablets, desktop computer)

Disadvantages:

- *Complicated measurement*: smartphone has to be opened to measure the power consumption.
- *Field strength*: The field strength of the internal module is heavily optimized for power consumption and so problems with the Proxmark3 communication may occur.

4.8.2 Breakout Board

Another approach with a very similar design to the actual system is the replacement of the Proxmark3 with the NXP PN532 NFC/RFID controller breakout board from Adafruit[18]. This board is a NFC extension for the Raspberry Pi and can be connected over the GPIO pins on the motherboard. The major advantage over the Proxmark3 extension is the direct control of the PN532 NFC chip so there is no firmware needed for the breakout board. Thus the Raspberry Pi handles the connection directly. The PN523 can be programmed

[18]`http://www.adafruit.com/products/364`

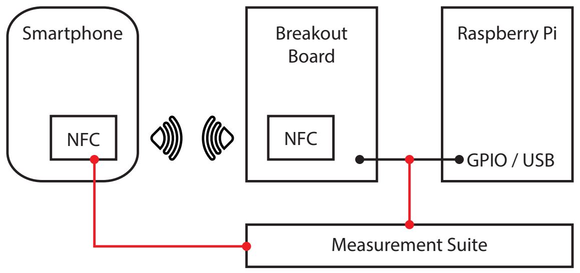


Figure 4.12: Approach with a NFC breakout board

with Libnfc [19] an open-source low-level NFC library. A tutorial how the breakout board can be connected to the Raspberry Pi and how the library is used can be found in [59]. This approach was evaluated and failed for the same reason because the emulated tag from the breakout board was dropped by the Android smartphone. But this approach may work with the external NFC reader extension in the current system (untested). Figure 4.11 shows an example setup for the targeted system with the Adafruit breakout board controller.

Advantages:

- *Cost efficiency*: The breakout board is relatively low priced compared to the Proxmark3 hardware.
- *Open-source libraries*: With use of the open-source Libnfc library the breakout board is able to emulate a broad bandwidth of transponders.
- *Single software component*: No firmware is needed for the breakout board and so the development of the use-case implementation can be achieved directly with the library.
- *Measurement friendly*: With the connection over the GPIO pins the measurement of the power consumption is accurate and effortless.
- *Operating System independent*: The breakout board can be connected over UART, I2C or SPI to Linux, Windows and Mac computers.

Disadvantages:

- *Proprietary*: NFC connection is established with the proprietary NXP PN532 chip and also the hardware design is not available for the breakout board.

4.8.3 Dual Gateway

An alternative could be a dual gateway system with two connected Proxmark3 devices. Thus the smartphones external NFC reader is a Proxmark3 which forwards the requests from the mobile phone to the Proxmark3 target gateway. The advantage of this system is

[19]`http://nfc-tools.org/index.php?title=Libnfc`

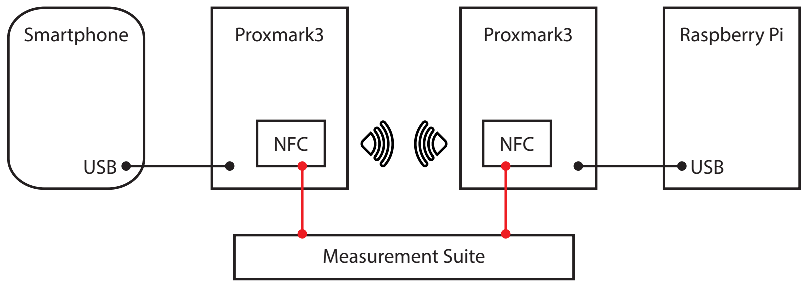


Figure 4.13: Dual gateway approach

the development of the gateway firmware which can be either executed in reader mode or otherwise in target mode. On Android side a client for the Proxmark3 is needed to invoke the reader mode and provides an interface for the use-case application. This system was basically only a concept and is untested in the current stage but it is a real open-source based solution without a proprietary NFC reader extension.

Advantages:

- *Non-proprietary system*: The used hardware for such a system is open-sourced in terms of hardware and software.
- *Free programmable*: RFID/NFC applications can be developed and evaluated on this system.
- *FPGA modifications*: Additional features can be added later without a redesign of the underlaying hardware.

Disadvantages:

- *Expensive hardware*: The Proxmark3 hardware is relatively expensive compared to other NFC extensions if it is purchased pre-assembled.
- *Rarely active development community*: The software for the Proxmark3 offers various features but the development is stagnating currently.
- *Lack of documentation*: The source-code documentation is nearly non-existent so it may be hard to understand how the firmware works.

Chapter 5

Case Study

This chapter covers the implementation of specific use-cases to demonstrate the capability of the proposed and implemented evaluation framework. This demonstration should prove that the run-time environment and measurement suite are working as expected within the predefined conditions. The specific applications will be evaluated with use of the attached automated NFC measurement suite and defined test-cases. The acquired power measurement data will be analyzed and presented in this chapter. Also the performed tests are designed to maintain repeatable and comprehensible. The results will be taken directly from the attached software suite and this description will also include and express the errors of measurement that occurred during the testing-phase. The case-study examples will be implemented and evaluated without any power optimizations. After the analysis of the power consumption, optimization techniques will be applied to the application and a further power evaluation iteration should illustrate the decreased power consumption.

5.1 Measurement Setup

The measurement setup for the evaluation of the use-case applications is based on the run-time environment, the Android development board and the measurement machine. Also the Android use-case application has to be extended for the suite so that the measurement controller invokes methods on Android side and measures the power consumption from the attached NFC reader. Figure 5.1 illustrates the actual hardware setup that was used during the use-case evaluation. Figure 5.2 shows a snapshot of the measurement setup (without the measurement machine, the I/O connector block and the development machine).

Voltage and current values are captured on the measurement machine with help of the I/O connectors. The USB connection from the development machine to the Android development board is used for observation of the executed use-case application and debugging and the LAN connection from the measurement machine to the Android development board is used for the socket connection between reader bridge and interface (direct connection for decreased delay) and the invocation of methods that should be observed in terms of power

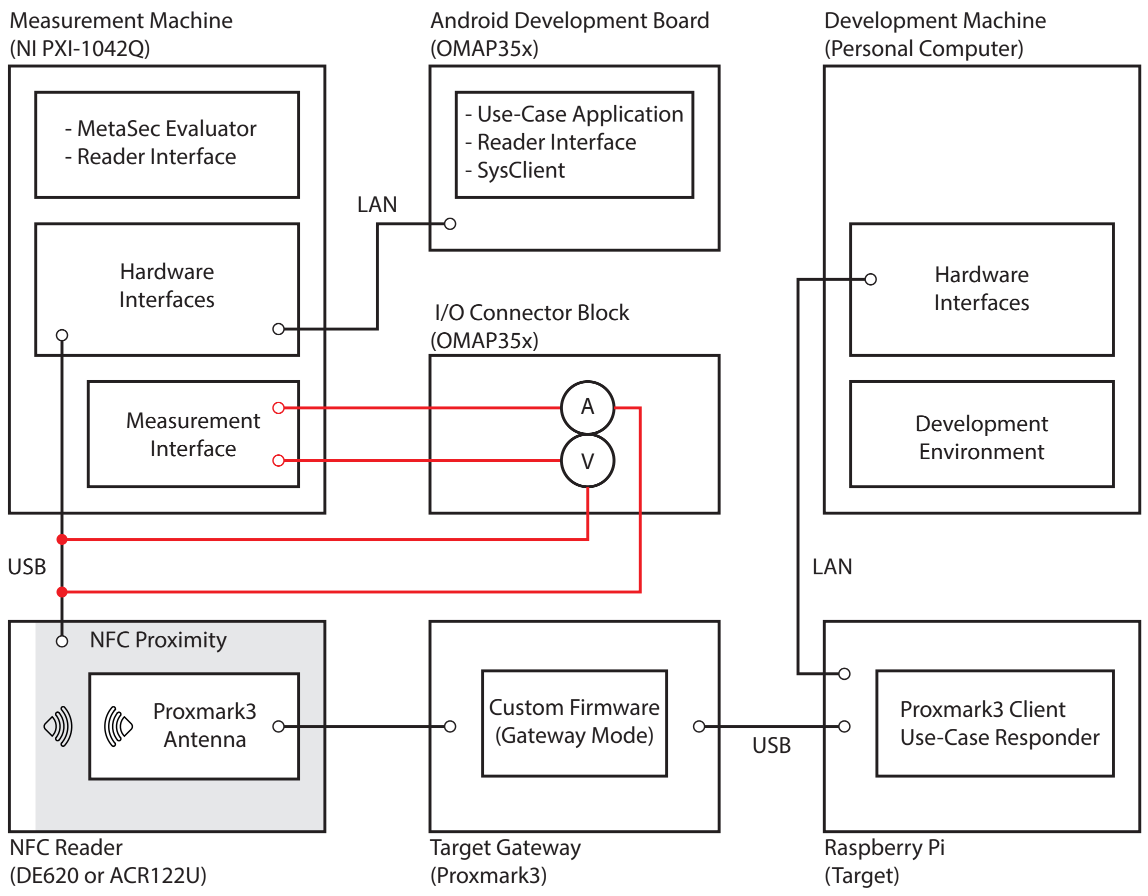


Figure 5.1: Measurement setup for the use-case study

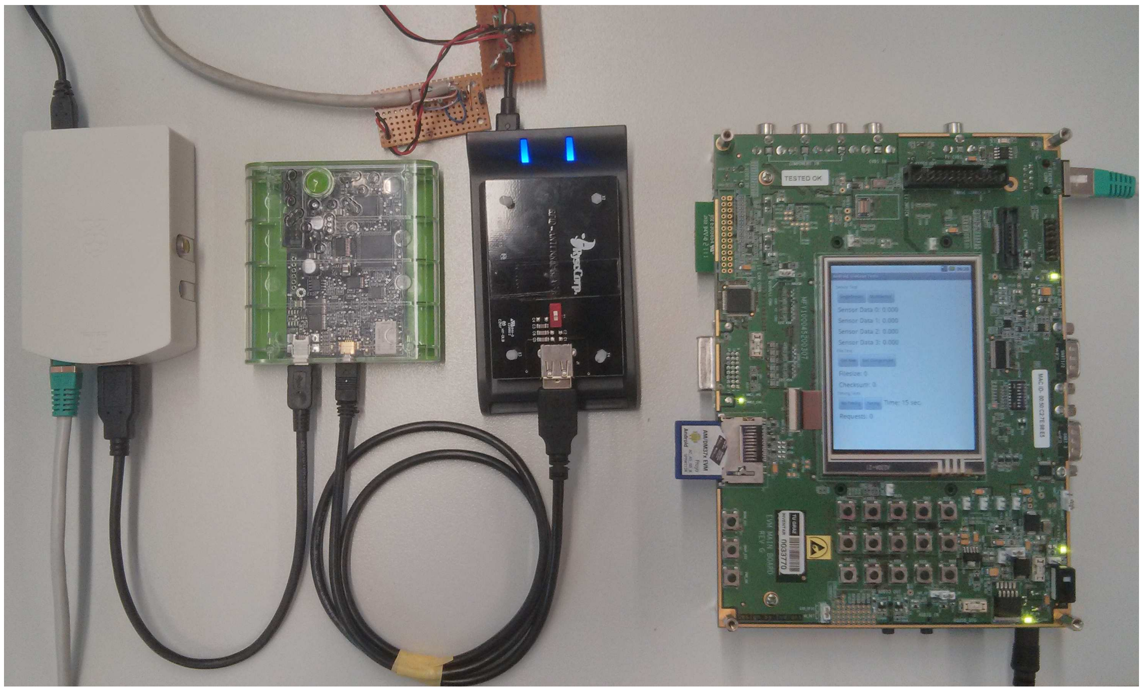


Figure 5.2: Measurement setup snapshot

consumption. Furthermore, the development machine and the Raspberry Pi are connected over LAN for SSH management.

One part of the measurement suite is called META[:SEC:] Evaluator (version 4) and is the primary software that is executed on the measurement machine and provides a simple object access protocol (SOAP) API over the LAN connection for the Android SysClient which is the secondary part. This SysClient controls the use-case application and invokes methods that should be evaluated and optimized.

5.1.1 Software Setup

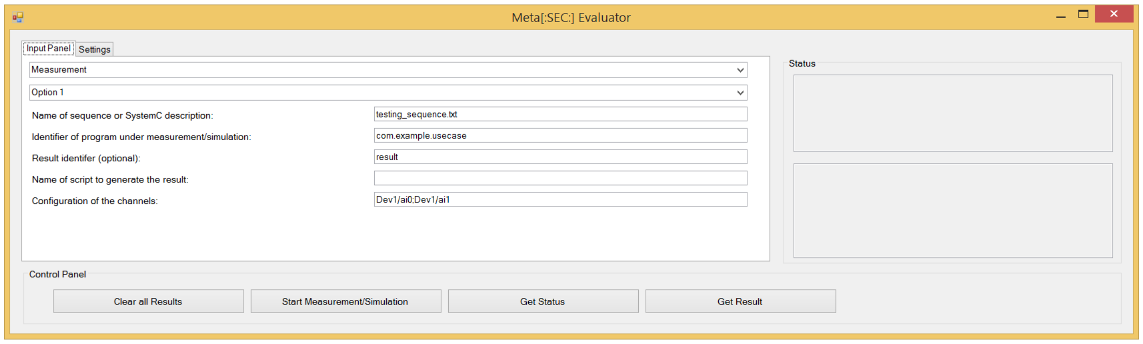
The measurement suite is mainly a method based testing tool that executes methods from the Android use-case application therefore the SysClient has to be integrated for testing purpose. The SysClient requires the IP address of the development machine for the connection to the SOAP API. The next step is the creation of an evaluation schedule which is a configuration file for the META[:SEC:] Evaluator and includes the sequence of the measured methods. The following list describes the steps for the software setup:

1. Implemented Android use-case application and responder for the Proxmark3 client.
2. Integration of the SysClient in the Android use-case application.
3. Configuration of the SysClient (IP address of the development machine).
4. Measurement sequence for the META[:SEC:] Evaluator as a configuration file.

5.2 Measurement Process

When the setup of the measurement is finished the application is ready for the evaluation cycle. The following list shows a step by step guide on how to initiate the implemented software components and starts the measurement process:

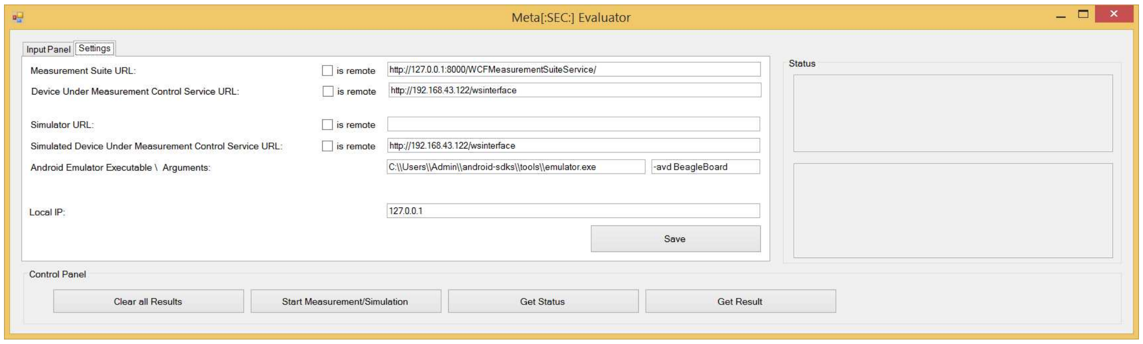
1. Raspberry Pi (Target)
   (a) Deploy the Java JAR file of the Proxmark3 client.
   (b) Start the Proxmark3 client with the desired responder.
2. Measurement Machine (NI PXI-1042Q)
   (a) Deploy the Java JAR file (reader interface).
   (b) Start the reader interface (32bit) with the module of the attached reader.
   (c) Start the META[:SEC:] Evaluator 4 suite.
   (d) Specify the `sequence file` in the input panel of the evaluator.
   (e) Specify the `package name` of the use-case application so that the SysClient knows which application to test.
   (f) Optionally specify the `result identifier` which is the filename of the generated Matlab data.
3. Android Development Board (OMAP35x)
   (a) Deploy the Android use-case application.
   (b) Deploy the SysClient application.
   (c) Start the Android use-case application.
   (d) Start the SysClient application (service will run in background).
4. Measurement Machine (NI PXI-1042Q)
   (a) A command prompt should indicate that the SysClient from the smartphone has connected.
   (b) Start the measurement by pressing `Start Measurement/Simulation`.
   (c) The suite now executes all methods from the `result identifier` on the smartphone.
   (d) Wait until the sequence has finished with all methods (indicated in the measurement command prompt).
   (e) Get the result data from the suite by pressing `Get Result`.
   (f) A new file should be added in the root directory of the META[:SEC:] Evaluator with the name of the `result identifier`.



Measurement
Option 1

Name of sequence or SystemC description:	testing_sequence.txt
Identifier of program under measurement/simulation:	com.example.usecase
Result identifer (optional):	result
Name of script to generate the result:	
Configuration of the channels:	Dev1/ai0;Dev1/ai1

Figure 5.3: META[:SEC:] Evaluator Input Panel for the main configuration



Measurement Suite URL:	☐ is remote	http://127.0.0.1:8000/WCFMeasurementSuiteService/	
Device Under Measurement Control Service URL:	☐ is remote	http://192.168.43.122/wsinterface	
Simulator URL:	☐ is remote		
Simulated Device Under Measurement Control Service URL:	☐ is remote	http://192.168.43.122/wsinterface	
Android Emulator Executable \ Arguments:		C:\\Users\\Admin\\android-sdks\\tools\\emulator.exe	-avd BeagleBoard
Local IP:		127.0.0.1	

Figure 5.4: META[:SEC:] Evaluator Settings for a remote SOAP web-service

Figure 5.3 and 5.4 show the META[:SEC:] Evaluator software that is used to perform the evaluation process. Figure 5.5 depicts the processing of the measurement suite.

5.3 Use Case Evaluation

The gathered and measured values for voltage (U) and current (I) are multiplied to get the power consumption (P) (see equation 5.1). The mean power consumption (P) and the execution time (t) are multiplied to get the energy consumption (E) (see equation 5.2).

$$P = U \cdot I \tag{5.1}$$

$$E = P \cdot t \tag{5.2}$$

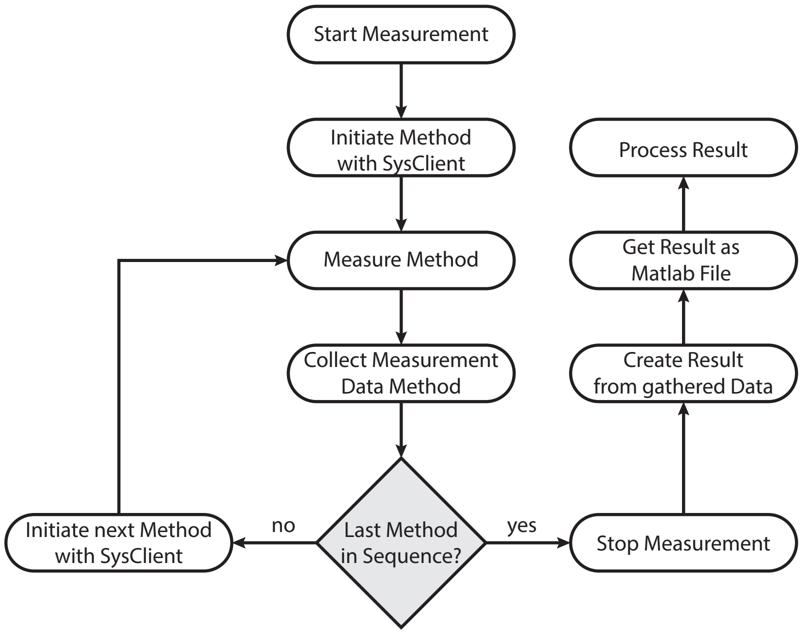
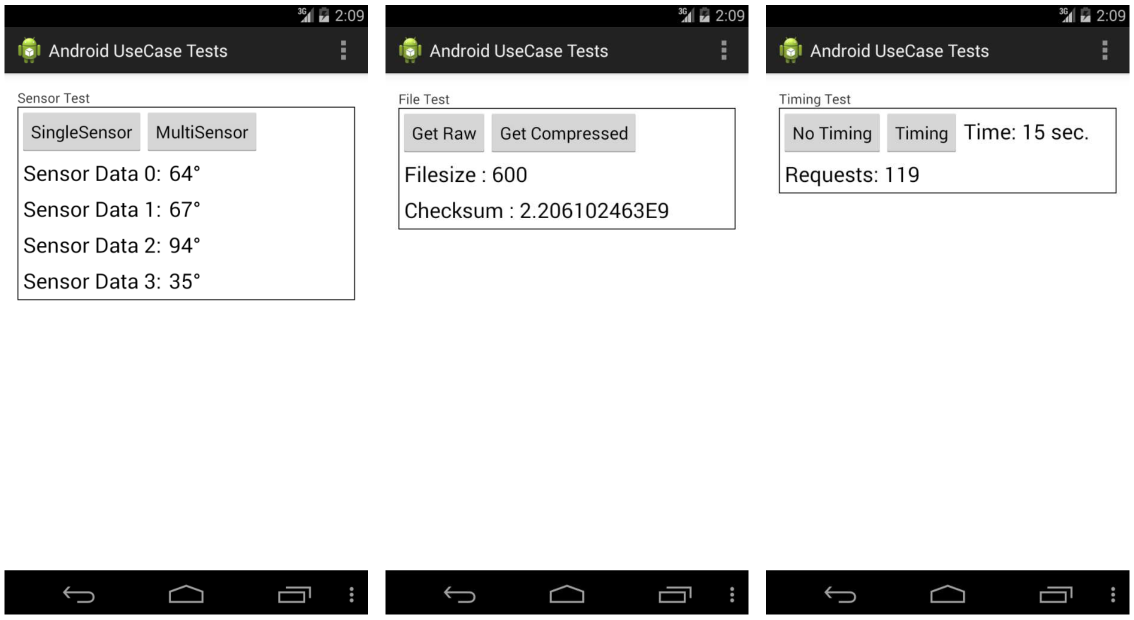

Start Measurement
Initiate Method with SysClient
Measure Method
Collect Measurement Data Method
Last Method in Sequence?
no
Initiate next Method with SysClient
yes
Stop Measurement
Create Result from gathered Data
Get Result as Matlab File
Process Result

Figure 5.5: Measurement process of the evaluation suite



Sensor Test

Sensor Data 0: 64°

Sensor Data 1: 67°

Sensor Data 2: 94°

Sensor Data 3: 35°

File Test

Filesize : 600

Checksum : 2.206102463E9

Timing Test

Time: 15 sec.

Requests: 119

(a) Sensor application (b) Data application (c) Timing application

Figure 5.6: Evaluated use-case applications

5.4 Example: Remote Sensor

The first use-case application is a peripheral sensor device attached to the Raspberry Pi similar to a smartmeter system. The Android application is reading the value of the sensor device and displays it to the user. Usually the initiator is moved towards the target and reads the data in close proximity. In this case the data has to be acquired manually. So the smartphone provides a user-interface to request the sensor data over the NFC connection. Figure 5.6a shows the Android use-case application for the sensor values. The first test only includes `Sensor Data 0` as label for the gathered data. The sensor data is displayed as temperature value and randomly generated on the target (Raspberry Pi). So the sensor data that is collected over the NFC interface is a generated value and does not come from a hardware sensor unit.

5.4.1 Proximity Range

The first test setup should prove that the measurement is working as expected and therefore the distance between the external reader and the Proxmark3 antenna was increased. The power consumption should increase accordingly to the distance. The Android use-case application is requiring the sensor value with a custom request with 1 Byte length and the response is encoded into a Java float (4 Bytes) with 2 Bytes CRC. To reduce measuring inaccuracy the Android application is requiring 30 values in row from the target. The distance between the Proxmark3 antenna and the reader is set to 0 cm, 1 cm and 2 cm.

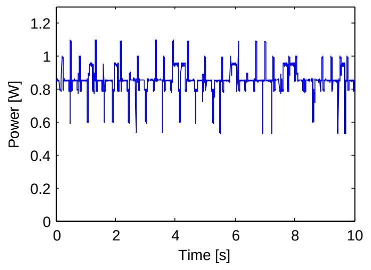
	Distance = 0 cm	Distance = 1 cm	Distance = 2 cm
Power Consumption (min)	0.527 W	0.522 W	0.531 W
Power Consumption (max)	1.099 W	1.123 W	1.147 W
Power Consumption (mean)	0.856 W	0.859 W	0.864 W
Execution Time	10.0 s	10.0 s	10.0 s
Energy Consumption	8.558 J	8.594 J	8.643 J

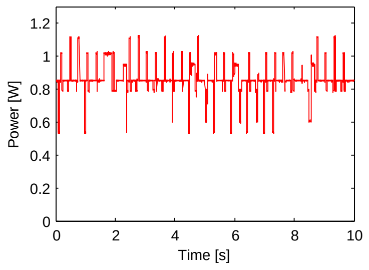
Table 5.1: Measurement: Proximity range

Figures 5.7a, 5.7b, and 5.7c show the power consumption during the request loop with different distances. The gathered measurement data in table 5.1 shows that the reader is consuming about 0.856 W on average with zero distance to the target. The average power consumption is slightly increasing with the transfer distance between the NFC components. So this very simple test proves that the increasing power consumption is measurable with the evaluation framework.

5.4.2 Multiple Sensors

In the last testing sequence a single sensor value was transfered from the sensor device (target) to the smartphone (initiator). Now there are multiple sensors connected which


(a) Distance = 0 cm


(b) Distance = 1 cm

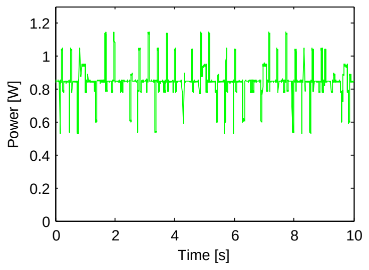
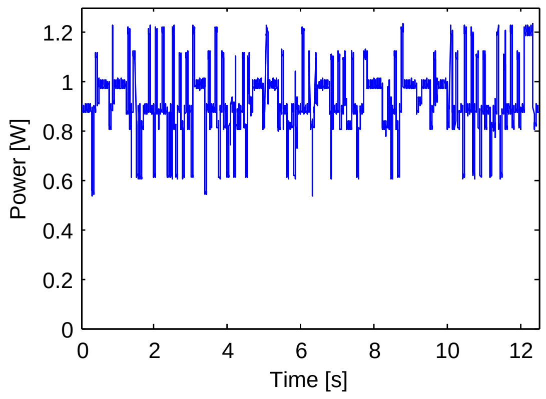

(c) Distance = 2 cm

Figure 5.7: Single sensor requests

should be readout over the NFC connection. This measurement should show how clustering of data reduces the power consumption. So therefore the first test is requesting every single sensor separately in every iteration whereas the second test is concatenating the sensor values so that just one request is used to transfer the desired sensor data. For this test four temperature sensors are simulated on the Raspberry Pi. The smartphone updates the sensor values 10 times so with the first test there are $10 * 4 = 40$ requests that are send and with the second test only 10 because one request returns the data of all sensor devices.

	Single Sensor	Clustered Sensor
Power Consumption (min)	0.536 W	0.606 W
Power Consumption (max)	1.233 W	1.225 W
Power Consumption (mean)	0.917 W	0.898 W
Execution Time	12.5 s	2.1 s
Energy Consumption	11.468 J	1.885 J

Table 5.2: Measurement: Multiple sensors


(a) Single sensor requests

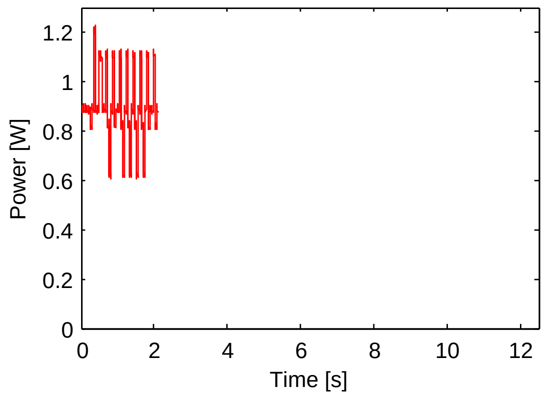

(b) Clustered sensor requests

Figure 5.8: Single and clustered requests

Figures 5.8a and 5.8b illustrate the power consumption during the requests. The measured values in table 5.2 show that the average power consumption for the reader is between 0.917 W and 0.898 W (nearly equal) but the execution time for the clustered sensor example is significantly shorter which reduces the energy consumption about $1 - \frac{1.885\text{J}}{11.468\text{J}} = 1 - 0.164 = 83.6\%$. This is an evident example and the results are as expected. The user does not notice the difference how the data is acquired and therefore such a clustering mechanic is a common optimization technique.

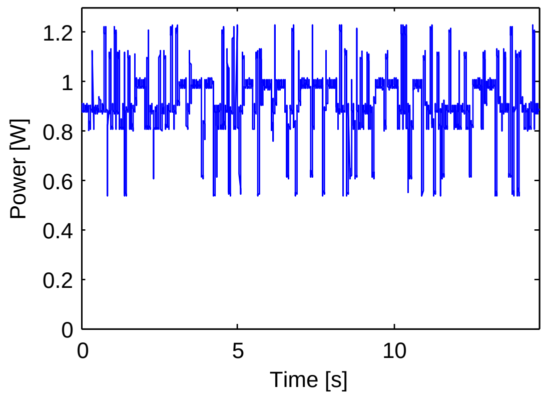
5.5 Example: Large Data Exchange

The second use-case application example is the transfer of a large data set and how compression of such data reduces the power consumption during communication. The target device generates random data with a predefined length. This raw data is transfered in chunks to the smartphone that is reassembling the original data. In the second step the data is compressed with the popular zlib library[1]. With compression fewer requests for the data transfer are needed and thus the power consumption is decreased. A CRC32 checksum should validate the file and expose transmission errors that may occurred during data transfer. The transfer of the data is double checked because every single response appends a CRC16 (2 Byte) checksum. For this use-case experiment the data size is set to 600 Bytes and the transfered chunk-size is 20 Bytes.

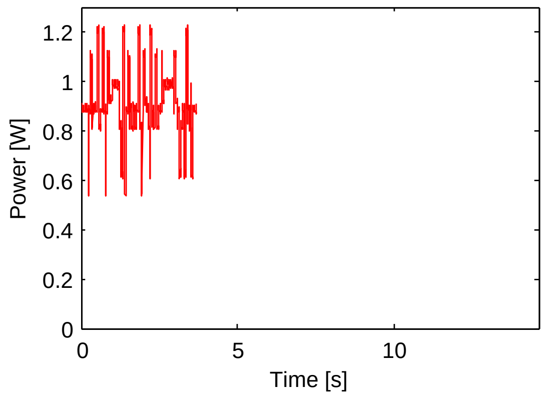
	Uncompressed Transfer	Compressed Transfer
Power Consumption (min)	0.534 W	0.532 W
Power Consumption (max)	1.230 W	1.228 W
Power Consumption (mean)	0.913 W	0.902 W
Execution Time	14.6 s	3.7 s
Energy Consumption	13.336 J	3.337 J

Table 5.3: Measurement: Large data set

Figures 5.9a and 5.9b are showing the power consumption during the requests. Measurement values can be seen in table 5.3 which are showing that the transfer of the compressed data takes about $1 - \frac{3.7\text{s}}{14.6\text{s}} = 1 - 0.253 = 74.7\%$ less time. Therefore the energy consumption is also reduced by almost the same factor because the average consumption is nearly equal again like in the last use-case test example.



(a) Raw data



(b) Compressed Data

Figure 5.9: Large data set transfer

[1]`http://www.zlib.net/`

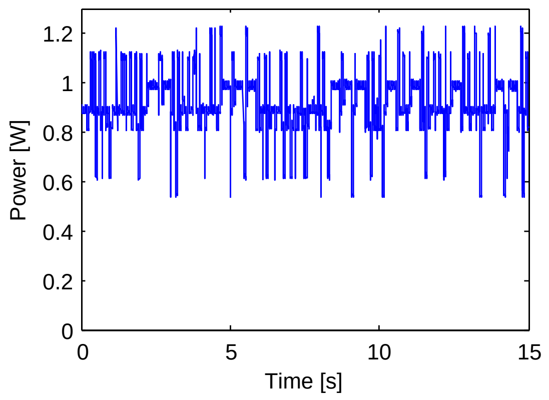
5.6 Example: Request Timing

This use-case application example demonstrates how timeouts and timing in general can reduce power consumption. It is a similar test like the previously discussed sensor example. A single value (data) is requested from the target device (in this case a single Byte) and displayed on the smartphone. An additional Byte for the timeout is also appended which indicates the data (single Byte value) is expected to be changed. In the first test the smartphone permanently requests the value from the Raspberry Pi and ignores the timeout for the next request. The target is requesting the same value over and over again until it changes after the timeout. The second test uses the timeout and waits with the next request until the data has expired. The timeout for this example is randomly generated between 0 and 1500ms and the the test is running for 15s.

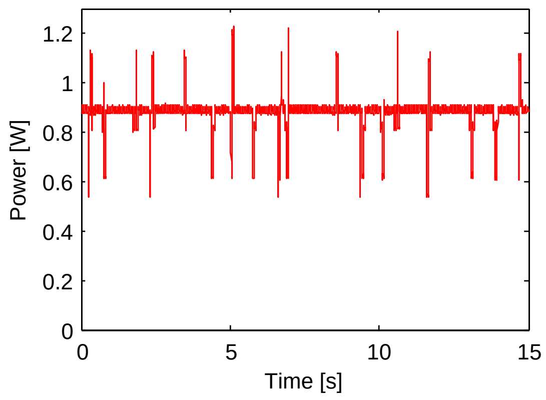
	No Timeout	With Timeout
Power Consumption (min)	0.533 W	0.534 W
Power Consumption (max)	1.228 W	1.230 W
Power Consumption (mean)	0.921 W	0.873 W
Execution Time	15.0 s	15.0 s
Energy Consumption	13.815 J	13.099 J

Table 5.4: Measurement: Timed data requests

Figures 5.10a and 5.10b show the power consumption during the requests. The variance of the average power consumption between the two test samples is not that high (average power consumption: 0.921 W (no timing) versus 0.873 W (with timing) see table 5.4). Nevertheless the timed requests are requiring less energy compared to the test that is ignoring the timeout completely. A more advanced and better optimization example would be turning off the magnetic field of the reader during the timeout which would drastically lower the energy consumption for the second test sample.



(a) No timeout



(b) With timeout

Figure 5.10: Timed data requests

Chapter 6

Conclusion

The following conclusion points out the advantages and disadvantages of the evaluation framework. Furthermore, which concepts or designs are satisfying and have a high potential to support developers in their productive implementation work. But also deficiency of the systems will be identified and highlighted and the conclusion also includes a critical view on such an evaluation framework. The contrast between positive and negative findings will be argued in avoidance of implementation faults that might happen in the development progress. The initial concept of an evaluation framework for NFC bridge systems was designed and implemented in this thesis.

The interoperability of NFC opens the way to a variety of application areas and thus an evaluation framework has a justified existence. The proposed and implemented evaluation framework should demonstrate how such a system can be created and support researchers and developers to create and build their own NFC applications on top of the fundamental run-time environment. As addition to the basic development foundation based on NFC bridges a measurement suite can help to evaluate and optimize applications that are executed on this system. The underlying architecture of this evaluation platform fulfills the requirements to evaluate a variety of novel NFC systems like the NFC bridge. The proposed design does not cover all evaluation aspects but provides a new baseline for NFC evaluation frameworks and can be adapted to cover special requirements for the developed applications that are evaluated on the platform afterwards. An example would be the extension with various measurement utilities for the fundamental run-time environment in order to evaluate different measurable values (e.g. magnetic field strength).

The proof of concept was shown in the case study especially with the evaluation of the use-case application examples. These examples were kept simple to demonstrate the capability of the framework by going through the optimization process for certain use-cases. More advanced power optimizations techniques (software and hardware) are described in the related work. Finally it needs to be mentioned that the actual implementation is working as expected and can be used to evaluate and optimize the power consumption of NFC bridge systems.

Chapter 7

Future Work

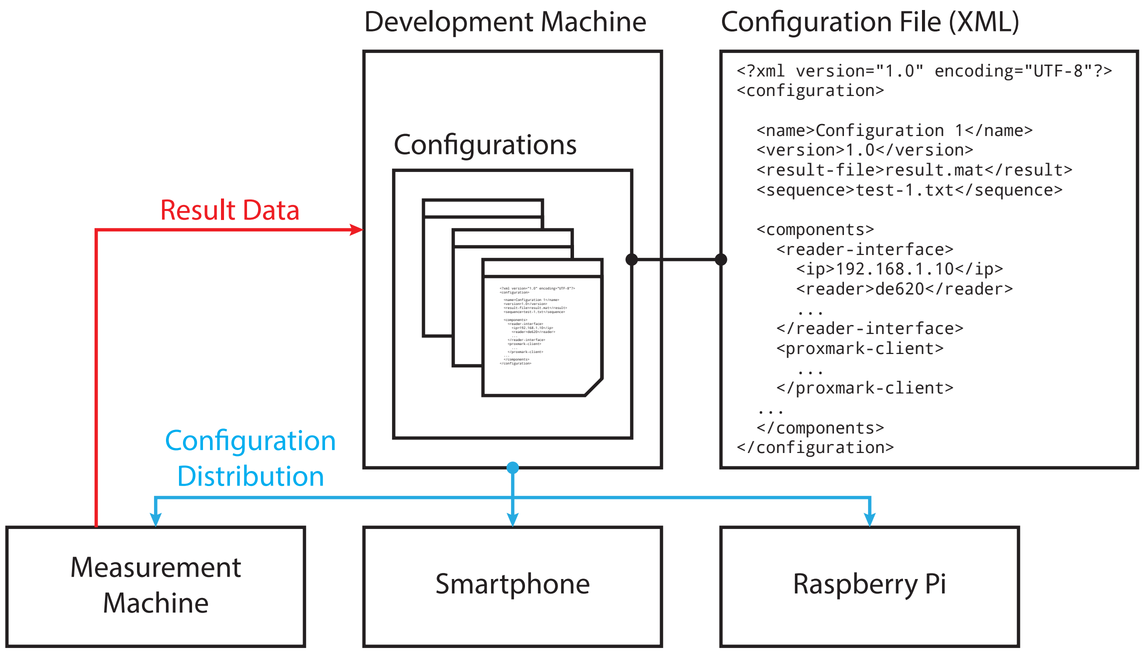
In this chapter potential new features and ideas are discussed in order to improve the evaluation framework of this thesis. This chapter also contains concepts that appeared mostly during implementation phase and evaluation of results, but are beyond the scope of the actual work of this project.

Tag Technology and Modes: The current implementation supports *NfcA* technology (ISO 14443-3A). The Raspberry Pi with the attached gateway is able to emulate tag types that are covered by this specification. Other tag technologies can be implemented additionally and extend the evaluation framework for new use-case applications. The introduction of NFC data exchange format (NDEF) messages is also an option that can be developed, because it is the standard data encoding for information transfer between NFC devices.

The smartphone as initiator is in reader-writer mode and the Raspberry Pi with the Proxmark3 gateway as target is in card-emulation mode. The peer-to-peer mode is not part of the current evaluation system and can be implemented in a further project. The Android API already supports the peer-to-peer mode (internal NFC chip) which can be utilized as foundation for the development of such an environment.

Reversed Communication: Another idea for the extension of the evaluation framework is to reverse the NFC communication direction. The smartphone act as the target and the Raspberry Pi as the initiator. This system can be applied where the initiator is a stationary device that is connected to the power network. Thus the power optimization of the smartphone is the primary task. Android 4.4 introduced the host-based card emulation[1] that turns the mobile phone in a smartcard which communicates with a standard NFC reader. Also the secure element can be accessed with this operation mode and used for various new use-case scenarios.

[1] `https://developer.android.com/guide/topics/connectivity/nfc/hce.html`



```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>

  <name>Configuration 1</name>
  <version>1.0</version>
  <result-file>result.mat</result>
  <sequence>test-1.txt</sequence>

  <components>
    <reader-interface>
      <ip>192.168.1.10</ip>
      <reader>de620</reader>
      ...
    </reader-interface>
    <proxmark-client>
      ...
    </proxmark-client>
  ...
  </components>
</configuration>
```

Figure 7.1: Central configuration example

Central Configuration and Execution: A central configuration and execution approach would also improve the usability of the current evaluation system. The configuration can for example include the measurement sequence, IP addresses, result identifier, use-case application specific data, mode for the reader interface (simulator, DE620, ACR120U), responder module, etc. which can be distributed to the devices that are executing the software components and are initialized with the parameters from the configuration file. The advantage of this system is a faster development process and a simple switching between different measurement configurations (e.g. responder modules). Currently the software components have to be configured and executed manually. Figure 7.1 shows a concept of a central configuration and execution from a central development unit.

Remote Deployment and Measurement: Also the deployment of the software components from the development machine to the devices that are part of the actual evaluation framework may be realized in a future extension. So the targeted objective for this idea is a central development environment which automatically generate the JAR files and distribute it to the hardware devices with the central configuration. Furthermore, this environment is able to start and stop the software components and is observing the whole execution process with enhanced logging mechanics.

The design of this development environment can be realized as a remote measurement system in which clients are able to connect and upload applications (plus configuration). Furthermore, the system evaluates the power consumption by starting the measurement process. The clients share the same evaluation framework and are not requiring a local hardware environment.

Framework Simulation Extension: The simulator of the reader interface is very similar to the custom responder class of the Proxmark3 client. The actual implementation can be refactored in order to use the exact same classes for both software components. The responder can then be developed straight-forward as a simulator for the reader interface and afterwards simply copied to the Proxmark3 client.

Power Analysis Extension of Code Segments: An advanced extension to the current system is a detection mechanism that is analyzing the executed code and automatically estimate which code segments are potential candidates for optimization modifications. Thus the measurement suite is able to measure the whole application without a measurement sequence and only with use of the source-code that is running either on the smartphone or the Raspberry Pi.

Appendix A

Acronyms

ADB	android debug bridge
AFSS	adaptive field strength scaling
APDU	application protocol data unit
API	application programming interface
APS	automatic power stepping
ASK	amplitude shift keying
AVD	android virtual device
CPU	central processing unit
CRC	cyclic redundancy check
CU	computational unit
DAM	data acquisition module
DLL	dynamic link library
DVFS	dynamic voltage and frequency scaling
DVM	dalvik virtual machine
FOSS	free and open-source software
FPGA	field programmable gate array
FSK	frequency shift keying
FU	functional unit
GCC	GNU compiler collection
GPIO	general purpose input/output

GPRS	general packet radio service
GPU	graphics processing unit
GUI	graphical user interface
HAXM	hardware accelerated execution manager
HF	high frequency (3-30 MHz)
HIL	hardware in the loop
IDE	integrated development environment
IP	internet protocol
JAR	java archive
JDK	Java development kit
JNA	java native access
JRE	java run-time environment
JSON	JavaScript object notation
LAN	local area network
LED	light emitting diode
LF	low frequency (30-300 kHz)
LLC	logical link control layer
MAC	media access control layer
NDEF	NFC data exchange format
NFC	near field communication
NIZE	near field communication interface enabling zero energy
OOK	on-off keying
OSI	open systems interconnection
OS	operating system
PAN	personal area networks
PC/SC	personal computer/smart card
PC	personal computer
PDA	personal digital assistant
PLC	programmable logic controller
PPF	power profile flattening

PSK	phase shift keying
PTF	power transfer function
RAM	random access memory
RFID	radio frequency identification
RF	radio frequency
SDHC	standard high capacity
SDK	software development kit
SIL	software in the loop
SLF4J	simple logging facade for Java
SMS	short message service
SOAP	simple object access protocol
SOC	state of charge
SSH	secure shell
SUT	system under test
TCP	transmission control protocol
TDD	test-driven development
UMTS	universal mobile telecommunications system
USB	universal serial bus
WLAN	wireless local area network
XML	extensible mark-up language

Appendix B

Development Setup

B.1 Eclipse and Android

Installation guide for the Android SDK and the Eclipse integration can be found in the Android documentation [1]. The software components for the evaluation framework can be simply imported to Eclipse (`File → Import → Existing Projects into Workspace`). The reader interface requires a 32bit Java environment because of the DLL library from the DE620 NFC reader.

B.2 Proxmark3 Firmware

The Proxmark3 firmware compilation and flashing process requires some addition development tools. Windows/Linux users can install the system with help of the Proxmark3 online guide:

- Windows: `https://code.google.com/p/proxmark3/wiki/Windows`
- Linux: `https://code.google.com/p/proxmark3/wiki/Linux`

The next step is the adaption of the used port for the serial connection in the Makefile accordingly to the system (Linux: `FLASH_PORT=/dev/ttyACM0`, Windows: `FLASH_PORT=COM5` port numbers vary). After the setup of the build tools the modified firmware can be compiled in this environment and flashed to the connected Proxmark3.

Compiling of all Proxmark3 software components:

```
$ make all
```

Flashing of the firmware (os):

```
$ make flash-os
```

[1] `http://developer.android.com/sdk/index.html`

Flashing of the bootrom (only if required):

```
$ make flash-bootrom
```

Flashing of the FPGA (only if required):

```
$ make flash-fpga
```

B.3 Raspberry Pi

The Arch Linux installation on the Raspberry Pi can be achieved with the guide [2] from Arch Linux ARM. Afterwards, the Proxmark3 driver has to be installed on the operating system in order to work with the Proxmark3 client software.

With use of `SSH` (Terminal or Putty) the JAR file for the Proxmark3 client software can be send to the Raspberry Pi:

```
$ scp client.jar user@host:client.jar
```

For the execution of the JAR file on the Raspberry Pi install the open Java development kit (JDK) environment over `SSH`:

```
$ pacman -S jre7-openjdk
```

[2] http://archlinuxarm.org/platforms/armv6/raspberry-pi

Bibliography

[1] NFC Forum. "Essentials for Successful NFC Mobile Ecosystems". In: (2008) (cit. on p. 11).

[2] Josef Langer and Michael Roland. *Anwendungen und Technik von Near Field Communication (NFC)*. Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-05496-9. DOI: 10.1007/978-3-642-05497-6 (cit. on pp. 16–18, 20).

[3] NFC Forum. "Type 1 Tag Operation Specification. Technical Specification". In: (2011). Ed. by NFC Forum, p. 47 (cit. on p. 19).

[4] NFC Forum. "Type 2 Tag Operation Specification. Technical Specification". In: (2011). Ed. by NFC Forum, p. 53 (cit. on p. 19).

[5] NFC Forum. "Type 3 Tag Operation Specification. Technical Specification". In: (2011). Ed. by NFC Forum, p. 33 (cit. on p. 19).

[6] Thomas Schmidt. "CRC Generating and Checking". In: *Microchip Technology Inc.* (2000) (cit. on p. 20).

[7] NFC Forum. "NFC Digital Protocol (DIGITAL 1.0). Technical Specification". In: (2010). Ed. by NFC Forum, p. 194 (cit. on p. 20).

[8] Esko Strömmer, Mika Hillukkala, and Arto Ylisaukko-oja. "Ultra-low Power Sensors with Near Field Communication for Mobile Applications". English. In: *Wireless Sensor and Actor Networks*. Ed. by Luis Orozco-Barbosa, Teresa Olivares, Rafael Casado, and Aurelio Bermúdez. Vol. 248. IFIP International Federation for Information Processing. Springer US, 2007, pp. 131–142. ISBN: 978-0-387-74898-6. DOI: 10.1007/978-0-387-74899-3_12 (cit. on p. 20).

[9] Charl A. Opperman and Gerhard P. Hancke. "Using NFC-enabled phones for remote data acquisition and digital control". In: *AFRICON* (2011), pp. 1–6. ISSN: 21530025. DOI: 10.1109/AFRCON.2011.6072147 (cit. on pp. 21, 22).

[10] Shun-Yu Chan, Shang-Wen Luan, Jen-Hao Teng, and Ming-Chang Tsai. "Design and implementation of a RFID-based power meter and outage recording system". In: *ICSET 2008 IEEE International Conference on Sustainable Energy Technologies* (2008), pp. 750–754. DOI: 10.1109/ICSET.2008.4747106 (cit. on pp. 22, 23).

[11] Alan K. Meier. "A worldwide review of standby power use in homes". In: *Lawrence Berkeley National Laboratory* (2001), pp. 1–5 (cit. on p. 23).

[12] Norbert Druml, Manuel Menghin, Rejhan Basagic, Christian Steger, Reinhold Weiss, Holger Bock, and Josef Haid. “NIZE - a Near Field Communication interface enabling zero energy standby for everyday electronic devices”. In: *IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (Oct. 2012), pp. 261–267. DOI: `10.1109/WiMOB.2012.6379085` (cit. on pp. 23, 24).

[13] Manuel Menghin, Norbert Druml, Christian Steger, Reinhold Weiss, Holger Bock, and Josef Haid. “The PTF-Determinator: A Run-Time Method Used to Save Energy in NFC-Systems”. In: *2012 Fourth International EURASIP Workshop on RFID Technology* 1 (2012), pp. 92–98. DOI: `10.1109/RFID.2012.12` (cit. on p. 24).

[14] Xunteng Xu, Lin Gu, Jianping Wang, Guoliang Xing, and Shing-chi Cheung. “Read More with Less : An Adaptive Approach to Energy-Efficient RFID Systems”. In: *IEEE Journal on Selected Areas in Communications* 29.8 (2011), pp. 1684–1697. DOI: `10.1109/JSAC.2011.110917` (cit. on p. 24).

[15] Norbert Druml, Manuel Menghin, Christian Steger, Reinhold Weiss, Andreas Genser, Holger Bock, and Josef Haid. “Adaptive Field Strength Scaling: A Power Optimization Technique for Contactless Reader / Smart Card Systems”. In: *15th Euromicro Conference on Digital System Design* (Sept. 2012), pp. 616–623. DOI: `10.1109/DSD.2012.20` (cit. on p. 25).

[16] Manuel Menghin and Norbert Druml. “Energy efficiency by using field strength scaling for multi-transponder applications”. In: *12th International Conference on Telecommunications (ConTEL)* (2013), pp. 263–270 (cit. on pp. 25, 31).

[17] K Warnick, B Gottula, Sushant Shrestha, and James Smith. “Optimizing Power Transfer Efficiency and Bandwidth for Near Field Communication Systems”. In: *IEEE Transactions on Antennas and Propagation* 61.2 (2013), pp. 927–933. DOI: `10.1109/TAP.2012.2220325` (cit. on p. 26).

[18] William A. Pearlman and Amir Said. *Digital Signal Compression: Principles and Practice*. Digital Signal Compression: Principles and Practice. Cambridge University Press, 2011. ISBN: 9780521899826 (cit. on p. 26).

[19] Fangming Gu, WS Harrison, DM Tilbury, and Yuan Chengyin. “Hardware-in-the-loop for manufacturing automation control: Current status and identified needs”. In: *CASE 2007 IEEE International Conference on Automation Science and Engineering* (Sept. 2007), pp. 1105–1110. DOI: `10.1109/COASE.2007.4341787` (cit. on p. 28).

[20] M. Bacic. “On hardware-in-the-loop simulation”. In: *CDC-ECC 2005 44th IEEE Conference on Decision and Control* (2005). DOI: `10.1109/CDC.2005.1582653` (cit. on p. 28).

[21] A. Bouscayrol. “Different types of Hardware-In-the-Loop simulation for electric drives”. In: *IEEE International Symposium on Industrial Electronics* (June 2008), pp. 2146–2151. DOI: `10.1109/ISIE.2008.4677304` (cit. on p. 29).

[22] Santiago Lentijo, S D'Arco, and Antonello Monti. "Comparing the dynamic performances of power hardware-in-the-loop interfaces". In: *IEEE Transactions on Industrial Electronics* 57.4 (2010), pp. 1195–1207. DOI: `10.1109/TIE.2009.2027246` (cit. on p. 29).

[23] Yuheng Li, Zechang Sun, and Jiayuan Wang. "Design for battery management system hardware-in-loop test platform". In: *ICEMI 9th International Conference on Electronic Measurement and Instruments* (Aug. 2009), pages. DOI: `10.1109/ICEMI.2009.5274292` (cit. on p. 29).

[24] A Monti, S D'Arco, and A Deshmukh. "A new architecture for low cost power hardware in the loop testing of power electronics equipments". In: *ISIE 2008 IEEE International Symposium on Industrial Electronics* 29208 (2008), pp. 2183–2188. DOI: `10.1109/ISIE.2008.4677306` (cit. on p. 29).

[25] Uwe Ryssel, Joern Ploennigs, and Klaus Kabitzsch. "Generative design of hardware-in-the-loop models". In: *APGES 2007* (2007), pp. 1–8 (cit. on p. 29).

[26] R Casas and O Casas. "Battery sensing for energy-aware system design". In: *Computer* 38 (11 2005). DOI: `10.1109/MC.2005.367` (cit. on p. 32).

[27] Rao Ravishankar, Sarma Vrudhula, and Daler N. Rakhmatov. "Battery Modeling for Energy-Aware System Design". In: *Computer* 36 (12 2008). DOI: `10.1109/MC.2003.1250886` (cit. on p. 32).

[28] Josef Langer, Christian Saminger, and Stefan Grünberger. "A comprehensive concept and system for measurement and testing Near Field Communication devices". In: *2009 Eurocon IEEE* (2009), pp. 2052–2057. DOI: `10.1109/EURCON.2009.5167930` (cit. on p. 33).

[29] Charl A. Opperman and Gerhard P. Hancke. "A Generic NFC-enabled Measurement System for Remote Monitoring and Control of Client-side Equipment". In: *3rd International Workshop on Near Field Communication (NFC)* (Feb. 2011), pp. 44–49. DOI: `10.1109/NFC.2011.11` (cit. on pp. 33, 34).

[30] John R. Ackerman. "Toward open source hardware". In: *U. Dayton L. Rev.* (2008) (cit. on p. 34).

[31] S. Davidson. "Open-source hardware". In: *IEEE Design and Test of Computers* 21.5 (Sept. 2004), pp. 456–456. ISSN: 0740-7475. DOI: `10.1109/MDT.2004.68` (cit. on p. 34).

[32] Joshua M Pearce. "Materials science. Building research equipment with free, open-source hardware." In: *Science Magazine 14* 337.6100 (2012), pp. 1303–4. ISSN: 1095-9203. DOI: `10.1126/science.1228183` (cit. on p. 34).

[33] Statista. *Anzahl der Smartphone-Nutzer in Deutschland nach genutztem Betriebssystem im September 2013 (in Millionen)*. 2014. URL: `http://de.statista.com/statistik/daten/studie/176811/umfrage/verbreitung-mobiler-endgeraete-nach-betriebssystem-in-deutschland/` (cit. on p. 35).

[34] Google Inc. *LG E960 (Nexus 4) Technical Specification*. 2014. URL: `http://www.google.de/nexus/4/specs/` (visited on 05/22/2014) (cit. on p. 51).

[35] GSMArena. *LG E960 (Nexus 4) Technical Specification*. 2014. URL: http://www.gsmarena.com/lg_nexus_4_e960-5048.php (visited on 05/22/2014) (cit. on p. 51).

[36] NXP Semiconductors. *NXP's NFC solution implemented in Galaxy Nexus from Google*. 2011. URL: http://www.nxp.com/news/press-releases/2011/11/nxp-nfc-solution-implemented-in-galaxy-nexus-from-google.html (visited on 05/22/2014) (cit. on p. 52).

[37] NXP Semiconductors. *NXP NFC controller PN544 for mobile phones and portable equipment*. 2010. URL: http://www.nxp.com/documents/leaflet/75016890.pdf (visited on 05/22/2014) (cit. on p. 52).

[38] Google Inc. *GT-I9023 (Nexus S) Technical Specification*. 2010. URL: http://googleblog.blogspot.co.at/2010/12/introducing-nexus-s-with-gingerbread.html (visited on 05/22/2014) (cit. on p. 52).

[39] GSMArena. *GT-I9023 (Nexus S) Technical Specification*. 2014. URL: http://www.gsmarena.com/samsung_google_nexus_s_i9023-3910.php (visited on 05/22/2014) (cit. on p. 52).

[40] Intel Corporation. *Intel® Hardware Accelerated Execution Manager*. 2014. URL: https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager (visited on 05/22/2014) (cit. on p. 53).

[41] Google Inc. *Host-based Card Emulation Technical Description*. 2014. URL: http://developer.android.com/guide/topics/connectivity/nfc/hce.html (visited on 05/23/2014) (cit. on p. 53).

[42] Google Inc. *Android 4.4 APIs - API Level: 19*. 2014. URL: https://developer.android.com/about/versions/android-4.4.html (visited on 05/23/2014) (cit. on p. 53).

[43] DUALi Inc. *DUALi DE-620 USB Technical Specification*. 2014. URL: http://www.duali.com/upload/bbs/DE-620%20HID.pdf (visited on 05/22/2014) (cit. on p. 54).

[44] Advanced Card Systems Ltd. *ACR122U USB NFC Reader Technical Specification*. Version 3.02. 2014. URL: http://downloads.acs.com.hk/drivers/cn/TSP-ACR122U-3.02.pdf (visited on 05/22/2014) (cit. on p. 54).

[45] Judd Vinet and Aaron Griffin. *Arch Linux, a lightweight and flexible Linux distribution*. 2014. URL: https://www.archlinux.org/ (visited on 05/23/2014) (cit. on p. 55).

[46] Bernhard Trummer. *Raspberry Pi GPIO Pin Hacking*. 2014. URL: http://glt14-programm.linuxtage.at/events/250.de.html (visited on 05/23/2014) (cit. on p. 55).

[47] Alexander Langer. *Raspberry Pi Model B (rev. 2) Technical Specification*. 2012. URL: http://raspberrycenter.de/handbuch/technische-daten (visited on 05/22/2014) (cit. on p. 56).

[48] Raspberry PI Foundation. *Raspberry Pi Model B (rev. 2) Technical Specification*. 2014. URL: http://www.raspberrypi.org/documentation/hardware/raspberrypi/ (visited on 05/22/2014) (cit. on p. 56).

[49] Proxmark3 Community. *Proxmark3 General Description*. 2014. URL: `https://code.google.com/p/proxmark3/wiki/HomePage?tm=6` (visited on 05/23/2014) (cit. on p. 56).

[50] Proxmark3 Community. *Official Proxmark SourceCode Repository*. 2014. URL: `https://github.com/Proxmark/proxmark3` (visited on 05/23/2014) (cit. on p. 56).

[51] Atmel Corporation. *AT91SAM7S256 Microcontroller Technical Specification*. 2014. URL: `http://www.atmel.com/devices/sam7s256.aspx` (visited on 05/22/2014) (cit. on p. 57).

[52] Proxmark3 Community. *Proxmark3 Technical Description*. 2013. URL: `https://code.google.com/p/proxmark3/wiki/HardwareDescription` (visited on 05/22/2014) (cit. on p. 57).

[53] Xilinx Inc. *Spartan-II FPGA Family Data Sheet*. Version DS001-1 (v2.8). 2008. URL: `http://www.xilinx.com/support/documentation/data_sheets/ds001.pdf` (visited on 05/22/2014) (cit. on p. 57).

[54] Sourcemaking. *Design Patterns*. 2014. URL: `http://sourcemaking.com/design_patterns` (visited on 05/28/2014) (cit. on p. 63).

[55] JNA Development Team. *Java Native Access (JNA)*. 2014. URL: `https://github.com/twall/jna#readme` (visited on 05/28/2014) (cit. on p. 63).

[56] Eclipse Foundation. *Eclipse IDE Website*. 2014. URL: `http://www.eclipse.org/` (visited on 05/28/2014) (cit. on p. 63).

[57] Apache Software Foundation. *Apache Ant Website*. 2014. URL: `http://ant.apache.org/` (visited on 05/28/2014) (cit. on p. 63).

[58] Apache Software Foundation. *Apache Maven Website*. 2014. URL: `http://maven.apache.org/` (visited on 05/28/2014) (cit. on p. 63).

[59] Kevin Townsend. *Adafruit NFC/RFID on Raspberry Pi*. 2012. URL: `https://learn.adafruit.com/adafruit-nfc-rfid-on-raspberry-pi/overview` (visited on 06/17/2014) (cit. on p. 76).

[60] Andreas Genser, Christian Bachmann, Christian Steger, Weiss Rudolf, and Josef Haid. "Estimation-based run-time power profile flattening for RF-powered smart card systems". In: *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)* December (2010), pp. 6–9. DOI: `10.1109/APCCAS.2010.5775006`.

[61] Doug Serfass and Kenji Yoshigoe. "Wireless Sensor Networks using android virtual devices and Near Field Communication peer-to-peer emulation". In: *2012 Proceedings of IEEE Southeastcon*. IEEE, 2012, pp. 1–6. ISBN: 9781467313759. DOI: `10.1109/SECon.2012.6196980`.

[62] Zhu Zhiyuan, Tan Jie, Hongsheng Zhao, Qiang Guan, and Na Li. "A dynamic RFID performance test system". In: *IEEE International Conference on RFID-Technology and Applications (RFID TA)* June (2010), pp. 17–19. DOI: `10.1109/RFID-TA.2010.5529855`.

[63] Google Inc. *Android Emulator Documentation*. 2014. URL: `http://developer.android.com/tools/help/emulator.html` (visited on 05/22/2014).