Master Thesis

Design and Implementation of an NFC Interface for Home Appliances and Consumer Electronics

Bašagić Rejhan, BSc

Institute for Technical Informatics
Graz University of Technology
Austria

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

Graz, May 2013

Abstract

At a time when smart phones occupy the market and have entered into everyday life, a space for new technological possibilities and exploitation of existing wireless technologies is created. There are many use cases where smart phones equipped with Near Field Communication (NFC) modules are used. Wider use of smart phones equipped with NFC technology depends greatly on the electronic devices that support this technology. Currently, an increasing number of NFC modules and NFC enabled smart phones are present on the market. Even for older mobile devices without NFC support, relatively simple and cost-favorable solutions appear and enable NFC for these devices. Problems arise in the search for appropriate and affordable solutions that would for non-NFC devices, such as home appliances and consumer electronics, enable this type of technology.

This project proposes a prototype, low-budget interface which introduces Near Field Communication as a wireless communication technology which can be successfully employed in interaction between users that operate Android NFC-enabled smart phones and non-wireless consumer electronics and home appliances. The practical part of this master thesis consists of design, construction and implementation of the proposed Near Field Communication Interface (NFC-I). Huge emphasis is on low-cost components that meet the set requirements of the NFC-I. The thesis also discusses the implemented procedures that enable communication between Android NFC-enabled smart phones and NFC-I equipped target devices. Android smart phone applications are designed and implemented to utilize all advantages of the NFC-I and to provide powerful graphical user interfaces with many features for interaction with the NFC-I equipped target devices. Additionally, possibilities and steps in the process of connecting the NFC-I to the target device are explained. This work is divided into several stages of design and implementation. Finally, all achieved results are presented on the example of implemented use cases.

Kurzfassung

Der Markt an Smartphones und mobilen Applikationen ist in den letzten Jahren in einem explosionsartigen Tempo gewachsen. Infolge dessen wurde auch Raum für neue technologische Anwendungsmöglichkeiten geschaffen, wie etwa Near Field Communication (NFC). Dank der Integration von NFC Modulen in aktuellen Smartphones, erlangte NFC einen hohen Verbreitungsgrad. Probleme ergeben sich jedoch bei der Suche nach geeigneten und preisgünstigen Lösungen für nicht NFC-fähige Geräte, wie beispielsweise Haushaltsgeräte und Unterhaltungselektronik, diese Art von Technologie zu ermöglichen.

Dieses Projekt schlägt eine preisgünstige und innovative NFC Schnittstelle für nicht NFC-unterstützende Haushaltsgeräte und Unterhaltungselektronik vor. Der praktische Teil dieser Masterarbeit besteht aus der Planung, der Konstruktion und der Implementierung der vorgeschlagenen NFC Schnittstelle (NFC-I). Der Schwerpunkt liegt auf Komponenten, die die gestellten Anforderungen des NFC-I erfüllen. Android Smartphone Anwendungen und deren leistungsfähige, mit vielen Funktionen ausgestattete, grafische Benutzeroberflächen werden entwickelt und eingesetzt um alle Vorteile der NFC-I ausgestatteten Zielgeräte zu nutzen. Zusätzlich werden Alternativen im Verbindungsaufbau zwischen NFC-I und dem Zielgerät erläutert. Schließlich werden alle erzielten Ergebnisse an Beispielen der implementierten Anwendungsfälle vorgestellt.

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)

Acknowledgement

This master thesis has been conducted at the Institute for Technical Informatics, University of Technology Graz in cooperation with Infineon Technologies Austria AG (Graz).

At this point, I would like to express my gratitude to Ass. Prof. Dipl.-Ing. Dr. techn. Christian Steger for his supervision and a great support during the preparation of this master thesis. Especially, I am very grateful to Dipl.-Ing. Norbert Druml, BSc, for his constant guidance, professional support and many hours of collaboration and constructive discussions which helped me during the work on the project and the master thesis. Further, I would like to thank Dipl.-Ing. Manuel Menghin and Dr. Josef Haid (Infineon Technologies Austria AG) for their inspiring support and invested time that helped me with the key parts of the project.

Finally, I would like to thank my family and friends for their invaluable support during all these years.

Graz, May 2013 Bašagić Rejhan

Contents

List of Figures

List of Tables

List of Abbreviations

ADT Android Development Tools

APDU Application Protocol Data Unit

APIs Application Programming Interfaces

C-APDU Command Application Protocol Data Unit

Dalvik VM Dalvik Virtual Machine

ECC Elliptic Curve Cryptography

ECDSA Elliptic Curve Digital Signature Algorithm

FRI Forum Reference Implementation

GPIO General Purpose Input/Output

I2C Inter-Integrated Circuit

IDE Integrated Development Environment

JDK Java Development Kit

JVM Java Virtual Machine

NDEF NFC Data Exchange Format

NDK Native Development Kit

NFC Near Field Communication

NFC-I Near Field Communication based Interface

NodeInt Node Interface Component

NtU NFC to UART Component

NVM Non-volatile Memory

OHA Open Handset Alliance

R-APDU Response Application Protocol Data Unit

RFID	Radio-frequency Identification
RTD	Record Type Definition
SDK	Software Development Kit
SPI	Serial Peripheral Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USCI	Universal Serial Communication Interface

Chapter 1

Introduction

For many people, life is unimaginable without mobile devices. Mobile devices, with their primary functions like sending and receiving messages and making calls, evolved into true small mobile computers with their own operating systems, many new features and imaginative applications. This thesis shows to which extent the smart phones have developed, which functions they have in our life and what are their innovative applications for the future. Research conducted in 2010 shows that smart phone sales rose by 72% in 2010 [Mic11] and the last quarter of 2010 showed, for the first time in history that the smart phone sales beat personal computer sales with 100.9 million smart phones sold [Sma11b]. Many statistic reports and experts say for 2011 that sales of smart phones is still in high growth and that the upcoming years will be the years of smart phones. Currently, the world's five leading smart phone platforms are: Android, Apple iOS, Blackberry, Microsoft Windows and Symbian. As depicted in Figure 1.1, especially Android, Apple iOS and BlackBerry continue with their growth, and stay on the top of the smart phone market enjoying the highest consumer's interests.

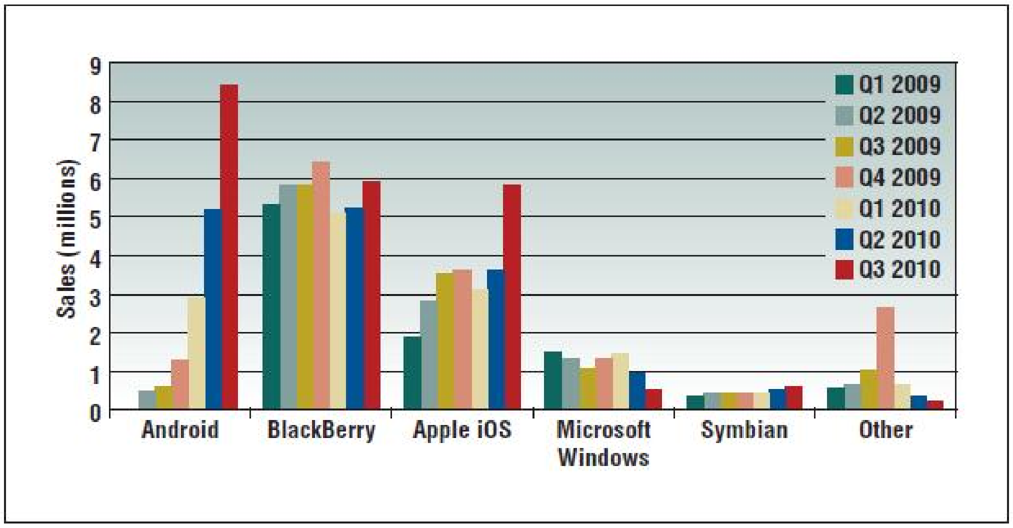


Figure 1.1: Smart phone Sale in North America (2009-2010) [But11].

According to [Sma11a], U.S. smart phone market for Q1 2011 is divided as follows:

37% of the market belong to Android devices, 27% to iPhone, 22% to Blackberry and 10% to Windows phone. In this paper, the focus is on the Android platform, its features, and all work covered by this thesis is based on the Android platform.

Along with the emergence and increasing prevalence of mobile devices, advanced technologies like Wi-Fi, Bluetooth, 3G and others, come to the fore. All these technologies have their applications, advantages and disadvantages. Yet with an idea where the mobile device could be used as universal tool for secure data exchange, secure identification or even advanced applications like contactless payment, a need for a new, universal solution arose. Technology that could be a solution for these problems is NFC. Although many call NFC the new technology, it's actually not, because it represents the evolution of existing specifications of Radio-frequency Identification (RFID) technology. Forecasters estimate that the NFC will become one of the leading technologies in contactless payment systems, marketing and organizational systems. According to latest reports [Poi11], the number of NFC equipped smart phones sold in 2011 will exceed the number of 116 million, and that number will rise to 500 million in the period between 2012 and 2015. Many mobile companies and mobile device manufacturers see this technology as a great opportunity to win the market. In the end, however, customers will be those who will benefit from NFC.

1.0.1 Objectives and Motivation

This project is part of the META[:SEC:][1] project which is conducted at the Institute for Technical Informatics in cooperation with the industrial partner Infineon Technologies Austria AG. The goal of the META[:SEC:] project is to analyse and develop new contactless smart card and RFID systems with an emphasis on power consumption analysis, security and optimization. As depicted in the Figure 1.2 the research is conducted in three approaches:

1. Simulation and software based approach
2. Emulation based approach
3. Hardware based verification approach

This master thesis' project focuses on the third approach, hardware verification, where a state-of-the-art Near Field Communication based system is developed and analysed. The two key motivation points reveal the main objectives of the project.

The first motivation is about wireless communication technologies which are used by consumer electronics and home appliances. In smart environments, home appliances and other similar applications, electronic devices are equipped with modules which enable wireless communications based on Wi-Fi, ZigBee, Bluetooth, etc.

This project proposes an innovative solution, in form of a low-budget interface, which provides wireless communication for non-wireless consumer electronics and home appliances. As a matter of fact, the idea is to construct and implement the Near Field Communication based Interface (NFC-I) which provides NFC for non-wireless consumer electronics and home appliances. With the help of the NFC-I, users are allowed to interact with

[1]**M**obile **E**nergy-efficient **T**rustworthy **A**uthentication **S**ystems with **E**lliptic **C**urve based **SEC**urity. Funded by the Austrian Federal Ministry for Transport, Innovation, and Technology under the FIT-IT contract 829586.

the NFC-I equipped home appliances and consumer electronics over NFC-enabled smart phones. Interaction in this case implies:

- Operating the target devices (consumer electronics or home appliances) over NFC
- Monitoring, controlling, maintaining and configuring the target devices over NFC

All these benefits and applications result in the emergence of new use cases where NFC can be effectively and profitably applied.

Each appliance within a home has a user interface through which the users operate these appliances. Such user interfaces consist out of numerous input and output hardware components such as: LCD displays, buttons, timers, lamps, etc. These components tend to consume significant amount of electrical energy. In an average household, there are more than 10 home appliances with visually different user interfaces. These interfaces are often also complex and users are facing difficulties using them.

As an input and output device through which the users are allowed to interact with NFC-I enhanced target devices, Android NFC-enabled smart phone is used. This project proposes Android applications with advanced graphical user interface which can replace the existing integrated user interfaces of home appliances. In such way, users interact with home appliance over an Android NFC-enabled smart phone and there is no more need for integrated user interfaces. This approach of interacting might also significantly reduce the costs for manufacturing appliances. For the purposes of the project, the Android smart phone platform was chosen because of its current leading role among other mobile platforms featuring NFC technology.

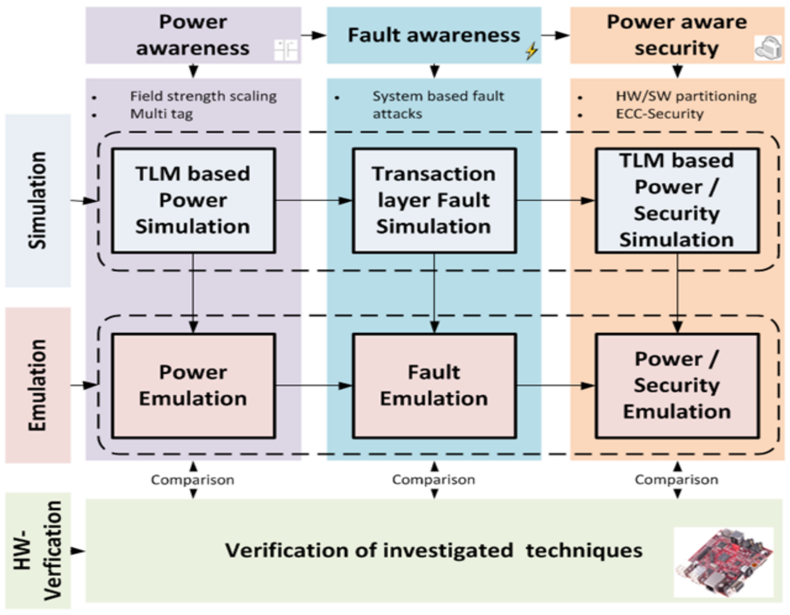


Figure 1.2: META[:SEC:] Project: Overview.

1.0.2 Structure of the Thesis

The most relevant topics and fundamentals which are subject to the project are summarized in Chapter 2. Section 2.1 briefly explains the Android operating system, its most important characteristics and components. A brief introduction to the development of Android applications is also covered. In this section, an overview of the NFC technology is given, typical use cases where NFC is applied are presented and a parallel between the NFC and Android is shown. A short introduction to Elliptic Curve Cryptography is also given. Section 2.2 surveys some interesting projects which are similar to this master thesis' project.

Chapter 3 deals with the detailed design of the implemented system, selection of hardware components used for realisation of the project and design of the software. In the beginning, overall system description, concept and user requirements for the system are outlined. Furthermore, the most important terms used during the design and implementation phases are explained. Last three sections in the chapter describe the individual components of the systems in detail, proposing the design decisions and solutions. Device connection schemes and protocols, used for the realisation of the project, and structure of the implemented applications are explained.

Chapter 4, with respect to the main three parts of the design process explained in Chapter 3, describes the development of software in detail. The main programs, initialization and interrupt routines are shown with complete listings of code and flow charts. The usage of implementation tools and environments is also described.

Chapter 5 surveys implemented use cases and gained results.

Finally, in Chapter 6, a conclusion and future work are given.

Chapter 2

Related Work

This chapter is devoted to the research on the most relevant technologies and topics for the master thesis and the project implemented within the thesis.

2.1 Fundamentals

2.1.1 Android

In November 2007, Open Handset Alliance (OHA), a consortium made up by the world's largest mobile operators, companies dealing with mobile devices, semiconductor producers and software companies, announced Android Software Development Kit (SDK) [All07]. Google, as a member of OHA is the initiator and leader in the development of Android. Android is a platform designed for mobile devices such as smart phones and tablet computers. One of the most important features of the Android platform is its openness and the idea to attract more and more users and developers who will make their own applications. Although the Android OS is relatively young and at the beginning of its development, it offers already a high-quality development environment and a database of applications developed by third party authors. According to [And11a] and [Gar11], Android platform with all features that is provides, in a relatively short period of time, has grown into a platform that will conquer mobile device market. Android provides a whole host of software for all types of mobile devices including smart phones tablets. The operating system itself is based on the Linux core. Other parts are adapted to pull the maximum from the hardware and to present end users an environment with rich user interfaces and easy application development.

2.1.2 Android Platform Architecture

As described in [Bur08], in addition to Android's openness, some other important characteristics are combined into this single platform:

- **All applications are treated equally:** There is no difference between the native Android application and additional (third party) applications. All applications are provided with equal access to resources of mobile device which gives the possibility to adjust the device for specific needs of individual users.

- **Automatic life-cycle management:** This feature provides additional stability to the Android system. Launching applications, switching between different modes of operation and closing of applications is perfectly regulated by the operating system. End users are no longer concerned about problems which are faced by running multiple applications on the mobile device.
- **Compatibility with the current and future hardware:** All applications are portable across different architectures including ARM and x86. The platform supports many input devices and user interface can be customized easily to meet the requirements of mobile devices with different screen resolutions.
- **Easy application development:** Android offers the ability to develop innovative and powerful application with different programming languages. It provides also powerful tools and useful software libraries. Many built-in services such as: location-based services, SQL database support, Near Field Communication, high quality sound and video support, support for various sensors etc., provide an ideal foundation for developing applications that have never seen before on mobile devices.

Android system architecture in based on the Linux kernel 2.6. Linux kernel is used as a Hardware Abstraction Layer (HAL) between the hardware and the software stack. However, the Android does not include all standard Linux utilities. According to [And13e], the reason for using Linux as a core for Android platform is its proven operational support and robustness, the ability to manage memory and processes, security model, network stack and constant development and improvement. Android system architecture is shown in Figure 2.1:

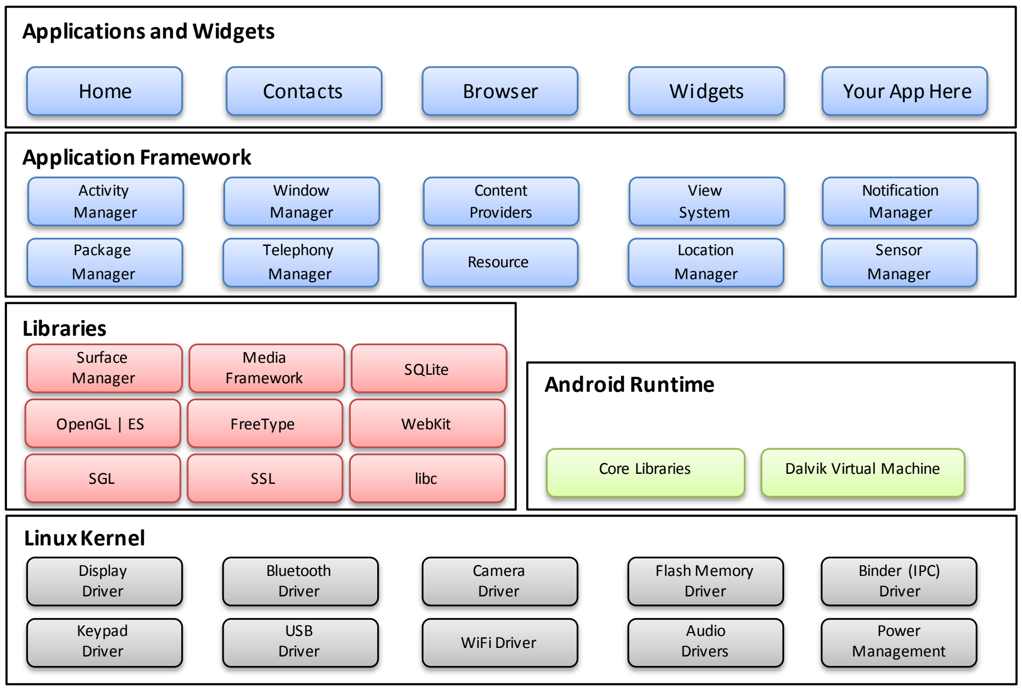


Figure 2.1: Android System Architecture [Bur08].

On the next few pages, a detailed overview of the Android architecture is given.

Linux Kernel

As already mentioned, Android relies on Linux kernel and contains Linux core system services. Stultz in [Stu11] outlines the parts (fixtures) which are added to Android in order to achieve the following:

- **Performance:** To improve the general performance of the system, three features are added: *Android Binder* - specific inter-process communication mechanism added to support method invocation and passing of arguments between the processes [Wik11]. *Ashmem* - Android shared memory manager introduces better memory allocation for low memory devices, and *LowMemoryKiller* - enables terminating of processes from user space in order to free up the memory.

- **Enable new hardware:** Support for a variety of input and output devices (touch screen, trackball, keyboard, etc.) is provided. Features like *Process Memory Allocator*, *Android timed output*, *Android Debug Bridge*, *Flash memory file system* (YAFFS2) and other small fixtures are implemented to enable easier adoption of a new hardware.

- **Improved power management:** Android features an improved power management to provide power savings. Wake locks in Android can be acquired and released both from user and kernel space. Applications use wake locks to prevent system from going to sleep state. When a wake lock is released, the CPU and other components of the system can go to sleep. Early suspend is another power management feature which sends components like LCD, sensors, etc. into the sleep state.

- **Error reporting:** Advanced reporting is achieved through Android *Logger*. Android RAM console allows storing messages in RAM after a kernel panic has occurred.

- **Security:** All Android applications have their unique user id (UID) and group id (GID). According to UIDs and GIDs, the system knows the security level for each application. Furthermore, Android network security mechanisms decide whether the application is granted to access networking features.

All these patches and changes are added on top of the Linux kernel. The changes are not that large, yet they give the Linux kernel exactly that, what an embedded system such as mobile device needs: better power management, better memory management and better inter-process communication.

Android Runtime

The next layer in the Android architecture is Android runtime which consist of Dalvik Virtual Machine (Dalvik VM) and Core Libraries. Core libraries (called Bionic libraries) are written in `Java` programming language and they represent all major `Java` functionalities and classes such as classes for manipulating collections, classes for communication with the environment, etc. Android uses `Java` as primary programming language. Yet, unlike other Java-based mobile platforms, instead of standard Java Virtual Machine (JVM) and standard Java Core Libraries, Dalvik VM and Bionic Libraries are used. The difference between classic JVM and Dalvik VM is that Dalvik VM is based on the registers and the JVM is based on stack. The most important characteristic of Dalvik VM is that it meets the needs of mobile devices in such way that it is optimized for low memory systems and slow CPUs [PK10]. Android applications are packed in *.apk* package file which contains, when converted, *.dex* (Dalvik Executable) files. Those lightweight files are executed and every Android applications runs as a separate process with its own instance of Dalvik VM.

Libraries

Native libraries sitting on the top of the Linux Kernel are written in the programming languages C and C++. They represent the next layer in the architecture of the Android system and according to [And13e] and [Goo08], some of the most important libraries are:

- **Libc:** Provides standard C library implementation for Android.
- **Secure Socket Layer (SSL):** This library provides security communication over the internet.
- **Scalable Graphics Library (SGL):** This graphical library is used by the majority of applications for construction of 2D graphical user interfaces.
- **OpenGL—ES:** Provides 3D elements which can be combined together with 2D elements in a single user interface. If available, 3D hardware will be used, if not, fast software renderer is used.
- **WebKit:** This, powerful open source web browser engine, which supports HTML, JavaScript and other web standards, is used for displaying web content and embedded web content.
- **FreeType:** This library is used for rendering of bitmap and vector fonts.
- **SQLite:** Lightweight database engine is provided by this library. It is used by Android applications for storing data in a database.
- **Media Framework:** Provides support for a variety of audio, video and image formats, as well as recording of audio and video.
- **Surface Manager:** The simple compositing window manager is responsible for rendering application components in time and space.

Libraries provide low level functionality and they are used by different components of the Android system through Application Framework.

Application Framework

The Application Framework layer is written in Java and contains a set of common software components or Application Programming Interfaces (APIs) which provide structural basis for application development. Pre-installed Android application utilize these APIs and developers are also provided with the full access to them. In other words, the Application framework facilitates the process of developing new Android applications by using existing and proven routines within the system. These routines provide for example: access to sensors and hardware components, access to phone features, access to background services, etc. For example, if an e-mail client application is provided by Android, it is not necessary to implement functions for handling notifications when receiving or sending e-mails. These functionalities are already provided by the system, and the user just needs to learn how to use existing components and classes. Furthermore, the user is given the opportunity to replace core Android applications by his own applications which perform exactly the same tasks. Some of the most important components of the Application Framework are:

- **Activity Manager:** During the life-cycle of an Android application, it can be in one of the following activity states: *Starting*, *Running*, *Paused*, *Stopped* and *Destroyed*. The developers define and implement the transition methods for switching between the states and Activity Manager calls them in appropriate time.
- **Package Manager:** Provides information about installed packages (applications) on the mobile devices. This information consists of capabilities and supported system features which applications have.
- **Window Manager:** The role of the Window Manager is to regulate the order of appearance of windows on the mobile device's screen. The windows belong to different applications which are operating at the same time on the mobile device.
- **Content Providers:** Content Providers enable sharing of data between applications. For example, information like contacts, phone numbers and e-mail addresses which are stored in the database, with the help of Content Providers, can be accessed from multiple applications.
- **View System:** The View System provides a set of graphical elements: text boxes, lists, tables, layouts, widgets, etc., which are used when building the application's user interface.
- **Notification Manager:** Manages all kinds of alerts as well as events such as: incoming SMS or e-mail, upcoming meeting, information about mobile networks, etc. These alerts are displayed in the status bar of the mobile device.
- **Telephony Manager:** Enables access to phone's features such as: state of the phone (e.g., ringing, idle), type of the network, SIM card security options, etc.
- **Resource Manager:** Handles all non-code application resources (e.g., images, graphics, audio files, video files, documents, etc.).
- **Location Manager:** Provides access to installed GPS devices and geolocation settings.

Beside these, there are many other APIs that provide access to hardware devices and features like Wi-Fi, USB, Bluetooth, NFC, different sensors, etc.

Applications and Widgets

The Applications and Widgets layer is the last layer in the architecture of the Android platform and represents a layer which is visible to the end user. This includes core system applications such as Web browser, Calendar, Clock, Contacts, Email client, Messaging applications, Application Widgets, Google Play, etc. In order to access and install third party applications, the user needs to visit the Android Play Store[1]. According to latest headlines [And11b], in September 2011, there are more than 500 000 unique applications and games published in the Android Play. Applications and games are divided into many categories and there are free applications and applications for which the user needs to pay. Android Play Store staff regularly publishes and updates the lists of the most popular free and paid applications. The same selection is conducted for every category in order to present users best, most popular and most downloaded applications.

2.1.3 Android Application Development

The process of creating Android applications, as shown in the Figure 2.2, consists of four parts. The first step is to prepare the development environment by installing all necessary tools and configuring them. An integral part of this work is the development of an Android application, and in that section, a detailed discussion on developing and testing is given. Final step is the publishing of the created application to the Google Play Store.

In general, the development of Android applications is quite easy and at the beginning, the Android based smart phone is not required. All that is necessary is a development computer with an installed `Java`, Android SDK and an appropriate Integrated Development Environment (IDE). Android SDK includes various tools, documentation, sample code, and it provides support for developing, testing and debugging of Android applications. Multiple platforms (Windows, Linux and Mac OS) are supported.

[1]`https://play.google.com/store`

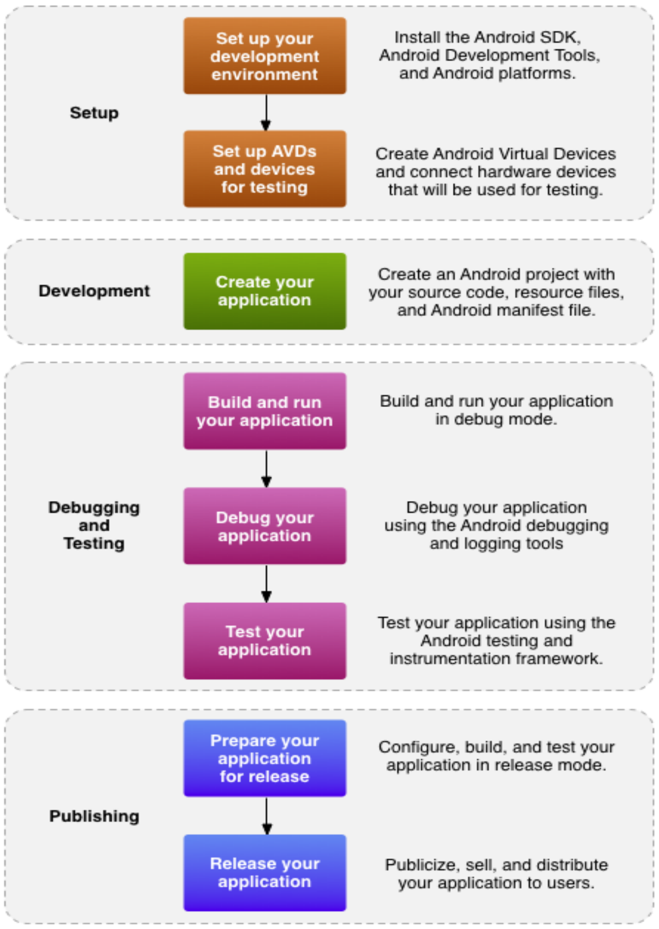

Setup
Set up your development environment
Install the Android SDK, Android Development Tools, and Android platforms.
Set up AVDs and devices for testing
Create Android Virtual Devices and connect hardware devices that will be used for testing.
Development
Create your application
Create an Android project with your source code, resource files, and Android manifest file.
Debugging and Testing
Build and run your application
Build and run your application in debug mode.
Debug your application
Debug your application using the Android debugging and logging tools
Test your application
Test your application using the Android testing and instrumentation framework.
Publishing
Prepare your application for release
Configure, build, and test your application in release mode.
Release your application
Publicize, sell, and distribute your application to users.

Figure 2.2: The Development Process of Android Applications [And13a].

Android NDK

Except `Java`, which is a main Android programming language, applications can be developed in native programming languages `C` and `C++` with the help of the Android Native Development Kit (NDK) [And13b]. The main reasons for using Android NDK are applications or portions of source code which require more performance. In that case, an application can be executed rather directly on the CPU than through Dalvik VM. Before start programming, the developers should consider the advantages and disadvantages of programming Android applications with native programming languages. It happens often that such an implementation is very complex, but in the end it does not improve performance [LJ10]. Android NDK is typically used for CPU-intensive applications, applications that include simulations, graphic-intensive applications or application which require direct access to hardware.

Android Hardware Level Programming

Through development packages, Android provides APIs which enable direct access to the individual hardware components of the mobile device. Such packages are for example:

`android.hardware` (provides access to camera and sensors), `android.hardware.usb` (support for connected external USB devices), `android.bluetooth` (provides access to Bluetooth adapter) and many others. If predefined Android APIs do not provide appropriate solutions or procedures for the project needs, it is necessary to make changes directly in the APIs, or in the kernel level. Such modifications are allowed, because Android is an open source software. Source code and obtaining instructions are available on the official web site [And13d].

2.1.4 Near Field Communication

NFC technology is being developed as a combination of existing wireless technologies. Yet, it brings entirely new possibilities for developing new great wireless systems where the interaction takes place between two NFC equipped devices or between a device and an NFC tag. The communication approach is very simple; it is enough to touch the device or special label (tag) to start communicating. The co-inventor of the NFC technology is NXP Semiconductors, founded by Philips. Along with Sony and Nokia, the company founded in 2004 the NFC Forum [For13], a non-profit organization for the promotion of this technology [Wan11]. NFC, as already said in the beginning, is not a new technology, but rather an evolution of existing specifications of RFID technology. As stated in [Poi11], the most frequent sort of communication in the world of NFC is between the active mobile device and a passive tag, but with the increasing emergence of smart phones with integrated NFC chips, the communication between devices over NFC is becoming more popular.

2.1.5 NFC Technology

NFC is a subset of RFID and operates in high frequency band at 13.56 MHz using the three protocols known in the RFID world: ISO 14443-Type A and Type B [ISO08] and FeliCa [Wan11]. The maximum speed transfer can be 106 Kbit/s, 212 Kbit/s and 424 Kbit/s at a distance of approximately 5-10 cm. Table 2.1 shows the comparison between the NFC and other related technologies:

	NFC	**RFID (UHF)**	**IrDA**	**Bluetooth**
Set-up time	<0.1ms	<0.1ms	0.5sec	6sec
Range	Up to 10cm	Up to 3m	Up to 5m	Up to 30m
Usability	Easy, fast	Item centric, Easy	Data centric, Easy	Data centric, Medium
Selectivity	High	Partly given	Line of sight	Who are you?
Consumer experience	Touch, wave	Get information	Easy	Configuration needed

Table 2.1: Comparison between NFC and Related Technologies [AITH09], [Sup12].

NFC supports three communication modes:

1. **Reader/Writer Mode:** Most commonly used for data exchange, i.e., reading the information stored on the tag (label) with the help of an NFC-enabled device. Tag represents a card which, as depicted in Figure 2.3 contains NFC antenna and small integrated circuit that can store small amount of data. Tag itself does not have a power source, yet it is powered with the help of NFC-enabled device (reader or writer). NFC Forum, in [NXP09], specifies four types of tags that can be used together with NFC:

   - **Type 1 and Type 2:**
     - Based on the ISO/IEC 14443 Type A standard
     - Small capacity (1 or 2 KB)
     - Cheap to produce
     - Transfer speed is limited to 106 Kbit/s
   - **Type 3:**
     - Based on the FeliCa standard
     - Higher capacity (1 MB)
     - Higher price
     - Transfer speed is limited to 212 Kbit/s
   - **Type 4:**
     - Based on the both Type A and Type B ISO/IEC 14443 standards
     - Capacity of 32KB
     - Transfer speed between 106 Kbit/s and 424 Kbit/s

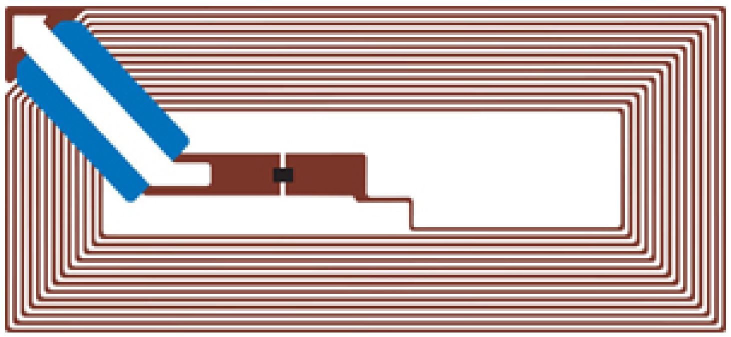


Figure 2.3: Tag (Label).

2. **Card Emulation Mode:** In this case, it is a reverse process. NFC device acts as a tag and works in a passive mode. The device can emulate unlimited number of smart cards which are read by an external NFC reader. It is mainly used as a method of payment or for identification.

3. **Peer-to-peer mode:** For this communication mode, two NFC-enabled devices are needed. It is used for direct data exchange between the devices.

Data structure defined by the NFC Forum and used to exchange data between two NFC devices or between NFC devices and tag is called NFC Data Exchange Format (NDEF) message. Each NDEF message can be composed out of unlimited number of NDEF records, each of which contains a description of information type that it holds and its length [Wan11]. These information types are defined as Record Type Definition (RTD). According to NFC Forum ([For13] and [Wan11]), commonly used RTD types are:

- **NFC Text RTD:** Used for exchange of simple text string.
- **NFC URI RTD:** In this record type, Uniform Resource Identifiers (URIs) which are used to describe the resources on the Internet, are compressed into 1-byte field of the NDEF header.
- **Smart Poster RTD:** This type allows combining of multiple text and URI NDEF records, creating a self-describing 'smart object'.
- **NFC Generic Control RTD:** Contains a request for an NFC device to perform an action: start application, store data, modify tag, etc.
- **NFC Signature RTD:** Defines a format which is used to secure NDEF records. Includes information about signature algorithms and certificates.

2.1.6 Concrete Implementation of the NFC Chip

Samsung Nexus S [Sam11], an Android based NFC-enabled smart phone used for the realization of the practical work of this masters thesis, is equipped with the PN65 NFC module [Res11a], which is manufactured by NXP. As shown in the Figure 2.4, this module combines NFC controller PN544 and the SmartMX secure element.

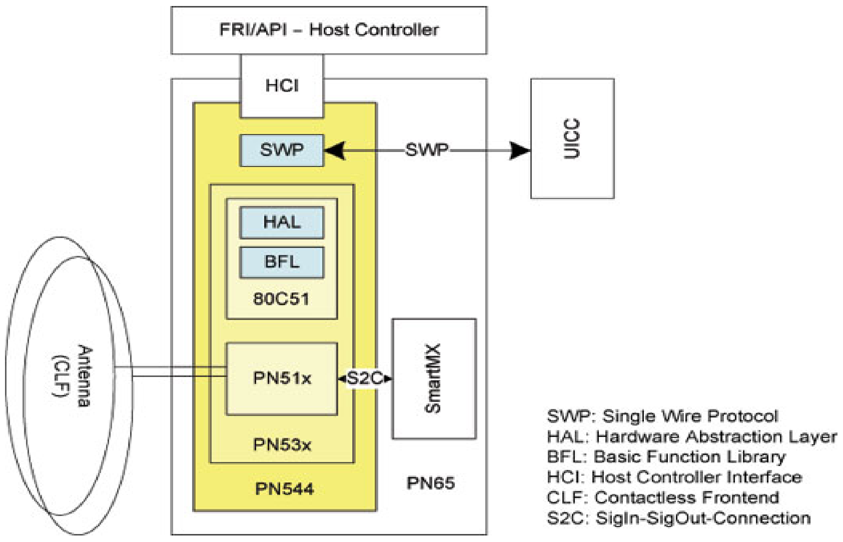

FRI/API – Host Controller
HCI
SWP
SWP
UICC
HAL
BFL
80C51
Antenna (CLF)
PN51x
S2C
SmartMX
PN53x
PN544
PN65
SWP: Single Wire Protocol
HAL: Hardware Abstraction Layer
BFL: Basic Function Library
HCI: Host Controller Interface
CLF: Contactless Frontend
S2C: SigIn-SigOut-Connection

Figure 2.4: NXP: NFC Controller PN65 [Res11a].

Moreover, PN544 controller consists of following components:

- PN512 - transceiver for contactless communication at 13.56 MHz which supports all three NFC communication modes (Reader/Writer, Card Emulation and Peer-to-peer mode) [NXP11b].
- 80C51 - a microcontroller which is running the firmware for the PN512 transceiver [Res11a].
- SWP - Single Wire Protocol is used to support SIM card (UICC) as an additional secure element.

In case of Samsung Nexus S, an embedded secure element P5CN072 (SmartMX) is used for security issues. According to controllers specification [NXP11a], it supports a variety of security algorithms such as: AES, DES, RSA, ECC, ECDSA, etc.

2.1.7 NFC and Android

For their NFC chips PN544 and PN531, NXP provides a software stack, so called Forum Reference Implementation (FRI) which is already supported and integrated in Android version 2.3.3 and later [Res11b]. The main role of FRI is the mediation between NFC chip drivers and the Android Application Framework layer, which provides NFC API for developing NFC applications. Figure 2.5 shows the whole software stack which involves FRI and Android NFC Java Layer:

FRI is implemented in native `C` and communicates directly with the NFC chip. On top of it, there is a Java Native Interface (JNI) that acts as a bridge between native programming languages `C/C++` and `Java`. The next layer is Android Application Framework which provides NFC API [Res11b]. Android NFC API provides support for NFC and allows developers to build applications involving NFC in all three communication modes (Reader/Writer, Card Emulation and Peer-to-peer mode). The main classes of this API are [And13c]:

- `NfcManager` - Is used to obtain an instance of the `NfcAdapter`.
- `NfcAdapter` - Represents the default adapter (NFC module), which is located inside the mobile device.
- `NdefMessage` - This class provides support for reading and writing of NDEF messages.
- `NdefRecords` - Represents NDEF records and provides methods for working with NDEF records.

NXP FRI is open source and it is located within the Android source code under:
`/external/libnfc-nxp`.

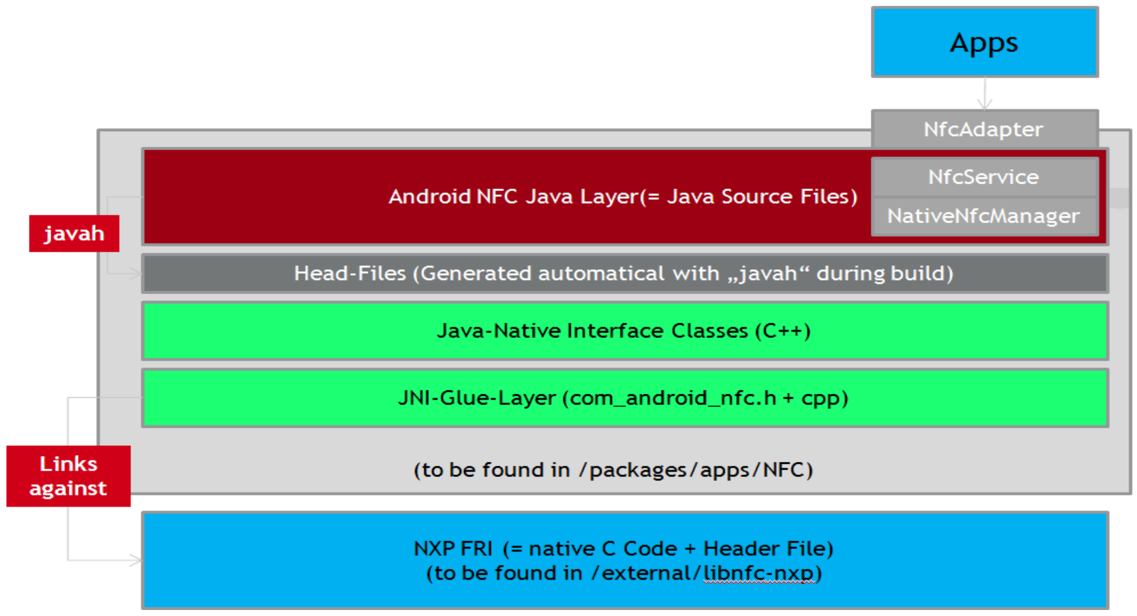

Apps
NfcAdapter
NfcService
NativeNfcManager
Android NFC Java Layer(= Java Source Files)
javah
Head-Files (Generated automatical with „javah“ during build)
Java-Native Interface Classes (C++)
JNI-Glue-Layer (com_android_nfc.h + cpp)
Links against
(to be found in /packages/apps/NFC)
NXP FRI (= native C Code + Header File)
(to be found in /external/libnfc-nxp)

Figure 2.5: NFC Software Stack [Res11b].

2.1.8 NFC Use Cases

NFC is one of the keenest technologies nowadays and its application is found in every branch of the economy, in all areas of science and society in general. Development of contactless technology has allowed one of the most important innovations in the payment industry - contactless payments. The spread of NFC chips and NFC-enabled smart phones should probably lead to smart phones as the only method of payment and identification. Google has already started the Google Wallet [PE11], an Android application based on NFC technology which, in the real sense, replaces standard payment methods.
Some industry examples in which the NFC found its application are surveyed by NFC Forum in [NF08]:

- **Mass and public transport**
  - Buying a ticket
  - Paying a taxi or a bus
- **Advertising**
  - Information gathering from public poster/tag
- **Drivers and Vehicle Services**
  - Paying parking fees
  - Use as a driver's license

- **Security**
  - Access secure areas
  - Entering/leaving buildings
  - Logging in to PC
- **Banking**
  - Contactless payment
- **Entertainment**
  - Buying tickets
  - Event information
- **Consumer electronics**
  - Bluetooth pairing
  - Direct access to printer and projector
  - Quick WLAN set-up

These are, of course, not the only fields where the NFC is successfully applied because the idea of NFC is quickly embraced by many in the world of technology. Linking NFC-enabled smart phones with the social network is just another idea where NFC could have a significant place. It is difficult to predict where the NFC will be used and in what context in the near future. Yet its role could greatly facilitate the lives of people.

2.1.9 Elliptic Curve Cryptography

Cryptographic systems based on elliptic curves are present in all sectors of the computer world. Due to its performance and small power consumption, ECC is heavily used in mobile devices with limited memory and slow CPUs, smart cards, electronic banking, etc. Electronic signature version of ECC, so called Elliptic Curve Digital Signature Algorithm (ECDSA) is a good alternative to a well-known Public-key Cryptography (PKI) algorithm RSA (Rivest, Shamir, Adleman), primarily because elliptic curves provide faster mathematical calculations, stronger safety parameters, faster key generation and shorter keys. Such improvements are very important for embedded systems and mobile devices which utilize cryptographic systems. Table 2.2 shows a comparison of key sizes between different cryptographic systems with an equivalent security level:

Symmetric	**ECC**	**DH/DSA/RSA**
80	163	1024
128	283	3072
192	409	7680
256	571	15 360

Table 2.2: Comparison between ECC and other Cryptographic Systems: Key Sizes (in Bits) [Lau04].

ECC was introduced in 1985, by cryptographers Neal Koblitz [Kob87] and Victor Miller [Mil86]. They worked on Discrete Logarithmic Problem of Elliptic Curves (ECDLP), from which the ECDSA was originated [Lau04].

2.1.10 Mathematical Background of ECC

According to Lauter in [Lau04], for the cryptography purposes elliptic curve can be taken in the following form:

$$y^2 = x^3 + ax + b, \tag{2.1}$$

where a and b are elements of the finite field F_p. Finite fields consist of a limited number of elements and operations addition and multiplication which can be carried out between two elements. Elliptic curves used in cryptography are defined by two types of finite field, and in this case, as shown in the equation 2.1, the finite field of prime numbers (where p is a prime larger then 3) is used. Addition and multiplication in F_p are defined as shown in the equations 2.2 and 2.3:

$$a + b = r \,(\mathrm{mod}\, p)\,, r \in [0, p-1] \tag{2.2}$$

$$a \cdot b = s \,(\mathrm{mod}\, p)\,, s \in [0, p-1] \tag{2.3}$$

As illustrated in the Figure 2.6 and according to [Lau04], mathematical property which makes the elliptic curve suitable for use in cryptography says: if two points ($\mathbf{Q}_1$ and $\mathbf{Q}_2$) on the curve are chosen, then the line that connects them intersects the curve in third point ($\mathbf{R}_1$). If $\mathbf{R}_1$ is reflected across the x-axis (the curve is symmetrical about the x-

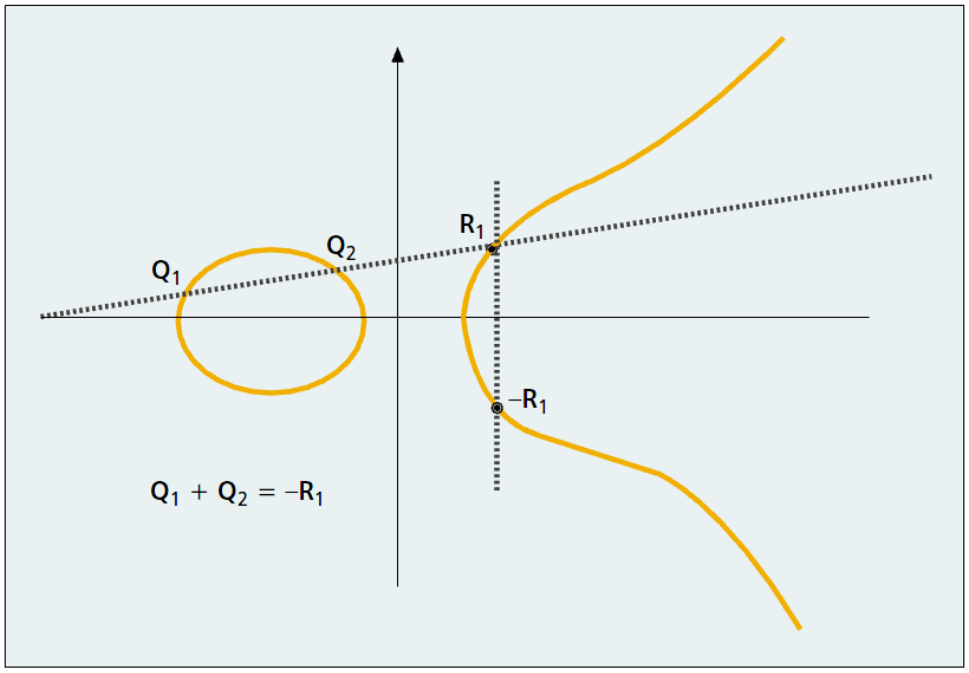


Figure 2.6: Elliptic Curve Cryptography: Representation of Points on the Elliptic Curve [Lau04].

axis), another point, $\mathbf{-R}_1$, is obtained. Provided that the infinity point is defined (0) and considering that $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{R}_1$ lie on the common straight line, it can be concluded that $\mathbf{Q}_1 + \mathbf{Q}_2 = \mathbf{-R}_1$.

If points $\mathbf{Q}_1$ and $\mathbf{Q}_2$ coincide, it is possible to define a reflection point $\mathbf{Q}_1 + \mathbf{Q}_1$ which is denoted as $2\mathbf{Q}_1$. By same logic, it is possible to determine the $k\mathbf{Q}_1$ for any integer k. From this follows the definition of ECDLP: if there is a point $\mathbf{Q}_1$ and a point $k\mathbf{Q}_1$ on the curve, value of k is unknown. It is believed that for a suitable elliptic curve and point $\mathbf{Q}_1$, finding of k is a very difficult problem.

2.1.11 Elliptic Curve Diffie-Hellman Key Exchange Protocol

When defining an elliptic curve system, it is necessary to define the curve and the base point **Q**. These two elements are not confidential and may be the same and public for all users. For a given base point it is easy to generate a private key (random integer **k**) and a public key (point **kQ** on the curve). Key exchange protocol, defined by Diffie and Hellman [Li10], is used in cryptography for exchanging keys and defining common secret keys in situations where the secure exchange channel is not provided. Diffie-Hellman key exchange protocol for ECC, between two participants, is shown in the following steps [ES06]:

1. Users A and B agree on an elliptic curve **E**, and a base point on the curve, **Q**. This pair, **(E,Q)**, represents the public key.
2. A chooses a private key **a**, computes $\mathbf{Q}_a = \mathbf{a} \cdot \mathbf{Q}$ and sends $\mathbf{Q}_a$ to B.
3. B chooses a private key **b**, computes $\mathbf{Q}_b = \mathbf{b} \cdot \mathbf{Q}$ and sends $\mathbf{Q}_b$ to A.
4. A computes $\mathbf{a} \cdot \mathbf{Q}_b = \mathbf{abQ}$ and B computes $\mathbf{b} \cdot \mathbf{Q}_a = \mathbf{abQ}$.
5. Users agree on a way how to extract a key from **abQ**.

If **E**, **Q**, $\mathbf{Q}_a$ and $\mathbf{Q}_b$ are known to the third user C, still, finding of **abQ** is infeasible.

2.1.12 Elliptic Curve Digital Signature Algorithm

ECDSA is a variant of the Digital Signature Algorithm (DSA) which uses Elliptic Curve Cryptography. The process of generating keys, signing and verifying messages with ECDSA is explained by Johnson et al. [JMV01] in several steps.

ECDSA is defined by the following six elements $\mathbf{D} = (\mathbf{p}, \mathbf{a}, \mathbf{b}, \mathbf{Q}, \mathbf{n}, \mathbf{h})$, where:

- **E** is the curve,
- **p** is a prime number representing size of the field,
- $\mathbf{F}_p$ represents finite field,
- **a**, **b** are elements of the field that determine the elliptic curve,
- **Q** is the base point on the curve $\mathbf{E}(\mathbf{F}_p)$,
- **n** represents order of the base point **Q**,
- $\mathbf{h} = \#\mathbf{E}(\mathbf{F}_p)/\mathbf{n}$ represents cofactor (number of points on the elliptic curve divided by the order of the base point)

Key Generation

Generation of ECDSA public and private keys is shown in the following steps:

1. Choose a random number **d** (private key) from the interval $[\mathbf{1,\ n\text{-}1}]$.
2. Calculate **R** (public key): $\mathbf{R} = \mathbf{d} \cdot \mathbf{Q}$.

The validity of the public key is checked as follows:

- Check that $\mathbf{R} \neq 0$,
- Check that **x** and **y** coordinates of **R** are correctly represented elements of finite field $\mathbf{F}_p$,
- Check that the point **R** lies on the elliptic curve **E**,
- Check that $\mathbf{nR} = 0$.

If any of these checks fails, **R** is invalid.

Process of Signing the Message

The procedure of signing message **m** is shown in the following steps:

1. Choose a random number **k** from the interval $[\mathbf{1,\ n\text{-}1}]$,
2. Compute $\mathbf{kQ} = (\mathbf{x,\ y})$ and $\mathbf{r} = \mathbf{x \bmod n}$ (if $\mathbf{r} = 0$, go back to first step),
3. Calculate $\mathbf{t} = \mathbf{k}^{-1}\, mod\, \mathbf{n}$,
4. Calculate $\mathbf{e}$ = SHA-1($\mathbf{m}$), where SHA-1 is a hash function,
5. Use private key **d** to calculate $\mathbf{s} = \mathbf{k}^{-1}(\mathbf{e} + \mathbf{rd})\, mod\, \mathbf{n}$ (if $\mathbf{s} = 0$, go back to first step).

Pair $(\mathbf{r,s})$ represents the signature of the signed message **m**.

Verification Process

Verification of the signature is the process where the recipient checks the received message **m** and its signature pair $(\mathbf{r,s})$. For the verification process all cryptosystem's parameters **D** are required, and the following actions are performed:

1. Ensure that **r** and **s** are form the interval $[\mathbf{1,\ n\text{-}1}]$,
2. Calculate $\mathbf{e}$ = SHA-1($\mathbf{m}$),
3. Compute $\mathbf{w} = \mathbf{s}^{-1}\, mod\, \mathbf{n}$,
4. Calculate $\mathbf{u}_1 = \mathbf{e} \cdot \mathbf{s}^{-1}\, mod\, \mathbf{n}$ and $\mathbf{u}_2 = \mathbf{r} \cdot \mathbf{s}^{-1}\, mod\, \mathbf{n}$,
5. Calculate the point $\mathbf{X} = (\mathbf{x}_1,\ \mathbf{y}_1) = \mathbf{u}_1 \cdot \mathbf{Q} + \mathbf{u}_2 \cdot \mathbf{R}$,
6. If $\mathbf{X} = 0$, signature is invalid and must be rejected, otherwise compute $\mathbf{v} = \mathbf{x}_1\, mod\, \mathbf{n}$,
7. Accept the signature for message **m**, if and only if $\mathbf{v} = \mathbf{r}$.

2.1.13 Importance of the ECC for the Thesis and the Implemented Project

In this section, an overview of the ECC is given because this cryptographic system is the most suitable for embedded systems, mobile devices and systems where smart cards are used. This master thesis proposes an NFC system where communication takes place over NFC between users, operating Android NFC-enabled smart phone, and target home appliances or consumer electronics equipped with the NFC-I. Such communication can be subject of an attack from outside. Therefore, it is desirable for the communication between the entities of the system to be secure. As previously mentioned, Android platform and the used NFC-enabled smart phone already provide support for the ECC. Furthermore, during the selection of components for the NFC-I, special attention is paid to ensure that the selected components are suitable for the implementation of ECC.

2.2 Similar Projects

This section surveys some interesting projects and ideas which, in similar context to this master thesis' project, attempt to make use of NFC and NFC-enabled smart phones. Furthermore, smart phone solutions involving other wireless technologies, such as Bluetooth and Wi-Fi, are discussed.

2.2.1 NFC-enabled Solutions

NXP, as a pioneer in NFC, leads in development of innovative projects where NFC is used in combination with Android NFC-enabled smart phones. NXP in [Mui12] demonstrates the washing machine which recognizes fabric type and color from NFC-tagged clothes buttons. This helps to avoid washing together clothes of different fabrics and colors. Also, the washing machine optimizes the washing program based on retrieved information about clothes types and colors. Furthermore, this project shows how, with the help of NFC-enabled smart phone, the technicians can perform maintenance of the washing machine over NFC.

More recently, in MWC 2013[1], NXP has demonstrated a range of new NFC applications and demos [NXP13b], [NXP13a]:

- Wireless charging pad: This solution enables wireless charging of smart phones featuring different wireless charging standards. The charging pad operates in zero energy standby mode, i.e. it consumes zero energy when not charging a mobile device. As depicted in the Figure 2.7, when user places the NFC enabled smart phone on the charging pad, the smart phone, with the help of NFC, supplies the charging pad with power necessary to activate the power supply of the charging pad and the charging can begin. Charging pad supports also exchange of data over NFC with the NFC-enabled smart phone.

- Think&Go NFC: In this project, NFC tags are used in supermarkets to provide buyers with additional information about the products. Each product in the shelves has its own NFC tag. Buyers, operating NFC-enabled smart phones, can retrieve

[1]`http://www.mobileworldcongress.com/`

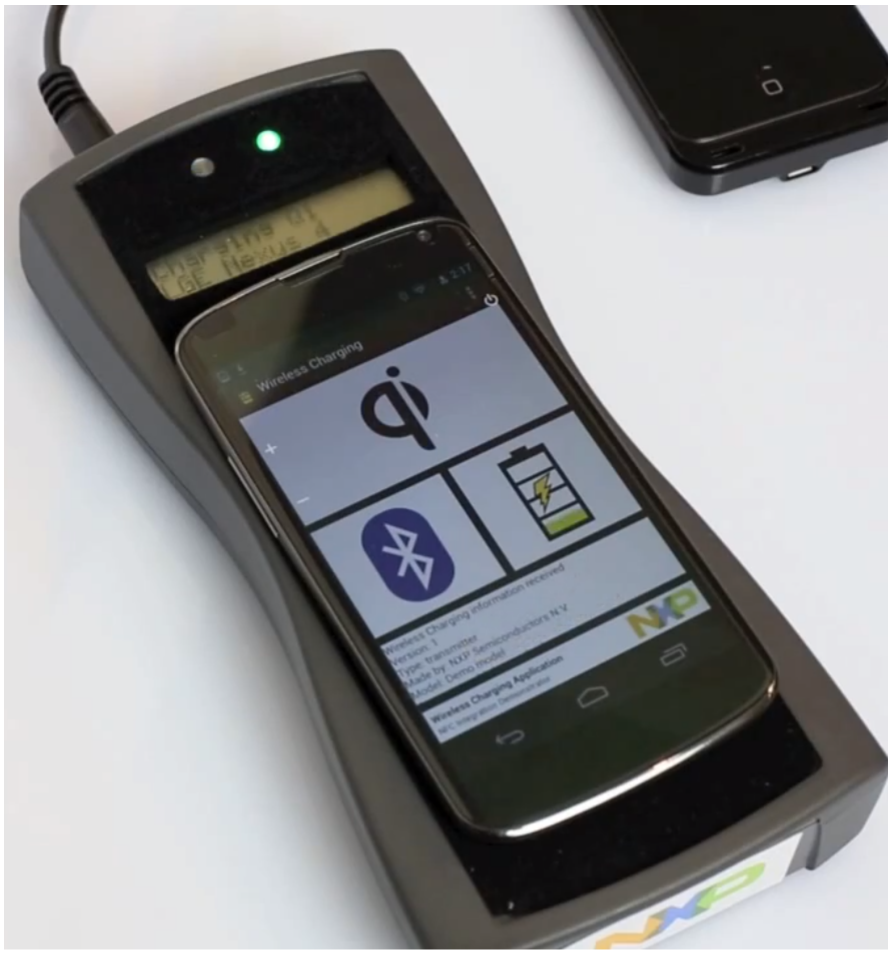

Wireless Charging
Wireless Charging information received
Version: 1
Type: transmitter
Made by: NXP Semiconductors N.V.
Model: Demo model
Wireless Charging Application
NFC Integration Demonstrator

Figure 2.7: NXP MWC 2013: Wireless Charging Pad [NXP13b].

additional information about products or add them to the shopping cart. Additionally, payment terminals in supermarkets are equipped with NFC technology allowing buyers to perform payment in fast way by placing the NFC-enabled smart phones upon the payment terminals.

- MIFARE4MOBILE: This service represents advanced mobile ticketing solution involving NFC-enabled smart phones and NFC equipped ticketing terminals. More than 650 ticketing systems around the world are equipped with this service, providing users with a standardized secure solution for buying tickets.

As it can been seen in the NXP's example, the market does not lack new ticketing systems enabling NFC technology. However, the question arises as to whether it is possible to provide NFC for existing ticketing systems without support for NFC. Widmann et al. in [WGSL12] introduce a solution that would for existing public transport systems provide NFC based ticketing. On the example of VDV Core Application (German electronic ticketing system) the paper shows which additional components are required to enable NFC based ticketing. Furthermore, use case scenarios demonstrate the process of interacting between the customers operating Android NFC-enabled smart phones and the implemented system. The result of this project is a proof of concept solution which was also successfully integrated in the existing ticketing system.

Despite the fact that iPhones are yet not provided with NFC technology, Apple in the 'Patently Apple' blog ([Blo10]) introduces the NFC technology which can be utilized to control all in-home electronic devices and consumer electronics with an iPhone. The NFC interfaces integrated in the controllable electronic devices provide iPhones with control parameters and other information which instruct the user how to control the device.

Another interesting project presented in [CPL12] shows how to make use of NFC tags and Android NFC-enabled smart phones. The idea is to simplify the connecting and exchange of data between Android NFC-enabled smart phones and consumer electronics. NFC tags containing information such as connection establishing parameters and supported media types are stored on a tag which is attached to the device. By placing the Android NFC-enabled smart phone upon the tag, the information is read from the tag. Depending on the type of tagged device and the read information, following implemented example Android applications can be started on the Android NFC-enabled smart phone [CPL12]:

- Touch&Connect: This application uses the NFC to acquire Wi-Fi connection establishing parameters and password from a tagged badge. Worldwide, visitors attending conferences receive the badges when entering the conference. The NFC tag attached to the badge contains all information required by users to establish the Wi-Fi connection with a router. When user scans the badge with the Android NFC-enabled smart phone, as depicted in the Figure 2.8, the application automatically establishes the Wi-Fi connection.

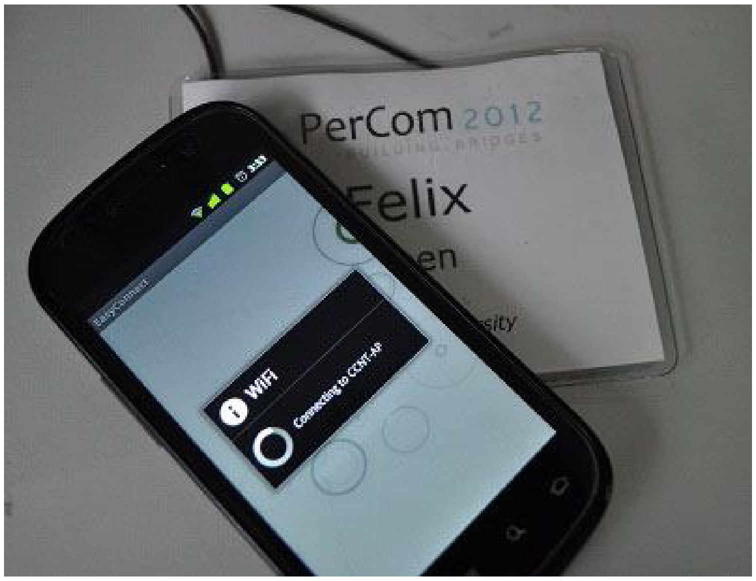


Figure 2.8: Touch&Connect: Establishing Wi-Fi connection by Scanning the Badge [CPL12].

- Touch&Listen: This application enables the users to automatically connect the Android NFC-enabled smart phone to the stereo system, and to stream the music from the smart phone to the stereo system. NFC tag attached to the stereo systems contains all necessary information for establishing the Bluetooth connection between the stereo system and the smart phone. When user places the Android NFC-enabled smart phone upon the tag, Touch&Listen application is automatically started and the Bluetooth connection with the stereo system is established. From the moment, stereo system's speakers are used to play the music which is streamed over Bluetooth.
- Touch&Watch: Similarly to Touch&Listen application, Touch&Watch application is started upon detecting the NFC tagged TV set. Android NFC-enabled smart phone, depending on what is supported by the TV, uses the Wi-Fi or the Bluetooth to establish the communication with the TV, and to transfer media data (photos and videos) to the TV screen. This allows users a more conform way of reviewing photos and videos.

Google Wallet

Google Wallet[1] application is another great feature of Android NFC-enabled smart phones. It allows users to use their smart phones instead of classic payment (credit) cards. Credit cards are emulated by the smart phone using the NFC card emulation mode. In order to make the transaction, the user just needs to place the smart phone to the NFC enabled payment terminal, as shown in the Figure 2.9. In addition to security features in communication between the smart phone and the payment terminal, the user needs to input the password to authorize the payment transaction.

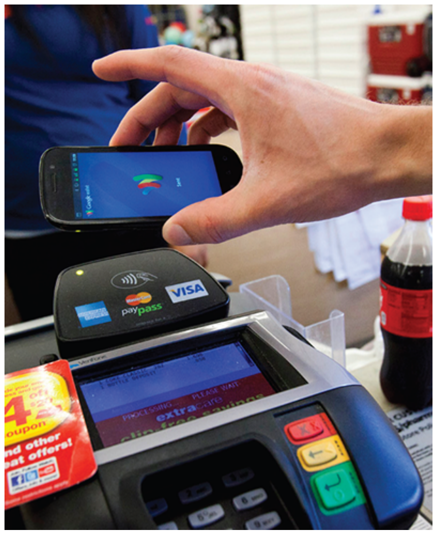


Figure 2.9: Google Wallet [P.E12].

Google has many plans for Google Wallet in the future. Not only multiple credit cards shall be emulated, but also identification cards, passports, access cards, etc.

[1] http://www.google.com/wallet/

2.2.2 Android Wi-Fi and Bluetooth based Solutions

Since its establishment and especially now, when it is one of the most popular mobile platform on the market, Android platform is used by many for creation of home automation and home control systems. With the help of wireless technologies such as Wi-Fi and Bluetooth, customers are provided with innovative solutions that in all spheres make living easier. These solutions employ additional hardware components, sensors and network interfaces. They often use central server connected to the internet through which all communication is done. In addition to controlling of the home from a distance, these systems provide other advantages such as secured communication and lower power consumption in a home. Due to the fact that almost all Android smart phones are equipped with Wi-Fi and Bluetooth, these wireless technologies are the most used for the implementation of home automation and control systems. Many companies are already offering market ready solutions that are easily adaptable to existing architecture and devices within the home. In addition to necessary hardware components, they offer applications for mobile devices which are used by users to carry out the controlling or monitoring of devices and hardware.

In [PS12], Potts and Sukittanon present a home security solution involving Android smart phones and Bluetooth. Android smart phone is used to enable locking and unlocking the door and checking the status (locked or unlocked) of the door. Presented system uses home security controller consisting of Arduino Mega Board and a Bluetooth adapter. The home security controller, which is connected to the locking mechanism, receives commands from the Android smart phone via Bluetooth and controls the locking mechanisms or, respectively, checks the status of the locking mechanism. The authors of the paper also describe the process of implementing the Android application which is used by the user to control and to monitor the status of the door. The presented system is relatively easy to use, yet the main drawback is the short range of the Bluetooth and communication which can be interrupted by the barriers such as walls within the home. Several other papers, such as [RTI12], present similar projects involving Bluetooth and Wi-Fi for home automation. They report on needs for such solutions which are important to all general customers and extremely important for people with special needs or disabilities.

Wireless technologies in combination with the Android platform can also be successfully used for industrial purposes. As explained and illustrated in the paper [TV12], Android smart phone is used to control and monitor the CNC machine via Wi-Fi. The architecture of the system consists of the Android smart phone running the implemented application which provides GUI with controlling options and parameters. Central component of the system is the web server installed on the PC which controls the CNC machine. The web server provides services for sending and receiving of commands and responses from and to the Android smart phone. Furthermore, it uses socket communication to exchange the data with the CNC machine. Although the paper shows just basic operations which can be carried out remotely on a CNC machine, the idea of utilizing Android smart phones to control industry machines and processes, via Wi-Fi, could significantly improve the work of factories.

Based on presented projects and examples in which the Android smart phone platform is used in combination with a variety of wireless technologies, it can be concluded that this platform is not only well suited for implementation of solutions for home control and automation, but in general, for all smart environments. Each of the aforementioned wireless technologies that are used for such solutions has its advantages and disadvantages. For some of them, disadvantages are expressed by the time required to establish the communication. For some of them, implementation costs and power consumption are high. Lack of appropriate security mechanisms and protocols is also one of the disadvantages.

Within this master thesis and the implemented project, the focus is on NFC and its benefits. NFC provides quick set-up time and relatively fast data exchange. Construction and implementation costs for solutions which involve NFC can also be minimized, which will be shown on the example of this master thesis' project.

Chapter 3

Design

This chapter provides an overall description of the system and the system design. Furthermore, system's devices and components on which the project relies are introduced and described in general. Functional and non-functional requirements defined in the user requirements document are also outlined. System design is conducted based on the documented functional specifications and gathered user requirements. Based on the basic architecture of the system and defined main components, the work can be divided into three main parts:

1. Design and architecture of the NFC-I
2. Design of the Android application and libraries that will utilize the NFC-I
3. Design of the PC application and libraries which are used to demonstrate the visual appearance and functionality of target devices

Each of these parts is further divided and addressed in detail. Within the design phase, the focus is on basic principles which are relevant for the successful implementation:

- The purpose of each component and its role in the implemented system
- Physical structure, behaviour and interdependency of system components
- Determination of the connection between the components
- Definition and design of interfaces and communication protocols between the components
- Modelling of functions and processes
- Criteria for reliability, portability, scalability and performance

All these parameters and principles are taken into account and as a result of this part of the thesis, the final specification of the system, consisting of series of models, diagrams and explanations which accurately and in detail describe various aspects of the system is determined.

3.1 System Overview and Operation Principles

As already stated in the motivation section, the major goal is to design and implement the NFC-I which enables the NFC communication between an Android NFC-enabled smart phones and the non-NFC target devices (home appliances and consumer electronics). Figure 3.1 illustrates the principle behind this project and the simplified high-level system architecture:

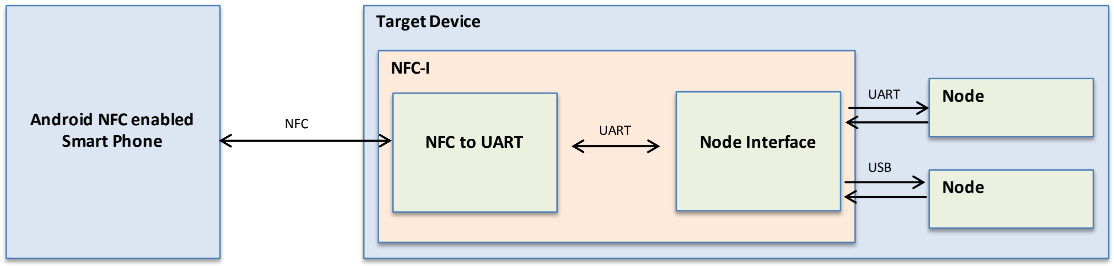


Figure 3.1: Simplified Architecture of the Implemented System.

As Figure 3.1 shows, the essential components and sub-components of the proposed NFC system are:

- **Android NFC-enabled smart phone:** NFC compatible Android smart phone runs implemented project specific Android application and enables user to communicate with the target devices oven NFC-I. For this purpose, **Samsung Nexus S** smart phone was chosen.

- **Target Device:** Target device represents a specific appliance (home appliance, consumer electronics or similar appliance) where the NFC-I is integrated. During the project, instead of real appliances, a software implementation on a PC, demonstrating appliances both visually and functionally are used.

- **Node:** Nodes represent integral components of the target device to which the NFC-I is directly connected. That are mostly: electronic circuits, electronic devices, sensors or actuators which perform certain tasks within the target device. By connecting the NFC-I to the target device's nodes, the user is allowed to operate them from the Android NFC-enabled smart phone. Furthermore, nodes consist of node elements, certain components or parameters which are controlled and maintained by the nodes. There are three types of node elements: control, monitor and configuration element.

- **NFC-I:** The NFC-I consists of two components: the first component, **NFC to UART Component (NtU)**, is responsible for the contactless communication with the Android NFC-enabled smart phone. The second component, **Node Interface Component (NodeInt)**, is responsible for the contact-based communication with the target devices and their nodes.

There are three main actor groups which interact or, alternatively, configure and adapt the implemented NFC system:

- **End users:** Users with minimal or no computer experience which utilize Android NFC-enabled smart phone and implemented Android application in order to interact with NFC-I enhanced target devices.
- **User interface designers:** Experienced users which are able to design additional user interfaces for Android applications, based on general user interfaces, provided instructions and requirements of specific target devices. Additionally, their role is to test the functionality of the implemented user interfaces and Android applications with other components of the system. The resulting user interfaces are intended for use by end users.
- **Hardware engineers:** Experienced users with medium hardware and software experience which intent to install and configure the NFC-I to the particular target devices and their nodes.

Depending on the type of target device which is used within the system, several use case scenarios, for the group of end users involving Android NFC-enabled smart phone, are synthesized and supported by the system:

- **Identifying:** The user is able to identify the target device upon moving the Android NFC-enabled smart phone to the place where the NFC-I is installed in the target device.
- **Monitoring:** It is possible to interactively monitor the status of the target device or, more specifically, node elements which have the characteristic that they can be monitored (e.g., status displays and timers).
- **Controlling:** Target device's nodes may posses the elements which can be controlled by a user (e.g., programmable selectors, knobs).
- **Configuring:** The target device or, respectively, the elements of target device's nodes can be configured to work in a specific mode (e.g., configuring of operating mode or program)
- **Storing and acquiring information about target device in off-line mode:** The user is able to store and acquire information (identification, information about nodes and their elements) about the target device, even if the device is off-line. The information about the target device is stored in the non-volatile memory of the NFC-I's NtU component.
- **Activation possibilities:** Before operating the NFC-I equipped target device, the user must activate the NFC-I to work with the attached target device. The activation process is also done with the help of Android NFC-enabled smart phone and involves secure authentication with the NFC-I. Further details on activating the NFC-I are given in the design chapter.

An example scenario is illustrated in the Figure 3.2. The scenario demonstrates the interaction between the end user operating Android NFC-enabled smart phone and the NFC-I enhanced washing machine as target device. The user reads the currently running washing program's remaining time, which is also indicated in the timer display of the washing machine:

1. Using the Android application which is particularly implemented to operate with the washing machine as target device, user pushes the button in the user interface which calls the method for getting the remaining time value of the currently running washing program on the washing machine.

2. The Android NFC-enabled smart phone initiates the NFC communication with the NFC-I sending information about the washing machine's node (in this case washing machine's control logic unit), information about the node's element (internal memory) and the command to read the currently running washing program's remaining time value from the internal memory.

3. The request is interpreted by the NFC-I, the right target device's node (control logic unit) and node element (internal memory) are identified and the request is executed.

4. The read remaining time value is transmitted back to the Android NFC-enabled smart phone over the NFC-I, and shown in the user interface of the Android application.

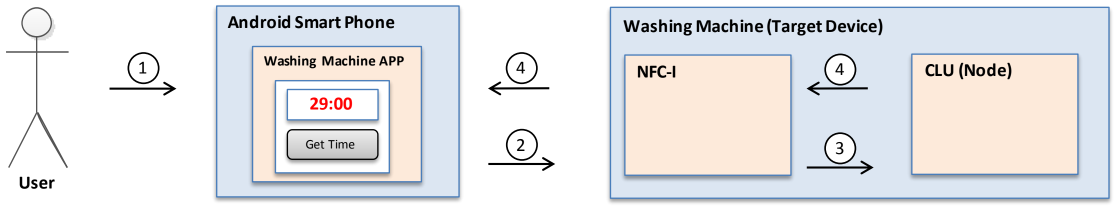


Figure 3.2: Use Case Scenario.

3.2 User Requirements

As the basis for the project plan and design phase, user requirements are defined for each component of the system separately.

3.2.1 NFC-I User Requirements

Basic user requirements for the NFC-I are identified and listed in the Table 3.1:

Nr.	Requirement description
1	NFC-I shall provide all necessary functionalities for enabling connection and communication between the Android NFC-enabled smart phone and the target devices.
2	Communication interface between the NFC-I and the Android NFC-enabled smart phone is to be NFC.
3	NtU component of the NFC-I shall provide an NFC communication interface for the communication with Android NFC-enabled smart phone.
4	Universal Serial Bus (USB) and Universal Asynchronous Receiver/-Transmitter (UART) communication interfaces shall be supported by the NFC-I's NodeInt component for the communication with the target device's nodes.
5	The communication interface between the two components of the NFC-I (NtU and NodeInt) shall be half duplex UART.
6	All mechanisms and methods which provide errorless communication flow within the NFC-I shall be implemented. This includes request and response buffering which shall provide no data loss.

Table 3.1: NFC-I: General User Requirements.

3.2.2 User requirements for Android Smart Phone and Implemented Applications

User requirements for the Android NFC-enabled smart phone and the implemented Android applications are identified and listed in the Table 3.2:

Nr.	Requirement description
General requirements	
1	The user shall be able to interact with the NFC-I equipped target device directly from the Android NFC-enabled smart phone.
2	The user shall be able to retrieve data about the target device, its nodes and elements, even if the target device is not identified yet (recognized correctly by the Android NFC-enabled smart phone).
3	It shall be possible to assign each NFC-I compatible target device's node a unique identification (type and ID number).

4	It shall be possible to access and list all possible internal elements or parameters of the target device's node.
5	The user shall be able to access configuration elements of the target device's node.
6	The user shall be able to monitor the internal elements of the target device's node.
7	The user shall be able to configure the target device's node according to provided configuration elements and configuration parameters.
8	The user shall be able to control the target device's node with the help of provided control elements and their options.
9	Data format shall be compatible across all target devices and their nodes.
	Activation of the NFC-I
10	Only an activated NFC-I shall provide access to the target device and its nodes.
11	The activation of the NFC-I shall be done using the Android NFC-enabled smart phone.
12	The deactivation of the NFC-I shall be done using the Android NFC-enabled smart phone.
13	The activation process shall involve secure authentication (password input) on the side of Android NFC-enabled smart phone.
14	The activation process shall be possible in on-line (target device is powered and running) and off-line (target device is not powered) operation mode of the NFC-I equipped target device.
15	Defined general functionalities of the target device shall only be available on an activated device.
	Android application requirements
16	The Android application shall provide graphical user interface which fulfils the requirements for interacting with the target device (input and output user interface elements).
17	Communication library providing support for general requirements (detection and identifying of the NFC-I enhanced target device, retrieving of data about the target device, its nodes and elements, activation and deactivation of the NFC-I enhanced target device, sending and receiving of commands and responses) shall be provided.
18	Beside the library which implements the general communication requirements, the application's architecture shall be, in general, designed to simplify the expanding of the application and to enable the adoption to different target devices with minimal effort.

Table 3.2: Android NFC-enabled Smart Phone: User Requirements.

3.2.3 Target Device User Requirements

Target devices, their nodes and node elements, and in general, their visual appearance and functionalities, are represented in software on a PC. Table 3.3 lists the requirements for the PC application which visualizes the target device:

Nr.	**Requirement description**
1	The application shall provide powerful graphical user interface which leaves an impression on end users as if they are interacting with real target devices.
2	A library which provides functionalities for connecting to the NFC-I and performing communication with the NFC-I (sending and receiving of commands and responses) shall be provided.
3	The application shall be designed to simplify the expanding and implementation of additional target devices, with minimal effort.

Table 3.3: Target Device: User Requirements.

3.3 Data Transfer Within the Proposed NFC System

At the system level, data is transferred in byte blocks, so called Application Protocol Data Unit (APDU) blocks. The structure and characteristics of an APDU are defined by ISO/IEC 7816-3 [ISO05]. There are two types of APDUs:

1. Command Application Protocol Data Unit (C-APDU)
2. Response Application Protocol Data Unit (R-APDU)

Diagram in the Figure 3.3 depicts the data flow in the whole system. User request, packed in C-APDU, is sent from the Android NFC-enabled smart phone over the NFC-I to the target device's specific node where the command is interpreted. After the interpretation is done, node sends the R-APDU containing the response information over the NFC-I to the Android NFC-enabled smart phone. The response is interpreted in application specific way. The target device and its nodes always have the role of the slave. Thus, they wait for the C-APDUs coming from the Android NFC-enable smart phone and send the R-APDUs back.

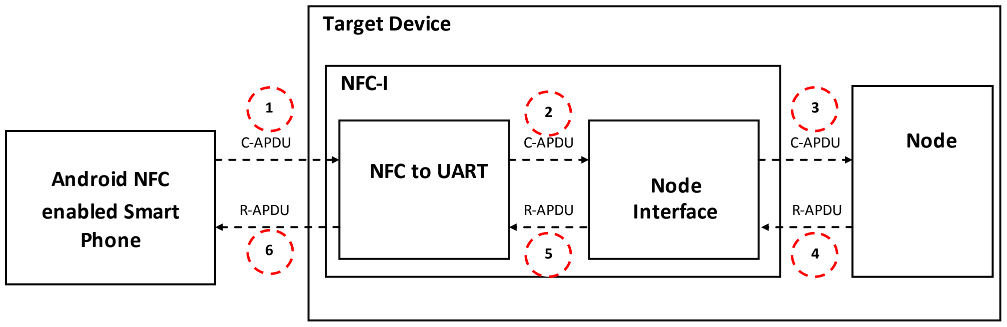


Figure 3.3: Data Transfer within the System.

Figures 3.4 and 3.5 show the structure of the C-APDU and R-APDU with their mandatory and optional parts:

CLA	INS	P1	P2	Lc	Data	Le
Class byte	Instruction byte	Parameter byte 1	Parameter byte 2	Length of the data bytes	Data bytes	Expected length of the R-APDU
Header - Mandatory				**Body - Optional**		

Figure 3.4: C-APDU Structure.

Data	SW1	SW2
Data bytes	Status word 1	Status word 2
Response Body - Optional	**Response status bytes - Mandatory**	

Figure 3.5: R-APDU Structure.

3.4 Design of the NFC-I

As it is stated in the introduction chapter, the main task of the NFC-I is to provide functionalities for establishing communication and performing data exchange between the Android NFC-enabled smart phone and the target devices. Part of this project is also the construction of the NFC-I, where the components of the NFC-I which best meet the required specifications are chosen. This section deals with these issues, it shows how the NFC-I is constructed, how the communication between the components is realized and in which way the NFC-I communicates with the Android NFC-enabled smart phone and the target device.

3.4.1 Selection of Components

The defined user requirements are used as the basis for the design of the system and selection of the hardware components used for the construction of the NFC-I. In addition to specified requirements, power consumption constraints, performance of the components and, especially, their costs are taken into the account during the selection. The chosen sub-components of the NFC-I are:

- **NFC to UART component (NtU)**: Infineon Security Microcontroller (**M7794**)
- **Node Interface (NodeInt)**: Texas Instruments **LaunchPad** featuring **MSP430G2553** (MSP430) microcontroller.

First integral component of the NFC-I is NtU which provides the NFC communication with the Android NFC-enabled smart phone. For this purpose, a dual-interface security microcontroller provided by Infineon was chosen. M7794 security microcontroller is typically used for personal and government identification solutions, payment solutions, authentication, mobile communications and security in general. The most important characteristic of the microcontroller are:

- Two communication interfaces: contact-based (ISO/IEC 7816 defined) and contactless communication interface (ISO/IEC 14443 A/B and ISO/IEC 18092 defined)
- Support for Mifare
- Crypto coprocessor featuring RSA up to 4096-bit and ECC up to 521-bit
- 240KB flash storage
- Non-volatile Memory (NVM)
- 16-bit CPU
- 6KB XRAM

For the purposes of interfacing the NFC-I with the target device and its nodes, the LaunchPad development board [Ins12a], provided by Texas Instruments is used as NodeInt.

As depicted in Figure 3.6, the LaunchPad features socket that supports all MSP430 family 20 pin microcontrollers. The microcontroller can be programmed and debugged with the help of the USB interface. LaunchPad features also programmable LEDs and buttons.

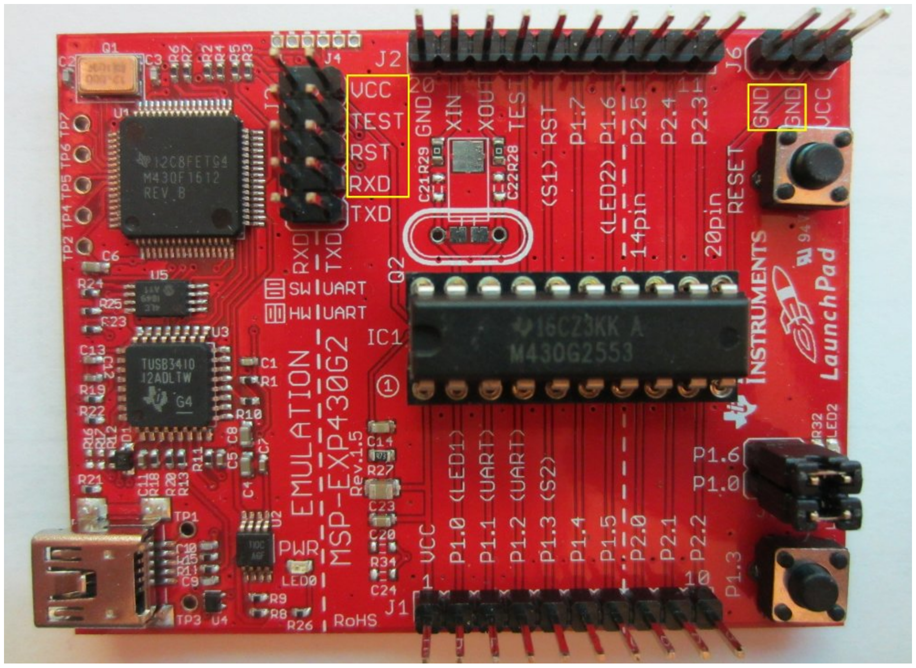


Figure 3.6: Node Interface: TI LaunchPad.

MSP430 family mixed signal microcontrollers are particularly well known for their low cost and low power consumption. Furthermore, they provide a large number of peripherals and other hardware features. Considering the needs of the project, the choice was made to use MSP430G2553 [Ins12b]. General characteristics and features of this microcontroller are:

- Five power modes
- 16-bit RISC architecture
- Fast wake up from standby mode
- 16KB flash storage
- 512B SRAM
- Clock frequency scalable up to 16 MHz
- Two 16-Bit Timers with capture/compare registers
- 16 GPIO pins
- Universal Serial Communication Interface (USCI) module
- Analog-to-digital converter

Figure 3.7 illustrates the functional block diagram of the MSP430G2553 microcontroller with all mentioned features:

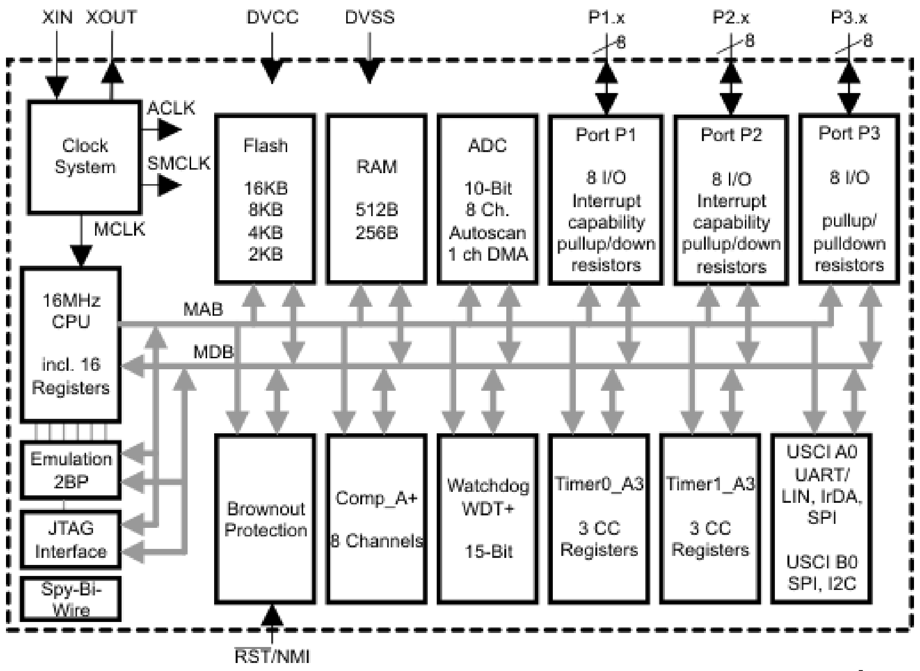


Figure 3.7: Node Interface: MSP430G2553 Functional Block Diagram [Ins12b].

Figure 3.7 depicts the detailed deployment model of the implemented system. NFC-I and its components are highlighted in green color.

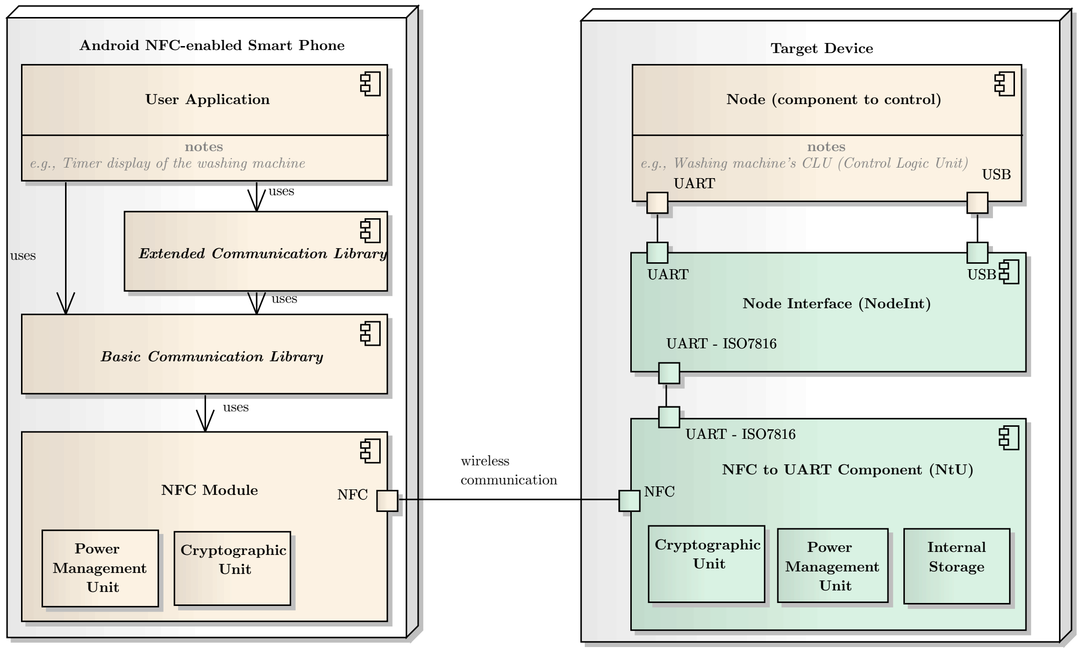


Figure 3.8: Deployment Model of the Implemented System: NFC-I.

3.4.2 NFC to UART Component

As its name suggests and as it can be seen in the Figure 3.8, the NFC to UART component (NtU) is used to establish the communication with the Android NFC-enabled smart phone over the contactless NFC interface and to transfer data from the Android NFC-enabled smart phone to the Node Interface (NodeInt) component. Furthermore, data received from the target device over the Node Interface (NodeInt) is transferred by the NtU to the Android NFC-enabled smart phone. The NtU communicates with the NodeInt via contact-based interface, using the UART communication protocol.

Theoretical Background

Smart cards in general can be classified into memory smart cards and microcontroller (chip) cards. Taking into account the nature of data transfer, contact-based and contactless cards, or smart cards which feature both communication interfaces can be distinguished. NtU is a smart card with an integrated circuit chip featuring two communication interfaces. During the master thesis, otherwise differently stated, the name security controller and the name NtU relate to microcontroller (chip) card. The NtU is based on the type 4 tag platform [For11] which uses NFC Type A and Type B signalling schemes. ISO standards ISO/IEC 7816:1-15 and ISO/IEC 14443:1-4 define the most important characteristics of smart cards.
According to ISO documentation [ISO98], four most important standards related to physical characteristics and contact-based communication are:

- ISO/IEC 7816-1: describes the physical characteristics such as size of a smart card.
- ISO/IEC 7816-2: determines the size, location and the function of the pin contacts on a smart card.
- ISO/IEC 7816-3: describes the electrical signals and communication protocols between a smart card and a reader. This standard declares also the operating voltage, current values and duration of the signals. Furthermore, it describes the basic communication protocols: T=0 and T=1 for communication with a reader.
- ISO/IEC 7816-4: defines the structure and content of messages, commands and responses which are transmitted between a smart card and a reader.

ISO/IEC 14443 standard defines the characteristics for contactless smart cards operating on 13.56 MHz frequency[ISO08]:

- ISO/IEC 14443-1: describes the size of a smart card and its physical characteristics.
- ISO/IEC 14443-2: describes the contactless communication interface and protocols for both modulation schemes: Type A and Type B.
- ISO/IEC 14443-3: determines the initialization procedures and anti-collision protocols for Type A and Type B cards.
- ISO/IEC 14443-4: defines the data transmission protocols, structure and content of messages which are transmitted between a smart card and a reader. Same protocols are capable to transfer the messages which are defined by ISO/IEC 7816-4 standard.

NtU Application Design Principles

Communication between the Android smart phone and the NtU is based on the ISO/IEC 14443 standard with Type B modulation scheme. NFC Type B is fully supported by the Android NFC APIs without any restrictions. As the basis for the implementation of necessary functionalities of the NtU, an Application Note, which was made available by Infineon, is used. The provided Application Note implements many necessary functionalities including: card and multi-interface initialization, anti-collision protocols for Type A and Type B, memory management, contactless and contact-based (UART) transmission protocols and interpreter function for received commands on the contactless interface.
For the purposes of in this master thesis proposed NFC system, few modifications and few functionalities are added to the provided Application Note. The most important improvements and functionalities which are added are:

- Changes regarding the interfaces (contactless and contact-based) and their initialization
- Functionality for accessing the NtU's NDEF tag application
- Implementation of a function which provides the main functionality: sending of received C-APDUs to the NodeInt and transmitting of R-APDUs to the Android NFC-enabled smart phone
- Implementation of a function for accessing (writing and reading) the NVM of the NtU

When the NtU gets power through one of two interfaces, it enters the initialization and anti-collision processes. After that, the application goes through interface initialization process. Depending on whether the NFC-I is attached to the target device and whether the magnetic field is provided by the Android NFC-enabled smart phone, two operating modes of the NtU can be distinguished:

1. **CL-CB mode (default mode):** The NtU is powered over contact-based interface. If available, operating frequency provided by the Android NFC-enabled smart phone is selected as primary clock and clock provided from the NodeInt over the contact-based interface is set as fall-back clock. Clock frequencies provided by the Android NFC-enabled smart phone and the NodeInt are used by the NtU's peripherals in order to enable the communication. However, the NtU main system clock operates at 30 MHz.
2. **CL mode (off-line mode):** The NtU is powered only by the magnetic field of the Android NFC-enabled smart phone. This means that the NFC-I is not attached to the target device or that there is no power on contact-based interface.

Regardless in which operating mode the NtU is running, the user, with the help of the Android NFC-enabled smart phone, always has access to the NVM and the NDEF tag application.

3.4.3 Node Interface Component

Node Interface component (NodeInt) is used to transfer data from the NFC-I to the target device and conversely. In this process, the NodeInt receives commands forwarded from the NtU and sends them to the appropriate node of the target device. The command is followed by the response from the target device's node, which is transmitted back to the NFC-I and forwarded to the Android NFC-enabled smart phone.

The main task of this component is to provide support for a large number of contact-based interfaces, through which the NFC-I could communicate with the target devices. The base of the NodeInt makes the TI Launchpad equipped with the MSP430G2553 microcontroller. The Launchpad is very suitable for these purposes, mainly because of the small size, numerous features and affordable price. All the features such as USB interface (provided by the Launchpad) and USCI module which provides three serial communication modes: UART, Serial Peripheral Protocol (SPI) and Inter-Integrated Circuit (I2C), make the device a perfect universal solution for the interfacing with the target devices. Also, given that all these features are implemented on a single chip, the possibility of errors in design and connection is reduced.

Basics about MSP430G2553 Microcontroller

The most important features of the MSP430G2553 which are essential for the realization of the project are described here.

Clock Generator: According to specification of the MSP430G2553 Microcontroller [Ins12b], clock module can generate clock signals from the integrated Digitally Controlled Oscillator (DCO), or using the external crystal oscillator. The module can generate three clock signals: Auxiliary clock (ACLK) - intended for use by peripherals, main clock (MCLK) - CPU main clock and sub-main clock (SMCLK) - clock for peripheral modules. The internal DCO is programmable to run the frequency up to 26 MHz depending on a voltage supply. Yet the calibrated frequencies are in range between 1 and 16 MHz.

I/O Ports: The microcontroller has two 8-bit General Purpose Input/Output (GPIO) ports. These are port P1 and port P2. All pins are multiplexed and have additional functionalities depending on the peripheral module which is used. Each of the pins can be configured to be input or output pin regardless of how the other pins on the same port are configured. Each pin features also a pull-up/pull-down resistor.

USCI Module: The microcontroller features one peripheral USCI communication module for the exchange of data with other electronic devices such as other microcontrollers. The module can work in two regimes, i.e. it supports synchronous communication protocols: SPI and I2C and asynchronous communication protocol UART. The data between the NodeInt and target device's nodes is exchanged over UART, using T=0 (character based) serial communication protocol.

Timers: Generally, timers are used for measuring time and counting external events. They generate interrupts when transitioning from one condition to another depending on the provided clock and state of the condition. Output compare registers and their contents are compared with the content of the counter. When their contents are conflated, timer flags are set and if the interrupts are globally enabled, the interrupt routines of the timers are executed. The MSP430G2553 features two timer modules: Timer0_A3 and Timer1_A3. Both of them are 16-bit with capture/compare registers and support for interrupts. Timers are used within the project to implement the single-wire, half-duplex software UART (T=0 character based) communication with the NtU component.

Hardware Connection between the NodeInt and the NtU

This section is dedicated to the hardware configuration, i.e., defining the connection between the NodeInt and the NtU. Figure 3.9 illustrates the hardware connection and pin assignment between the two components:

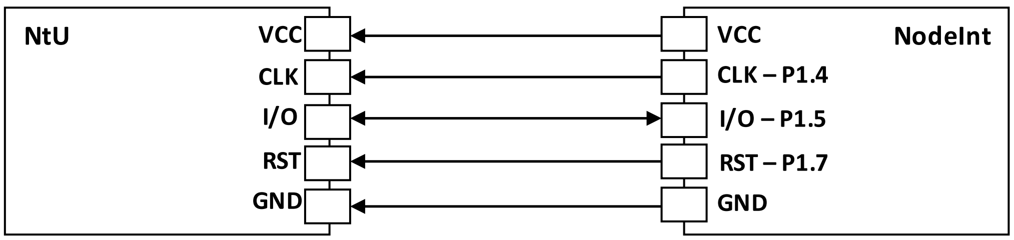


Figure 3.9: Hardware connection between the NtU and the NodeInt.

NtU is mainly powered by the magnetic field of the Android NFC-enabled smart phone and additionally by the NodeInt in order to be able to use the UART module. VCC, CLK, RST and GND are input from the NodeInt to the NtU component. Data exchange pin, I/O, is output and input for both components. NtU component is the master and initiates the communication and data transfer. This prevents situations where both components are transmitting data at the same time. This type of interfacing and communication is based on standard defined in ISO/IEC 7816-3 [ISO06].
Table 3.4 shows detailed information about used pins and their functions:

PIN	**Function**	**Used for**
VCC	Output voltage	NtU
GND	Ground	NtU
CLK P1.4	Output clock (frequency)	NtU
I/O P1.5	Data pin (half-duplex)	NtU
RST P1.7	Reset	NtU

Table 3.4: Node Interface: Pin Assignment Details.

Communication between the NodeInt and the NtU

Communication between the NtU and the NodeInt is subject to ISO/IEC 7816-3 [ISO06] standard. Generally, the communication which is used between these two components is a half-duplex UART, using only one wire for communication. The communication protocol requires I/O communication line which supports external interrupts and timer capture/compare functionalities. These requirements are met by using TI Launchpad featuring MSP430G2553 microcontroller. The standard, in this type of communication, defines two transmission protocols: T=0 (character based protocol) and T=1 (block based protocol). The T=0 transmission protocol is chosen for this communication because of the compatibility with the NtU and the provided implementation.

Data is transferred one bit at a time and a frame format, for a character, consists of: 1 start bit, 8 data bits, 1 parity bit and 2 stop bits. Frame format is illustrated in the Figure 3.10:

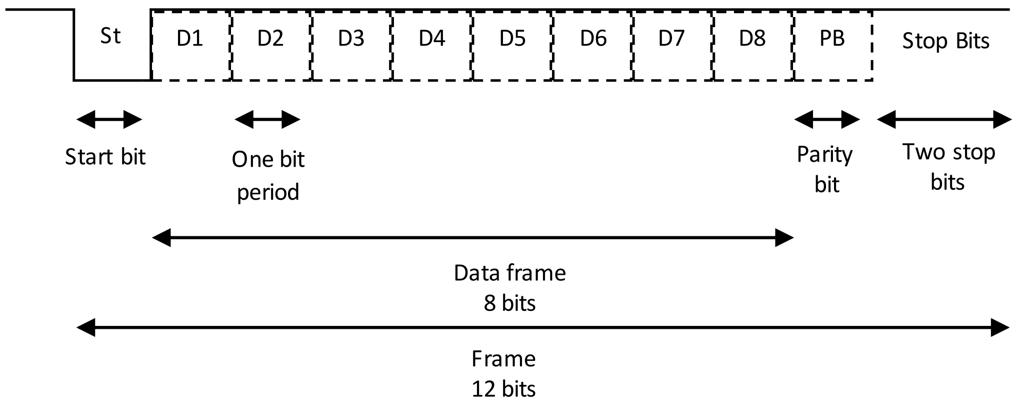


Figure 3.10: Node Interface: UART Frame Format.

In asynchronous half-duplex communication, the data transmission rate is defined by Elementary Time Unit (etu), which is a duration of one bit transmitted over the I/O line. The etu is obtained by (3.1):

$$1etu = \frac{F}{D} \times \frac{1}{CLK} \tag{3.1}$$

As the equation shows, bit duration depends on the clock frequency (CLK) and two parameters: F - clock rate conversion factor and D - bit rate adjustment factor. These parameters have predefined values which can be found in the ISO 7816 documentation [ISO06].

Hardware Connection and Communication between the NodeInt and the Target Device

NodeInt communicates with the target device's nodes with the help of USCI module which is configured to support UART communication. Unlike the communication between the NodeInt and the NtU, the proposed communication is a full-duplex hardware UART based communication. There are many advantages in relation to a version of the software UART which is implemented with the help of timers. Beside the fact that the communication is achieved over two lines which significantly accelerates the data transfer, the implementation is quite simplified. Furthermore, the module supports automatic baud rate detection.

In order to enable the hardware UART, J3 TXD and RXD jumpers on the Launchpad must be changed to horizontal position, as shown in the Figure 3.11.

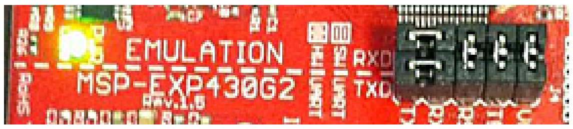


Figure 3.11: Node Interface: Hardware UART Jumper Setting.

Data frame format for a transmitted character, as shown in the Figure 3.12, consists of: 1 start bit, 8 data bits and 1 stop bit:

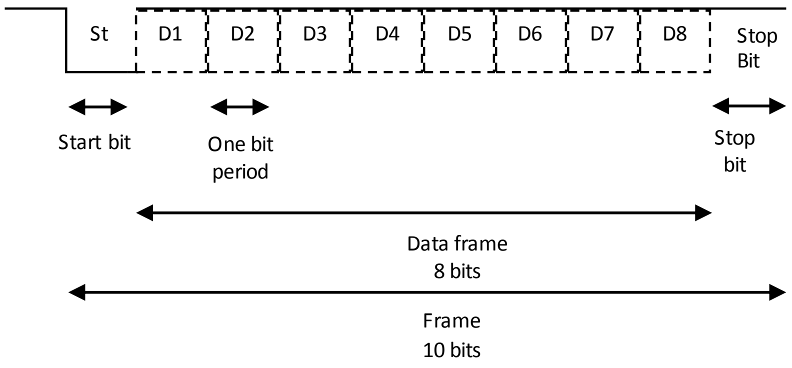


Figure 3.12: Node Interface: Hardware UART Frame Format.

Launchpad features also an UART-USB converter, and communication with the target device's nodes can either be done via USB or by wiring the node directly to the Launchpad. In the second case, the UART transmission pin (P1.1) from the Launchpad is wired to the reception pin of the node, and the Launchpad's UART reception pin (P1.2) is wired to the node's transmission pin. This type of connecting is illustrated in Figure 3.13.

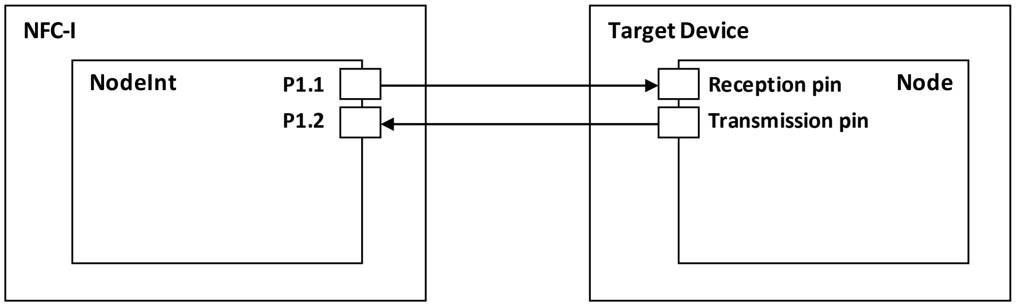


Figure 3.13: Hardware connection between the NodeInt and the Node.

3.4.4 Data Buffering

Each component of the system forwards the received commands and responses to the component next in the line. Components that constitute the NFC-I play a major role in data transfer and therefore must provide data buffering mechanisms. Buffering for the

NtU is already provided by the Application Note, thus in the master thesis, the design and the implementation of command/response buffering for NodeInt is discussed.

The proposed solution involves two ring buffers, one for storing C-APDUs and one for storing R-APDUs. Both buffers have fixed length and a pair of global index variables for keeping track of writing and reading into/from buffer.

3.4.5 NFC-I General Information Record

As a prerequisite for using the NFC-I with other components of the system (Android NFC-enabled smart phone and the target device), it is necessary to provide the NFC-I with information about the used target device and its nodes. This information (*General Information Record*) consists of the unique identifier of the target device, the list of nodes that are available, as well as lists of control, configuration and monitoring elements for each node. In order to fulfil the prerequisite, the information is defined according to the NDEF specification [For11] and it is stored in the NtU's NDEF tag application as NDEF message.
Table 3.5 shows and explains the parts of the *General Information Record*:

Activation flag	A flag indicating if the target device is activated or not
Identifier	A keyword for the NFC-I needed to identify it as NFC-I (a default term can be used)
Type of the NFC-I	Type of the NFC-I (e.g., name of the target device)
Node List	List of available nodes with their unique numerical identifiers. The node list is separated through ';'. The numerical identifier and the name of the node are separated by ':'
List of control, configuration, and monitor elements of a node	List of control, configuration and monitor elements of all target device's nodes. The list is separated through ';'. The numerical identifier, element's name and the type of the element are separated by ','. The first term of the list is the identifier of the node

Table 3.5: NFC-I: General Information Record.

3.5 Design of the Android Application

Android NFC-enabled smart phone within the proposed system is used as a device through which users interact with target device. Mediator in this interaction is the NFC-I. As deployment model in the Figure 3.14 shows, the most important internal component of the Android NFC-enabled smart phone is the NFC module which enables the NFC communication with the NFC-I. As for software components, two communication libraries, the main application and the user application are presented and designed. Figure 3.14 also describes the basic relationship between the software components and the NFC module.

The design of software components is based on the requirements set out in Chapter 3.1. Android NFC-enabled smart phone featuring NFC module, which is used for the project, is Samsung, model Nexus S [Sam11]. This section provides the conceptual solution and organization of necessary libraries and applications.

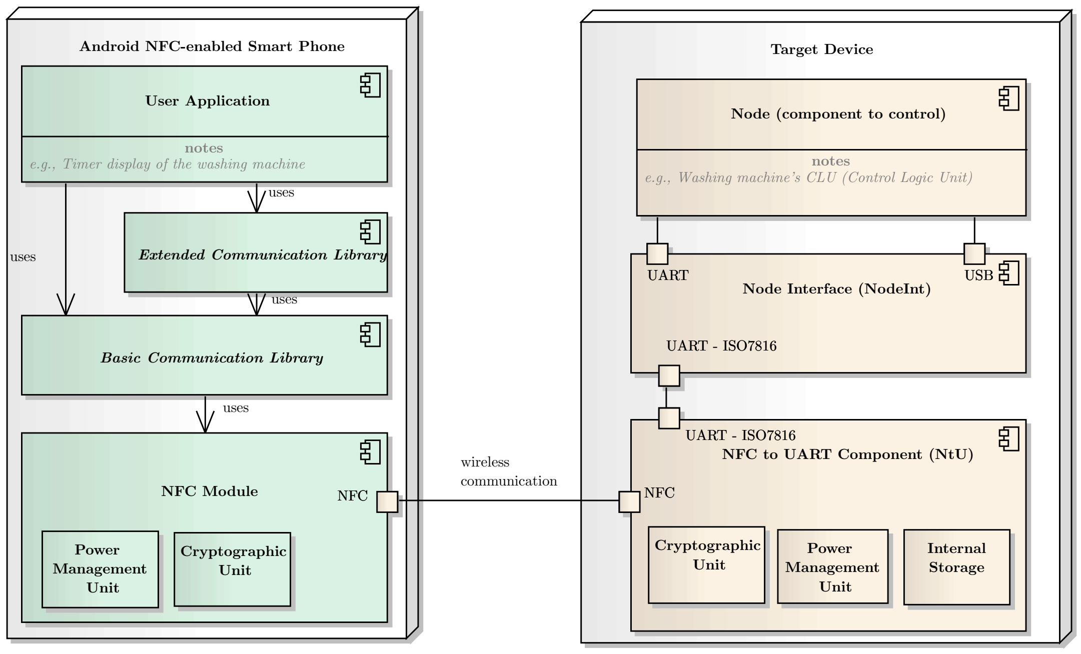


Figure 3.14: Deployment Model of the Implemented System: Android NFC-enabled Smart Phone.

3.5.1 Basic Communication Library

The main task of the *Basic Communication Library* is to provide all necessary functionalities for using the NFC module and establishing communication between the Android NFC-enabled smart phone and the NFC-I enhanced target device. This library implements all general requirements which are predefined in Chapter 3.1. This includes:

- Detection and connecting to the NFC-I enhanced target device
- Activation of the NFC-I and activation verification
- Reading and writing of *General Information Record* which contains information about the target device
- Realisation of data exchange (sending/receiving of C-APDUs and R-APDUs) between the Android NFC-enabled smart phone and the NFC-I enhanced target device

Library interacts directly with the NFC-I and it is designed as an independent component which can be used as the basis for the implementation of specific applications and libraries for interaction with the target devices.

3.5.2 Extended Communication Library

The main purpose of the *Extended Communication Library* is to allow developers to implement additional features and functionalities that are specific to a particular user application or a group of user applications. Everything that requires direct communication with the NFC-I and is not part of the generalized functionalities which are contained in the *Basic Communication Library*, is located in this library. By this approach, the *Basic Communication Library* in further modifications and adjustments for different target devices remains unchanged. The additional features and functionalities which can be implemented as part of the *Extended Communication Library* are typically related to operations that require certain processing of data received from the target device and its transmitting to the user application.

3.5.3 Main Application

During the design phase of the Android application, special attention is paid to fundamental components of the Android system which are used for the realization of the application. The most important aspects for the design of the application that have been taken into account are:

- Definition of an *Activity* and possible states of *Activity* within the application
- Definition of *Intents*
- Structure of the Android manifest file

Designed Android application consist of a main class which inherits the Android *Activity* class. An activity can be identified as the main routine in the application and that what creates the window of the application using the defined user interface. An application may have one or more activities, yet only one can be defined as primary. *Activity* class implements methods which are responsible for transitions between different states of an *Activity*, and that is the starting point of the application.

Android Manifest File

For every Android application an Android manifest file must be specified. An Android manifest file contains information relevant for the Android system to successfully run an application. This description, structured as a XML file, contains following information:

- General information about the used Android version
- Name of the application
- Definition of permissions for accessing the individual smart phone's hardware components
- Definition of application's *Intents*

3.5.4 User Application

User application represents the target device specific part of the application. An example is the application which controls the washing machine as the NFC-I enhanced target device. User application consists of the graphical user interface and the target device specific implementation which are realised with the help of the *Basic* and the *Extended Communication Library.*

3.6 Design of the Target Device Application

This section provides a basic overview of the idea behind the design of the application which represents a target device. In order to fully demonstrate the operational and functional capabilities of the implemented NFC system and the NFC-I as the central component, target devices, their operational functionalities and appearance are visually represented and implemented in software. This approach enables transparent testing of the entire NFC system, without the need to research through a detailed architecture and communication capabilities of real target devices.

NFC-I is connected via USB to the PC and the implemented application running on the PC represents the target device. The application consists of two main parts. First part is the communication library which provides all necessary functionality for using the NFC-I and establishing the communication between the PC and the NFC-I. Second part is the user program. User program consists of main application and user interfaces which visually represent target device's operations and appearance.

3.6.1 NFCI Communication Library

The *NFCI Communication Library* provides all necessary functionalities for communicating between the PC application and the NFC-I. PC and the NFC-I are physically connected via an USB interface, and the NFC-I is recognized by the operating system as a device connected on a serial communication port (COM). The library provides basic methods for creating, opening and closing of the serial communication port. Furthermore, it provides functionalities for data exchange between these two entities.

3.6.2 Main Application and User Interfaces

The primary role of the main application is to provide functionality for selecting between two or more user interfaces which demonstrate the target devices and to execute the selected user interface.

Target devices and their appearances are visually represented by user interfaces. For their functionalities and operations, specific implementations are provided. In order to increase the user experience of the applications which represent target devices, user interfaces are designed to closely resemble the appearance and functionalities of real target devices. As a result, simple, functional and visually appealing graphical user interfaces are created. Due to a fact that each target device features its own functionalities and operations, user interfaces are created separately for each target device.

Chapter 4

Implementation

Following the design concept, this chapter describes the realization of the software components and steps in the development process in detail. The chapter is structured similarly as the previous one, consisting of three main parts: implementation of the NFC-I software, implementation of the Android application and implementation of the PC application which represents the target device. All three parts are substantiated with flow charts, code listings and explanations. Furthermore, tools and development environments used for the project are covered and explained.

4.1 Tools and Development Environments

Implementation of the project was conducted on the Windows 7 32-bit workstation. The software for the NodeInt component is programmed in `C`, using the Eclipse-based IDE Code Composer Studio (CCS) that supports all MSP430 family microcontrollers. A limited free version of the CCS is available in Internet[1]. The main features of this IDE are that it provides compiling and debugging of the generated code. In addition, during the debugging, the insight in the state of registers and memory is enabled. CCS also provides the drivers for the USB interface of the Launchpad, which is used to upload the implemented applications to the microcontroller.

Implementation of the NtU is based on the provided Application Note and it is conducted in `C`, using the Keil IDE[2]. Furthermore, Smart Card Manager, in combination with the Duali DE-620[3] smart card reader/writer, is used instead of the Android NFC-enabled smart phone to facilitate the testing process.

Android development environment consists of JDK7, Eclipse IDE and Android SDK 4.1.2 (API16). All Android related programming tasks are performed in `Java` programming language. Detailed procedure for installing and configuring Android development environment is given in the appendix.

[1]http://www.ti.com/tool/ccstudio-msp430
[2]http://www.keil.com/uvision/
[3]http://www.duali.com/goods_detail1.php?goodsIdx=269

Applications that demonstrate target devices are implemented in `C#` programming language, using the Microsoft Visual Studio 2010. This combination proved to be the best solution, because it enables direct access to serial communication interface, without the use of additional libraries.
Figure 4.1 outlines the preparatory work and performed tasks during design and implementation phase. Preparatory work is conducted in the first chapter of the thesis. As a result of the preparatory work, detailed design documentation is created. Based on the design decisions, implementation is conducted in three parts. In the last step, the use case implementation is conducted.

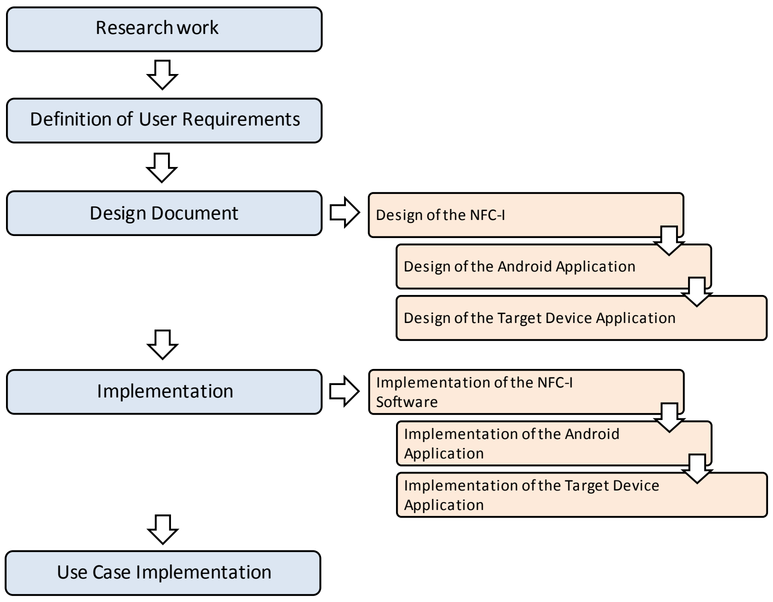

Research work
Definition of User Requirements
Design Document
Design of the NFC-I
Design of the Android Application
Design of the Target Device Application
Implementation
Implementation of the NFC-I Software
Implementation of the Android Application
Implementation of the Target Device Application
Use Case Implementation

Figure 4.1: Steps Performed during Design and Implementation Process.

4.2 Implementation of the NFC-I

As explained in the design phase, the NFC-I consists of two components, NtU and NodeInt. Particular attention, in this section, is paid to hardware settings, realization of communication between the NtU and the NodeInt and implementation of communication between the NFC-I and the target devices. A large part of the implementation for the NtU was provided by the Application Note, so only the rest of the implementation is described here. Software architecture is specified by flow charts of different modules and routines.

4.2.1 NodeInt Main Routine and Hardware Setup

The main routine of the NodeInt application consists of function calls for the preparation of hardware, calls for initialization of data reception buffers and an infinite loop, which performs the checking if a C-APDU was received from the Android NFC-enabled smart phone, or if a R-APDU was received from the target device. Received APDUs are transferred further to the appropriate component. The main loop is paused by the interrupts which are called when the application is receiving or sending the APDU. The whole process is depicted by the flow chart in the Figure 4.2:

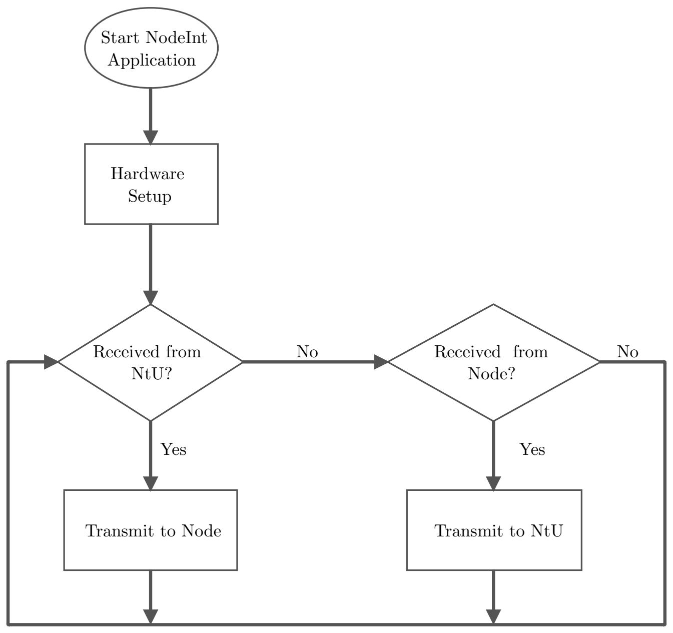


Figure 4.2: NodeInt: Main Routine.

In Hardware Setup process, the initial configuration of the NodeInt is completed. This process consists of configuring the pins of the NodeInt and setting of used register values. MCLK (CPU main clock) and SMCLK (peripheral clock) are set to 16 MHz, Watchdog timer is disabled and global interrupts are enabled. The following settings that apply to half-duplex UART communication with the NtU component are performed: SMCLK clock output on P1.4 (CLK) pin is enabled, the I/O pin is set to input and falling edge interrupt on the I/O pin is activated.

As for the settings for the communication with the target devices, the following configuration is done: the USCI module is configured to work in UART mode, pins P1.1 and P1.2 are configured as data transmission and reception pins, the SMCLK clock is used by the USCI and the interrupt on the reception pin is enabled. Settings concerning the SMCLK clock of 16 MHz which is supplied to the NtU and the USCI module, result in a bit duration of 372 and baud rate of 9600.

The Hardware Setup process also comprises the definition and initialization of ring data buffers with a default size of 180 characters.

4.2.2 NtU Main Routine and Multi-interface Initialization

The flow chart illustrated in the Figure 4.3 depicts the main program of the NtU. The main routine operates in the infinite loop. After the multi-interface initialization, the NtU enters the sleep state (low power consumption mode where the CPU is off). The NtU remains in the sleep state until the command from the Android NFC-enabled smart phone is received.

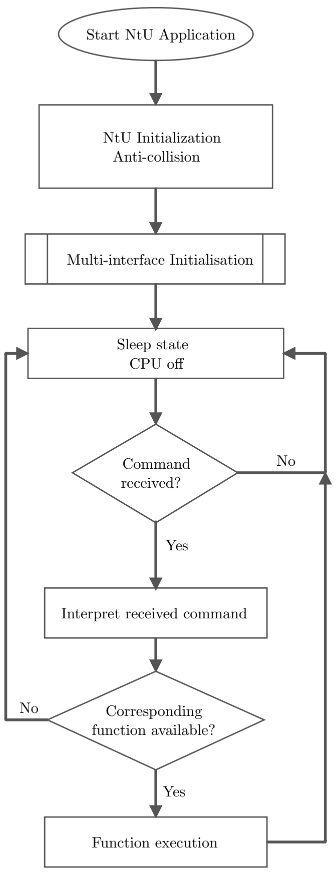


Figure 4.3: NtU: Initialization and Main Routine.

All commands that the application can accept are predefined, and those that can not be recognized are automatically rejected. Upon receiving the command (C-APDU) from the

Android NFC-enabled smart phone, the command is interpreted, i.e. it is checked whether there exists a function which corresponds to the received command.

For the purposes of the project, beside the default functions that are implemented in the Application Note, a function, which provides the main functionality of the NtU is implemented. Depending on the content of the received C-APDU, the function transmits the C-APDU to the NodeInt or the writing and reading of the internal NVM memory is executed.

Prior to entering the main routine, NtU goes through the process of interface initialization which is illustrated by the flow chart in the Figure 4.4.

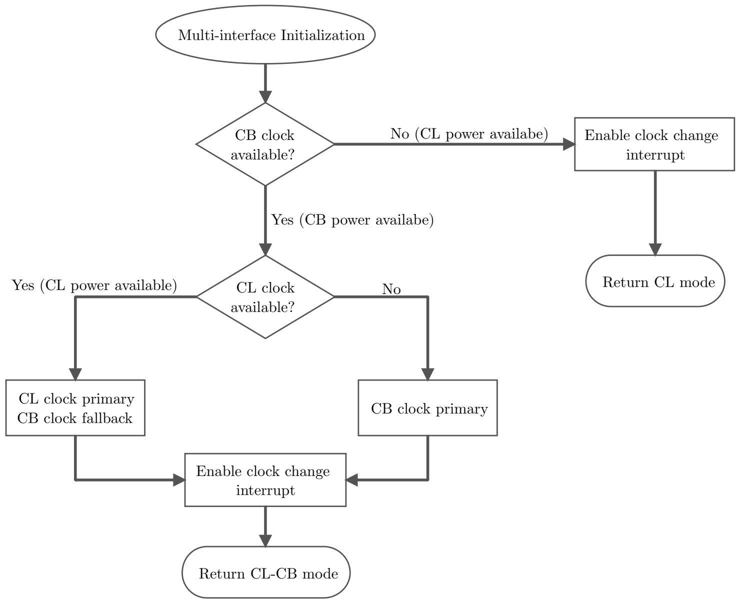


Figure 4.4: NtU: Multi-interface Initialization.

The result of this process determines whether the NtU is going to work in default (CL-CB) mode or in the off-line (CL) mode. In the default operating mode, both interfaces are functional and thus, the component is able to carry out its main task. In the off-line mode, the limitations are confronted. The enabled interface in this mode is the contactless interface and therefore, only operations concerning the writing and reading of the NtU's NVM are available. Furthermore, the access to the NtU's tag application and the *General Information Record* is enabled. This process also implements the interrupt handler, and any new event, caused by clock change on any of two interfaces, leads to reinitialization of the interfaces.

Command-Response Function

This function, as depicted by the flow chart in the Figure 4.5 utilizes both interfaces and implements the main task of the NtU component. Basically, in the function, the received C-APDU from the Android NFC-enabled smart phone is transmitted to the NodeInt and upon receiving the R-APDU from the NodeInt, the R-APDU is transmitted to the Android NFC-enabled smart phone.

In addition, the function distinguishes two more types of commands which refer to writing into the NVM memory and reading from the NVM memory. This feature allows the user to store the additional information about the target device in the NVM memory, which can be retained even if the NFC-I is not powered.

Identification of the received C-APDUs and finding of the appropriate functions corresponding to the received commands is done by checking the Class and the Instruction byte of the received C-APDU. By default, the Class byte for all C-APDUs is 0xC0. The Instruction byte determines which function shall be executed. For this function, the Instruction byte is fixed to 0xF2. Parameter bytes (P1 and P2) of the C-APDU regulate further filtering within the function. Insofar as the first Parameter byte (P1) is not 0xFF, the received APDU is transmitted to the NodeInt. Yet if the P1 is 0xFF, the C-APDU is about writing or reading of the NVM. The second parameter byte P2 determines if the C-APDU concerns the writing (P2 is 0x01) or reading (P2 is 0x00). Data bytes of the APDU contain information which is stored in the NVM.

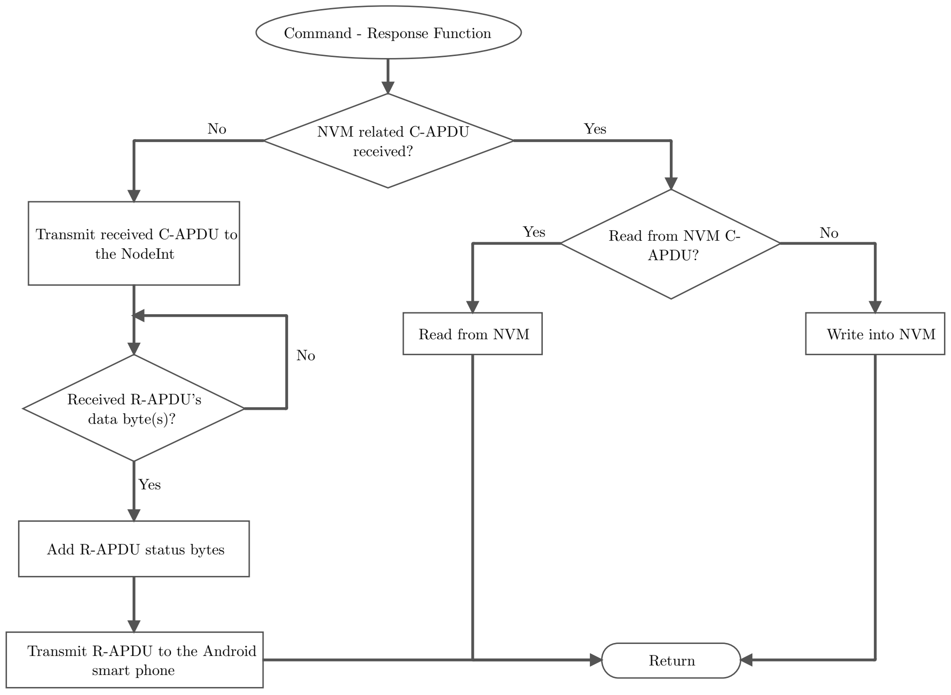

Command - Response Function
NVM related C-APDU received?
No
Yes
Transmit received C-APDU to the NodeInt
Read from NVM C-APDU?
Yes
No
Read from NVM
Write into NVM
No
Received R-APDU's data byte(s)?
Yes
Add R-APDU status bytes
Transmit R-APDU to the Android smart phone
Return

Figure 4.5: NtU: Command-Response Function.

Based on the received C-APDU, the NodeInt establishes the communication with the target device and transmits the received command. Upon receiving the R-APDU, the NodeInt forwards the R-APDU which consists only of Data bytes, to the NtU. Throughout this process, the NtU waits for the response in the main infinite loop of the function. If the reception was successful, to the received Data bytes of the R-APDU, Response status bytes (SW1 and SW2), which are by the default 0x90 and 0x00 and indicating the success, are added. At this point, the R-APDU is complete and it is transmitted to the Android NFC-enabled smart phone.

Access to the General Information Record

In order to enable the access to the NDEF tag application of the NtU and in order to write the *General Information Record* structured as NDEF message, following functions are defined and implemented: SELECT - provides functionality to select the NDEF file, READ - provides reading of the NDEF file and UPDATE - provides procedures for writing or updating of a NDEF message which is located inside the NDEF file. Functions and corresponding C-APDUs are defined and implemented according to the type 4 tag technical specification [For11].

4.2.3 Communication between the NodeInt and the NtU

Application Note provides all the necessary functionalities for the realization of communication on the NtU side. This section explains the communication on the side of the NodeInt, which is realised with the help of two interrupt routines, as shown in the Table 4.1:

`__interrupt void Port_1_ISR(void)`	**I/O data line interrupt handler**
`__interrupt void Timer_A_ISR(void)`	**Timer interrupt handler**

Table 4.1: Node Interface: Interrupt Routines.

As NodeInt acts as a slave in this communication; at first it receives the command from the NtU and then continues with transmission to the target device.

The process of receiving data from the NtU is performed as follows: by the default, or more precisely, in idle operation mode, the I/O data pin is configured as input pin with interrupt activated to detect the falling edge (logical transition from 'high' to 'low'). The start bit of the transmitted character, marked as logical transition from 'high' to 'low' level triggers the I/O data pin interrupt handler whose purpose is only, as depicted in the Figure 4.6 to detect the start bit and to prepare the timer for reception. After the execution, the interrupt on I/O pin is automatically deactivated for a time which is required by the timer interrupt to receive all 12 bits of a character.
The process in which the timer is prepared to receive data includes:

- Setting the timer registers to use the SMCLK in count up mode (timer repeatedly counts from 0 to the value set in the compare register of the timer)
- Setting the value of the timer compare register to the half bit time (bit duration of 372 divided by 2)
- Activating of the timer interrupt
- Setting of the timer flag to 1 in order to enable the reception mode

Timer interrupt handler consists of two routines or operational modes:

1. **Reception mode** - Handles the receiving of data from the NtU
2. **Transmission mode** - Handles the transmission of data to the NtU

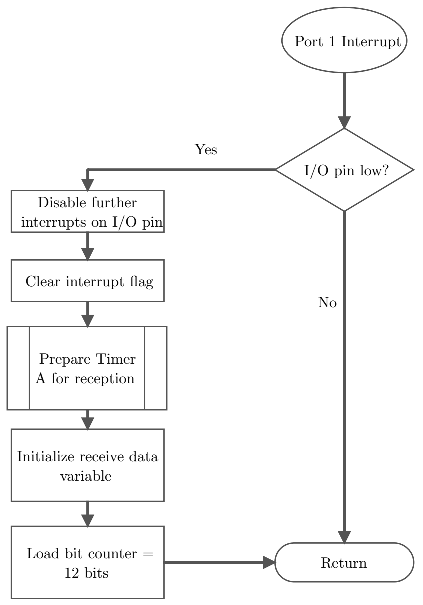

Port 1 Interrupt
I/O pin low?
Yes
No
Disable further interrupts on I/O pin
Clear interrupt flag
Prepare Timer A for reception
Initialize receive data variable
Load bit counter = 12 bits
Return

Figure 4.6: NodeInt: I/O Data Line Interrupt Handler.

Operational mode can be chosen, as shown in the Table 4.2, by setting the timer flag to the value of the desired mode:

`timer_flag == 0`	**Idle (default mode)**
`timer_flag == 1`	**Reception mode active**
`timer_flag == 2`	**Transmission mode active**

Table 4.2: Node Interface: Timer flags.

By finishing the detection of the start bit and leaving the Port 1 interrupt routine, the reception mode of the timer is started.

Interrupt reception handler checks the I/O pin signal. If the signal is 'low', the received bit is 0, and if the signal is 'high', the received bit is 1. For each received character, the interrupt cycle is triggered 12 times, to receive all 12 bits. At the beginning of each cycle, in order for timer to keep the track of time which takes to receive one bit, the bit duration of 372 is loaded into the compare register. At the end of each cycle, the bit counter is decremented by one. The data is sent in little endian format (least significant bit first). When bit counter equals zero, the character is received. Start, parity and stop bits are removed, and the 8 bit data value is stored in the buffer. After the successful reception of a transmitted character, timer flag is set to 0 and the I/O data pin interrupt is reactivated. In order to avoid data framing errors, once a character is received, it is necessary to clear the contents of timer registers. For the next received character, the timer registers are initialized anew in the Port 1 interrupt routine.

The complete timer interrupt reception mode implementation is depicted in the Figure 4.7, by means of flow chart graph.

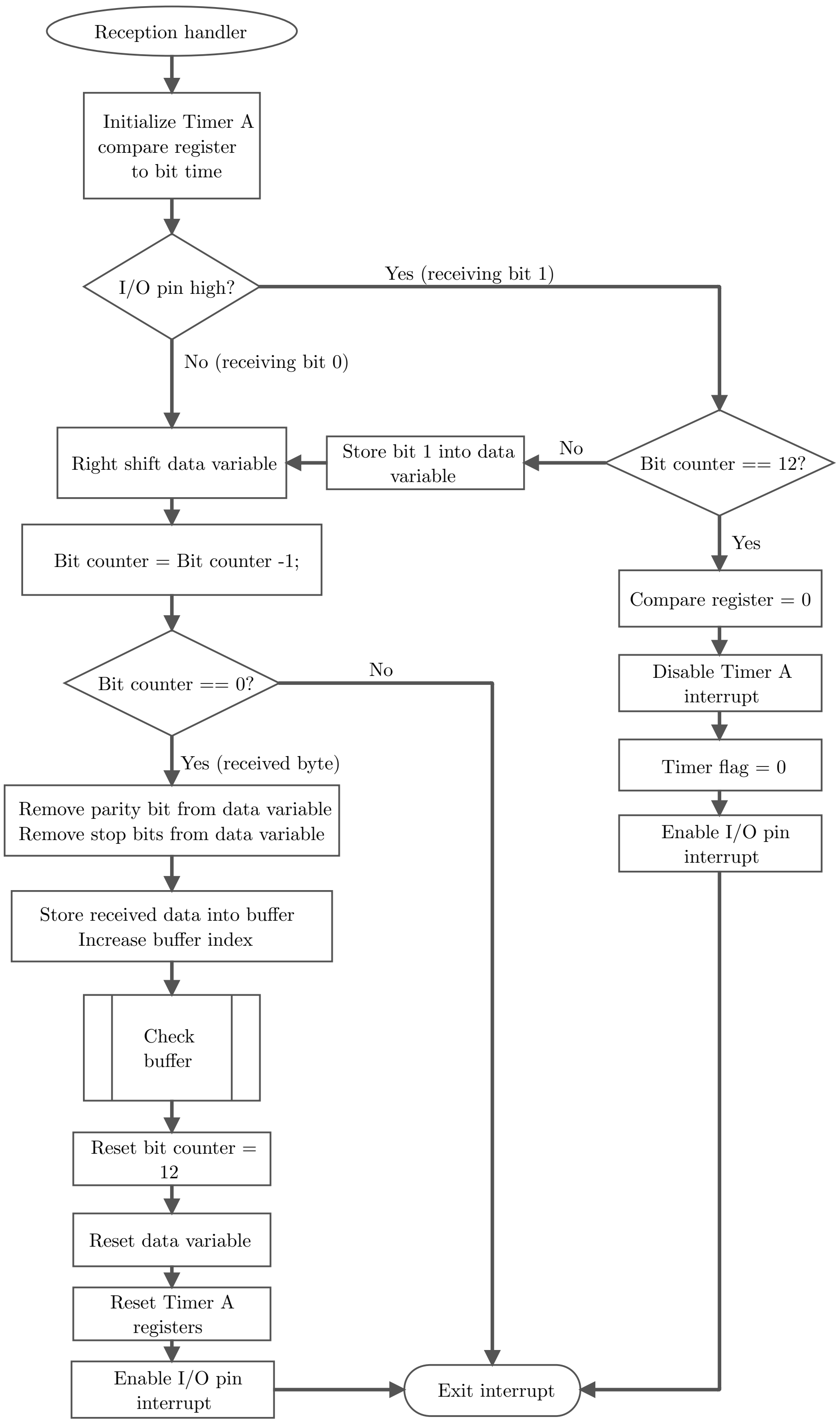


Figure 4.7: NodeInt: Timer Interrupt Reception Handler.

The transmission of data from the NodeInt to the NtU is accomplished by using the transmit function and the timer interrupt routine. The implemented function, whose implementation is depicted by the Figure 4.8, takes a character as an argument and it is called each time a character is transmitted. Before the interrupt routine is called, the following preparations are performed by the transmission function:

- The I/O pin is set to output mode and timer functionality of the pin is activated
- Timer clock register is set to use SMCLK in count up mode
- Timer compare register's initial value is set to bit duration time (372)
- For the transmitted character the parity bit is calculated and other bits, including the start bit and 2 stop bits are stored in the transmission variable
- Bit counter is initialized to 12
- Timer flag is set to 2 to enable the transmission mode
- Timer interrupt is activated

The function waits until all bits are transmitted and accordingly, until the timer interrupt is disabled. This is followed by a deactivation of the I/O pin's timer functionality, and the I/O pin is set to input mode.

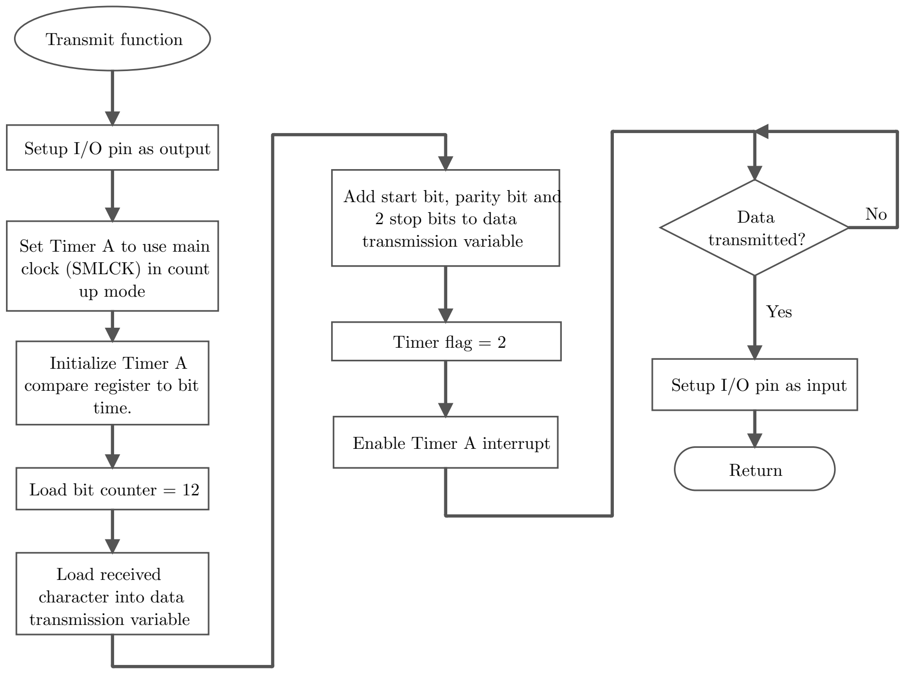

Transmit function
Setup I/O pin as output
Set Timer A to use main clock (SMLCK) in count up mode
Initialize Timer A compare register to bit time.
Load bit counter = 12
Load received character into data transmission variable
Add start bit, parity bit and 2 stop bits to data transmission variable
Timer flag = 2
Enable Timer A interrupt
Data transmitted?
No
Yes
Setup I/O pin as input
Return

Figure 4.8: NodeInt: Transmit Function.
]

For each bit of the transmitted character, the timer transmission handler is called. The flow chart in the Figure 4.9 shows all steps of the timer transmission handler. In the timer initialization step, timer compare register's value is set to bit duration time (372). Before the transmission occurs, the routine checks the bit counter and its current value. The initial value of the bit counter is 12 and it is decrement after each transmission. For each 12 bits of the transmitted character, the routine checks the least significant bit. If this bit is 1, I/O pin's output signal is set to 'high', and if the bit is 0, I/O pin's output signal is set to 'low'. In order to move to the next bit within the character, it is necessary to shift the character (its bit representation).

After successful transmission of all 12 bits, bit counter's value is zero and the timer interrupt is disabled. Furthermore, timer registers values are set to zero in order to avoid data framing errors when transmitting the next character, or in case when the last character of the command is transmitted, to prepare the timer for reception mode.

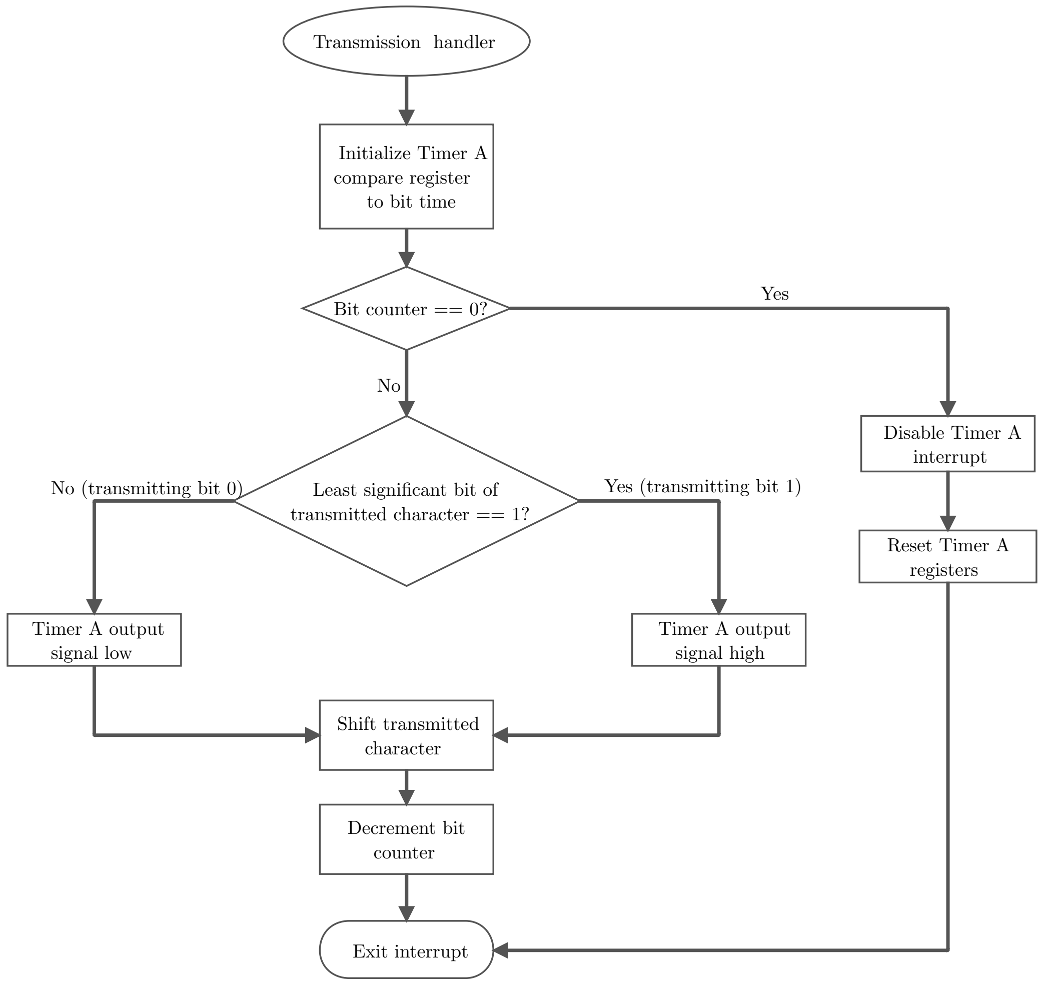

Transmission handler
Initialize Timer A compare register to bit time
Bit counter == 0?
Yes
No
Disable Timer A interrupt
Reset Timer A registers
Least significant bit of transmitted character == 1?
No (transmitting bit 0)
Yes (transmitting bit 1)
Timer A output signal low
Timer A output signal high
Shift transmitted character
Decrement bit counter
Exit interrupt

Figure 4.9: NodeInt: Timer Interrupt Transmission Handler.

4.2.4 Communication between the NodeInt and the Target Device

In the Hardware Setup process the USCI module is configured to operate in UART mode. The implementation consists of an interrupt routine which regulates the reception of data and a function that performs the data transmission. In the interrupt routine, the received character is copied from the USCI reception buffer UCA0RXBUF to the implemented ring reception buffer. It is important to store the received character in the internal ring buffer immediately after the reception, in order to avoid the overwriting by the next received character.

Function which performs the transmission takes a character as an argument and writes the character to the transmission buffer UCA0TXBUF of the USCI module. The rest of the process of sending and receiving is automatically regulated by the USCI module.

4.2.5 Data Buffering Implementation

Two ring buffers are provided by the NodeInt's implementation for storing of received commands and responses:

1. Buffer for storing received C-APDUs from the NtU
2. Buffer for storing received R-APDUs from the target device

Transmitted characters are stored in binary format. Both buffers have fixed length of 180 bytes and each buffer has a pair of global index variables:

1. Writing index keeps track of amount of data (bytes) which is stored in the buffer
2. Reading index keeps track of amount of data (bytes) which is transmitted from the buffer

During receiving, writing index is incremented for every received byte of the C-APDU or the R-APDU. During transmitting, reading index is compared to the writing index and incremented every time a byte is transmitted. The transmitting is performed until the two variables have the same index position value of the data being transmitted from the buffer.

To ensure that there will be no buffer overflow, every time a byte is stored in the buffer, the writing index is compared with the length of the buffer. If the end of the buffer is reached, the writing index is reset to zero and application begins to overwrite the oldest value in the buffer. The same procedure is performed during reading and transmitting from buffer. If the reading index reaches the last element of the buffer, reading starts from the first element of the buffer.

4.3 Implementation of the Android Application

In this section, a realization of the conceptual design of the Android application and additional libraries is presented. The structure of the application is described at the package level. This approach gives insight into the components, used classes and resources.

4.3.1 Implementation of the Basic Communication Library

As described in the design chapter, the main task of the *Basic Communication Library* is to provide functionalities for establishing connection and communication between the NFC module of the Android smart phone and the NFC-I. The library consists of six classes and the following Android NFC API classes are used for the realization of this task:

- `android.nfc.NfcAdapter`
- `android.nfc.Tag`
- `android.nfc.NdefMessage`
- `android.nfc.NdefRecord`
- `android.nfc.tech.IsoDep`
- `android.nfc.tech.Ndef`

Figure 4.10 shows the relationship between the classes of the library. *NFCI* is the main class of the library, whose instance is used by user applications and other libraries for the establishment of communication with the NFC-I. It acts like the interface and provides all necessary communication operations at the top, library level. The whole process of interaction with the NFC-I is controlled by the *NFCI* class using the *TagCommunicator* and the *Node* classes.

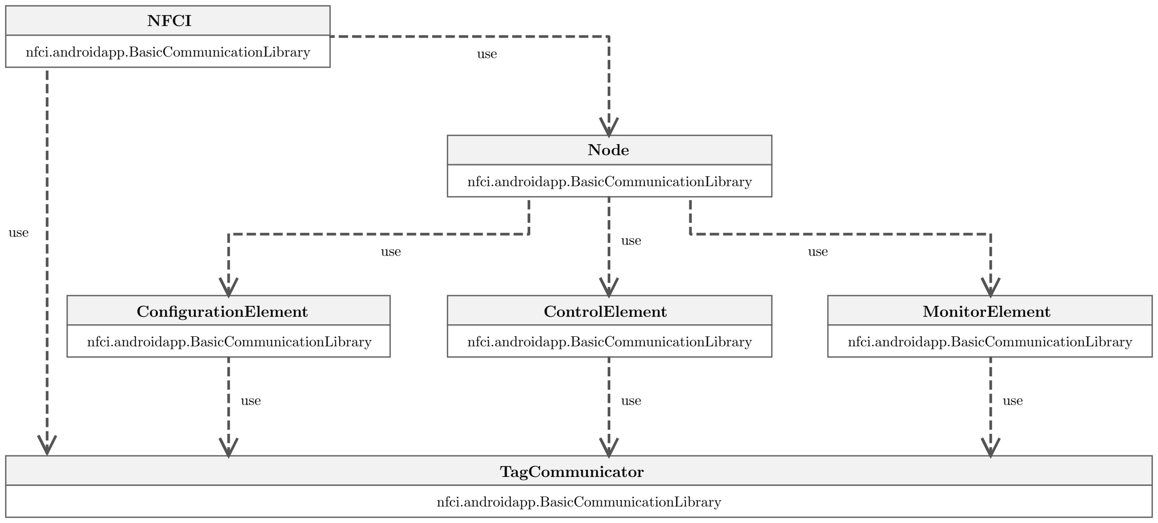

NFCI
nfci.androidapp.BasicCommunicationLibrary
use
Node
nfci.androidapp.BasicCommunicationLibrary
use
use
use
use
ConfigurationElement
nfci.androidapp.BasicCommunicationLibrary
ControlElement
nfci.androidapp.BasicCommunicationLibrary
MonitorElement
nfci.androidapp.BasicCommunicationLibrary
use
use
use
TagCommunicator
nfci.androidapp.BasicCommunicationLibrary

Figure 4.10: Android Smart Phone: Basic Communication Library Class Diagram.

TagCommunicator is the low level class which utilizes Android NFC API and other necessary packages to access and control the hardware NFC module. This class provides methods for the following operations:

- Detection of the NFC-I
- Identification and activation of the NFC-I
- Establishing of the communication with the NFC-I, or more precisely, with the NtU component of the NFC-I
- Accessing of the NFC-I's NDEF tag application
- Retrieving and writing of the *General Information Record*
- Native communication (sending of C-APDUs and receiving of R-APDUs)

The process of detecting the NFC-I is implemented in the method that is shown in the Listing 4.1.

Listing 4.1: Basic Communication Libraray: Detection of the NFC-I.

```
public boolean TagAvailable(){

    IsoDep tag = IsoDep.get(this.connectedTag);
    try
    {
      tag.connect();
      boolean connected = tag.isConnected();
      tag.close();

      return connected;
    }
    catch (Exception e)
    {
      Log.v("TagCommunicator","TagAvailable failed (NFC-I unavailable)");
    }

  return false;
}
```

If the process of detecting the NFC-I is successfully completed, the next step consists of connecting to the NFC-I and reading of the *General Information Record*. Connecting to the NFC-I and the extraction of the *General Information Record* from the NDEF message is performed in the `Connect` method of the *TagCommunicator* which is depicted in the Listing 4.2.

Listing 4.2: Basic Communication Libraray: Connecting to the NFC-I.

```
public void Connect(Intent intent)
{
  try
  {
  // Get the detected NFC-I
  Tag tagFromIntent = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
  //Retrieve the ndef information stored on the tag
  Ndef tagType4Tag = Ndef.get(tagFromIntent);
  //Retrieve the NDEF message
  NdefMessage message = tagType4Tag.getCachedNdefMessage();
  //Retrieve the General Information Record
  tagInformation = message.getRecords();
  this.connectedTag = tagFromIntent;
  }
  catch (Exception e)
  {
    Log.e("TagCommunicator","Connecing failed (NFC-I unavailable or
        incompatible)");
  }
}
```

TagCommunicator provides two methods for working with the *General Information Record*. First method is used to read and parse all information from the *General Information Record* and the second method is used to modify the content of the *General Information Record*. The content and form of the *General Information Record* is described in the Section 3.4.5. During the initial reading of the *General Information Record* and based on the extracted parameters *Activation flag*, *Identifier* and *Type* of the NFC-I, the checking is performed whether the NFC-I is activated and if it has the right identifier. Only in case if all these parameters are valid and the NFC-I is activated, the process of parsing the nodes of the target device and their elements can be performed. In case when the NFC-I is not activated

Node class represents the nodes of the target device which are connected to the NFC-I and it is used by the main *NFCI* class to store the information about available nodes and their elements. All nodes and corresponding node elements are mapped into the list according to their names, identifiers and types. *NFCI* has access to the list of all available nodes. With a reference to lists of all existing types of node elements and their identifiers, the *Node* class provides methods for getting the list of all elements of a certain type, getting the value of a certain element and setting the value of a certain element of a node. Furthermore, *NFCI* provides the method for activating the NFC-I in case if it is not activated. Activation process is accomplished by providing the defined password. If the password is correct, the method which provides modifying of the *General Information Record* within the *TagCommunicator* is called, and the *Activation flag* value is set to 1.

Three different types of node elements are supported by the implementation: control, configuration and monitor element. Classes *ControlElement*, *ConfigurationElement* and *MonitorElement* represent these elements and provide methods specific for each type of an element. These classes have the instance of the *TagCommunicator* and direct access to the native communication methods which implement sending of C-APDUs and receiving of R-APDUs.

Native communication with the NFC-I

If the process of identifying all available nodes and their elements is successfully completed, the application has all necessary information for performing the native communication with an NFC-I enhanced target device.

When a request is sent from the user application, it contains information of a certain node (name of a node) and the target element of a node (name of an element). Depending on whether the request is related to getting or setting the value of a certain element, the request might also contain data bytes. Data bytes are certain values which are transmitted to a node element if the request is related to setting the value of an element.

The request is parsed by the *NFCI* class methods which have access to the list of all available nodes. If the appropriate node is found in the list, the next step is to find the element corresponding to the element name. The search is carried out with the help of *Node* class methods which have access to lists of all element types (configuration, monitor and control) of a node. When the appropriate element and its type are identified, the methods within the element classes are called taking the node identifier, element identifier and data bytes as an argument. Due to a fact that the element classes have direct access to the *TagCommunicator*, the whole request consisting of node identifier, element identifier and optionally data bytes is passed as an argument to the native methods which communicate directly with the node elements of an NFC-I enhanced target device.

Native communication is implemented in two methods of the low level *TagCommunicator* class. The first method, *WriteNative* is responsible for setting the value of an element, while the *ReadNative* is responsible for getting the value of an element. The method which is used to set the value of an element, as shown in the Listing 4.3, receives three arguments: node identifier, element identifier and data bytes.

Listing 4.3: Basic Communication Libraray: Write Native.

```
public void WriteNative(byte node, byte element, byte[] value) {
  try {
      IsoDep tag = IsoDep.get(this.connectedTag);
      tag.connect();

      byte[] capdu = new byte[value.length+6];
      capdu[0] = (byte)0xC0;
      capdu[1] = (byte)0xF2;
      capdu[2] = node;
      capdu[3] = element;
      capdu[4] = (byte)value.length;
      capdu[5] = (byte)0x01;
      System.arraycopy(value, 0, capdu, 5, value.length);

      // Send C-APDU
      tag.transceive(capdu);
      tag.close();
  }
  catch (Exception e)
  {
    Log.e("TagCommunicator","WriteNative failed");
  }
}
```

As Listing 4.3 shows, the C-APDU is constructed in such a way that the first two elements, Class and Instruction bytes are by default always the same (0xC0, and 0xF2), as it is also defined in the specification for NFC-I, see Section 4.2.2. Node identifier is added as Parameter byte 1 and element identifier is added as Parameter byte 2. The data bytes length and data bytes itself are added thereafter. Now, the C-APDU is fully constructed and transmitted to the NFC-I.

The second method, *ReadNative*, works in a similar way, i.e. C-APDU is constructed in the same way, only in this case, there is no data bytes. After sending the C-APDU, the method waits for the response in form of an R-APDU, which is returned to the user application. The method is depicted in the Listing 4.4:

Listing 4.4: Basic Communication Libraray: Read Native.

```java
public byte[] ReadNative(byte node, byte element)
{
  Log.v("TagCommunicator","ReadNative(" + node + element + ")");
  byte[] response = new byte[]{};

  try {
      IsoDep tag = IsoDep.get(this.connectedTag);
      tag.connect();

      // Send C-APDU and wait for response (R-APDU)
      response = tag.transceive(new byte[] { (byte)0xC0, (byte)0xF2, node,
          element, 0x01, 0x00});

      tag.close();
  }
  catch (Exception e)
  {
    Log.e("TagCommunicator","ReadNative failed");
  }

  return response;
}
```

4.3.2 Implementation of the Extended Communication Library

As Figure 4.11 depicts, the developer defines the class, as part of the *Extended Communication Library*, which provides additional functionalities required by a specific user application or a group of user applications.

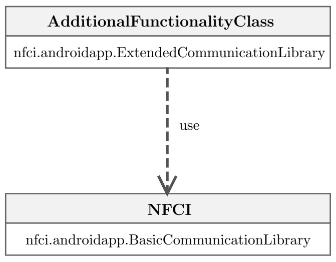

AdditionalFunctionalityClass
nfci.androidapp.ExtendedCommunicationLibrary
use
NFCI
nfci.androidapp.BasicCommunicationLibrary

Figure 4.11: Android NFC-enabled Smart Phone: Extended Communication Library Class Diagram.

The defined class uses the instance of the *Basic Communication Library* to establish the communication with the NFC-I. The implemented methods of this class, which provide the desired additional functionalities, are directly used by the user application. Additional functionality provided for the use cases implemented in this project are related to getting the status of the target device. When user sends the command for getting the status of the target device, the request is executed within the method of the *Extended Communication Library*. By using the communication methods of the *Basic Communication Library*, the request is sent to the target device. The status information received from the target device is processed by the method of the *Extended Communication Library*, and transferred to the user application.

4.3.3 Implementation of the Main and the User Application

When user starts the Android application, the initialization process is started in which the application starts the main user interface that just says that the application is ready and waiting for the Android NFC-enabled smart phone to detect the NFC-I enhanced target device. Besides, the application gets access to the NFC module and maps all NFC related detections to itself. This is achieved by defining an *Intent*. To put it simply, the application presents the intention to perform certain actions with the detected tags or, in this case, the NFC-I enhanced target device. After that, the application waits for the NFC module to detect the target device and to take the control over it. When the target device is detected, the following process is started: the application uses the instance of the *Basic Communication Library* to establish the connection to the target device and to read out the *General Information Record*. According to the information about the type of the NFC-I (target device), the application switches the user interface to one that matches the target device. In case when the NFC module detects the other target device or if the present target device disappears and reappears in the magnetic field of the NFC module, the process is repeated anew. Before the application switches the user interfaces, the reading of the *General Information Record* is completed and the application has access to information about nodes and their elements.

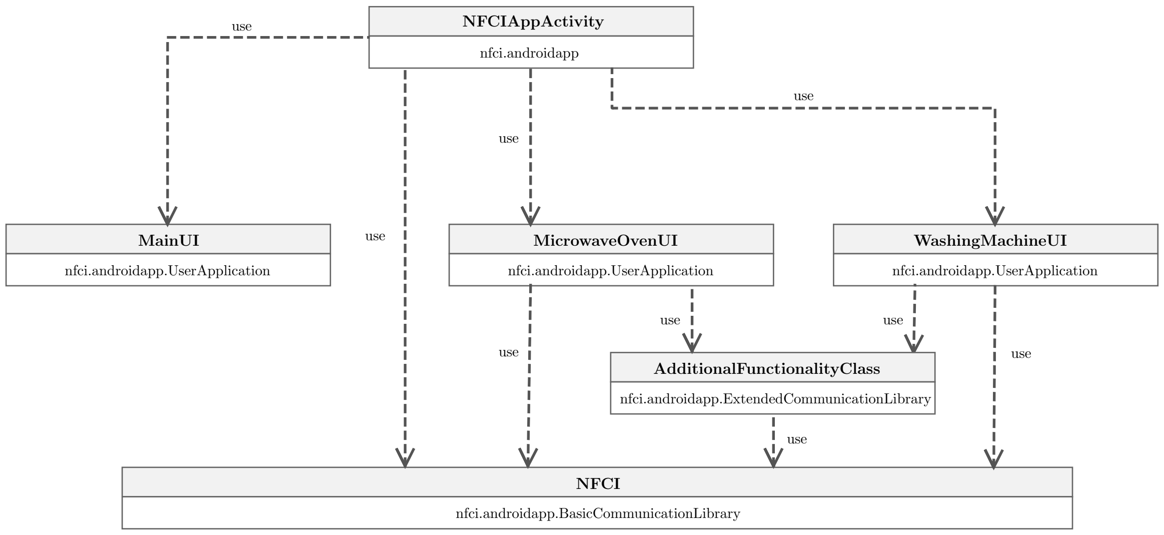

NFCIAppActivity
nfci.androidapp
use
use
use
use
MainUI
nfci.androidapp.UserApplication
MicrowaveOvenUI
nfci.androidapp.UserApplication
WashingMachineUI
nfci.androidapp.UserApplication
use
use
use
use
AdditionalFunctionalityClass
nfci.androidapp.ExtendedCommunicationLibrary
use
NFCI
nfci.androidapp.BasicCommunicationLibrary

Figure 4.12: Android NFC-enabled Smart Phone: Application Class Diagram.

As already described, the main class of the application is concerned about the detection of the target device and determining which user interface shall be started, while the classes which implement the user interface perform the target device specific operations. The simplified class diagram in the Figure 4.12 shows the relationship between the user application, the main application class and the *Basic* and *Extended Communication Libraries*.

Structure of the Android Manifest File

Android manifest file for the implemented application defines *NFCIAppActivity* as the main class which implements the *Activity* and represents the entry point of the application. *Intent Filter* is defined to let the system know the application's intentions, i.e. intention to discover an NFC-I enhanced target device. Even when the application is not started and the NFC module detects an NFC-I enhanced target device, the system offers/suggests the user to start the application. Furthermore, the permissions for the application are defined. Specifically in this case, the application must have permission to use the NFC module.
The content of the manifest file is shown in the Listing 4.5:

Listing 4.5: Android Manifest File Source Code.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="infineon.nfci.app"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="14" />

    <uses-permission android:name="android.permission.NFC" />
    <uses-feature android:name="android.hardware.nfc" android:required="
        true" />

    <application
        android:icon="@drawable/identifier"
        android:label="@string/app_name" >
        <activity
            android:label="@string/app_name"
            android:name=".NFCIAppActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
                    />
            </intent-filter>
            <intent-filter>
                <action android:name="android.nfc.action.TAG_DISCOVERED"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

User Application

For each target device which is operated with the help of Android application, a separate class, as part of the application, is defined with the name of the target device. User interfaces are designed in such way that users can easily, without facing difficulties, use them to operate the target device. In other words, if the target device is the washing machine, and the application is used to control the different washing programs for different clothes and different temperatures, then the application should provide intuitive user interface with images and descriptions suggesting the desired washing program. This is accomplished by utilizing Android user interface packages: `android.view` (building of blocks for user interface elements) and `android.widget` (provides user interface elements such as: button, edit text, check box, toast, etc.). Furthermore, several other resources such as images are used to create the user interfaces.

For each user interface element which performs certain operations, event listeners are defined. By interacting with the user interface (e.g., user presses button), the native communication between the Android application and the NFC-I enhanced target device is executed. User application classes use the instance of the *NFCI* class for performing communication with the NFC-I enhanced target device.

4.4 Implementation of the Target Device Application

This section deals with the implementation of target device application and its components which are presented in the conceptual design stage. As a simplified class diagram in Figure 4.13 shows, the implementation consists of the main application class, user interface classes which visually and functionally represent target devices and the *NFCI Communication Library.*

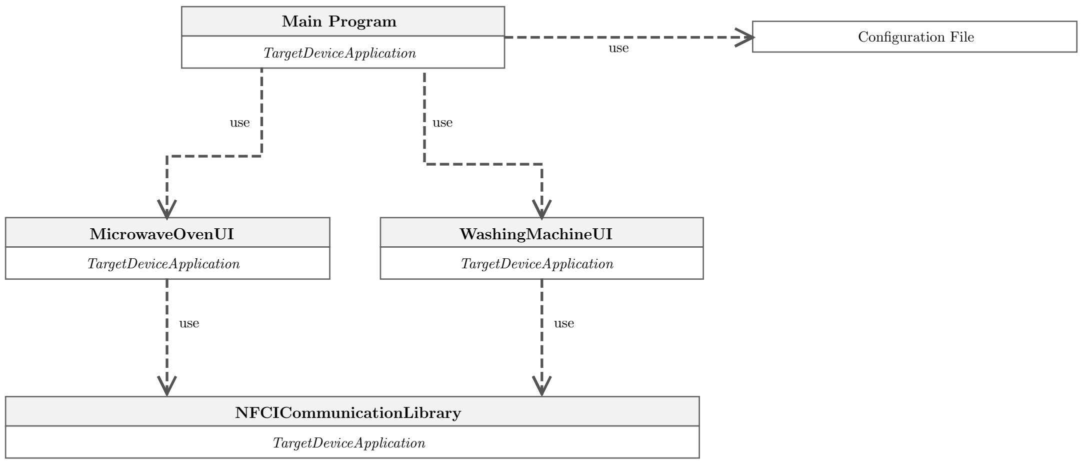


Figure 4.13: Target Device Application: Simplified Class Diagram.

4.4.1 Implementation of the NFCI Communication Library

As already stated in the design phase, the main task of the *NFCI Communication Library* is to provide all necessary methods for establishing communication and exchange of data between the NFC-I and the computer which runs the target device application. The communication is accomplished through serial communication port and the library provides several primitives and methods for handling basic tasks on serial port:

- Creating and initialization of serial port
- Methods for opening and closing serial port
- Method for checking if the serial port is open

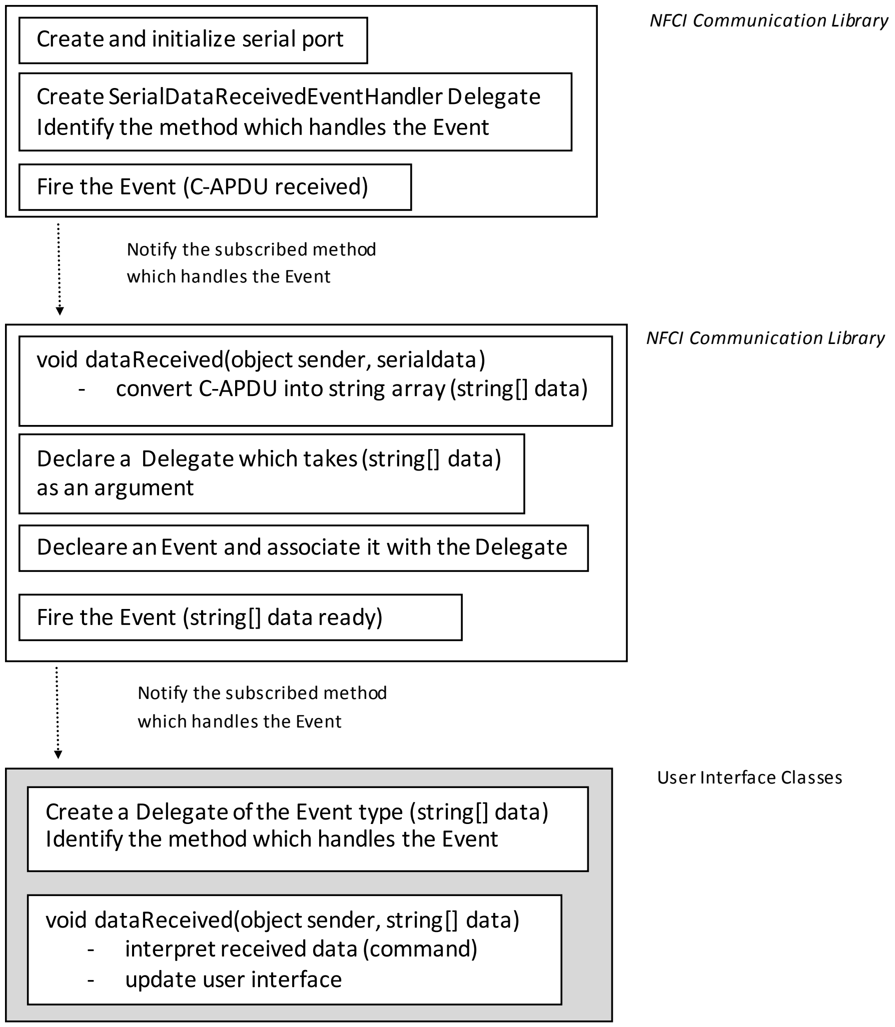

NFCI Communication Library
Create and initialize serial port
Create SerialDataReceivedEventHandler Delegate
Identify the method which handles the Event
Fire the Event (C-APDU received)
Notify the subscribed method which handles the Event
NFCI Communication Library
void dataReceived(object sender, serialdata)
- convert C-APDU into string array (string[] data)
Declare a Delegate which takes (string[] data) as an argument
Decleare an Event and associate it with the Delegate
Fire the Event (string[] data ready)
Notify the subscribed method which handles the Event
User Interface Classes
Create a Delegate of the Event type (string[] data)
Identify the method which handles the Event
void dataReceived(object sender, string[] data)
- interpret received data (command)
- update user interface

Figure 4.14: NFCI Communication Library: Serial Data Processing.

In addition to these basic functionalities, the *NFCI Communication Library* implements methods which are responsible for sending and receiving data over the serial port. In order

to enable the reception of data (C-APDUs) coming from the NFC-I, it is necessary to create a *SerialDataReceivedEventHandler* and identify it with a new method that will handle the event. Defined method reads the bytes from the serial port (receives C-APDU's bytes) and converts them into series of strings. Such data is ready for further processing by the user interface classes that implement specific target devices. In order to forward the converted data to the user interface classes, a new delegate within the *NFCI Communication Library* is defined taking the converted data as an argument. Also, an event is declared and associated with the defined delegate. In such way, user interface classes implement event handlers, which respond to these events and consume received event data in form of series of strings. This process is shown in more conform way by the diagram in the Figure 4.14.
For the purpose of sending the response (R-APDU) from the target device to the NFC-I, the *NFCI Communication Library* provides a method which takes byte array as an argument and writes these bytes to the serial port.
NFCI Communication Library is designed and implemented in a way that it can be easily involved in development of user interface classes.

4.4.2 Implementation of the Main Application and User Interfaces

The target device application supports multiple devices/hardware which are visualized and their functionalities are implemented with user interface classes. Within the main application class, the user can choose which target device shall be executed. As Figure 4.13 depicts, this is accomplished by using configuration file which is parsed each time the application is started. The structure of the configuration file is quite simple and contains information about all available target devices and the one which is active in the moment of execution.

In order to create the photo-realistic user interfaces for target devices which are visualized, it is necessary to use a number of image resources that visually resemble the appearance of real target devices and their parts. By using images and standard user interface components, such as forms, panels, buttons, text boxes, timers, etc., static and dynamic parts of the target device user interface are designed. Static parts refer to visual design components which are not influenced and which remain unchanged during the interaction between the target device and the Android application controlled by the user. Dynamic parts are actually node elements which provide target device specific functionalities and which are altered based on received C-APDUs. The only action which must performed by the user on the target device application is creating and closing of the serial communication port between the PC and the NFC-I. This is accomplished by integrating a power on/power off button, which performs following operations:

- Establishment of communication with the NFC-I (creating and opening of serial communication port)
- Creation of the delegate and associating it with the method which is responsible for parsing of received C-APDUs
- Closing of serial communication port

Furthermore, several utility methods are used by the user interface classes for working with image resources.

As class diagram in Figure 4.13 shows, user interface classes use the instance of the *NFCI Communication Library* to establish the communication with the NFC-I and to perform data exchange. Received C-APDUs, depending on type of the node element and depending on whether the C-APDU is related to setting or getting the value of the node element, might modify the user interface or just read node element's actual value. All received requests (converted C-APDUs) are parsed within the method which is registered as event handler. Accordingly, for each received C-APDU, the response (R-APDU) is automatically generated, containing information about the node element. The generated R-APDU is transmitted over the NFC-I to the Android application.

Chapter 5

Results

This chapter presents the results obtained during the realization of the project. The main result of this project is a prototype version of the innovative NFC-I that provides target electronic devices with NFC communication. In order to adequately demonstrate the functioning and operation of the NFC-I, two use cases are implemented. In both cases, the functioning and operation of the NFC-I is demonstrated using household appliances as target devices. For testing purposes, an Android application is implemented which provides support for NFC-I and proposed target devices.

5.1 NFC-I Prototype Version

All construction costs for the prototype version, including both components of the NFC-I, do not exceed the sum of 4 Euro. Launchpad featuring MSP430G2553 microcontroller is relatively cheap for the features it provides. The M7794 security microcontroller provided by the Infineon is also one of the cheapest in the series.

Hardware setup, which is done according to the specification about connecting between the components of the system (see Section 3.4.3), is depicted by Figure 5.1 and Figure 5.2. Although the figures show the NFC-I as independent component, its final version is integrated inside the target device.
The user starts the detection of the NFC-I by placing Android NFC-enabled smart phone upon the NFC antenna of the NFC-I or, more precisely, upon the place where the NFC-I is installed into the target device. The NFC-I is physically connected with the target device over one of the supported contact-based interfaces (UART, USB). In case of prototype setup, the NFC-I is physically, via USB interface, connected to the PC which runs the target device application. In general, the NFC-I is powered by the target device, and in case of the shown prototype, the NFC-I is powered through USB.

Figure 5.3 illustrates the Android NFC-enabled smart phone interacting to the NFC-I enhanced washing machine. The goal of the project was not to test the interface with real device or household appliances, yet to show that this idea is possible and that the NFC-I provides all necessary functionalities to work with real target devices. Thus, Figure 5.3 only shows how it would look like to operate the NFC-I enhanced real target devices using the Android NFC-enabled smart phone.

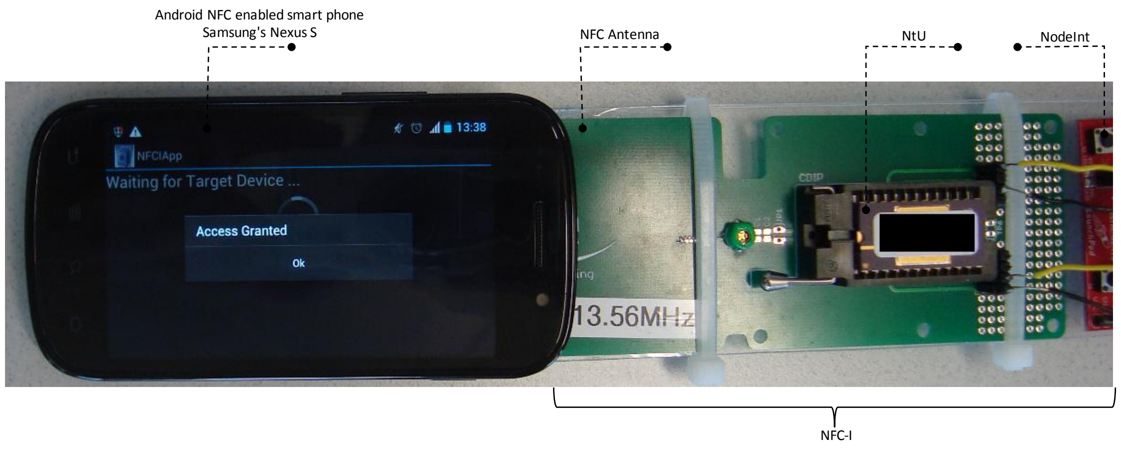


Figure 5.1: NFC-I: Hardware Setup (a).

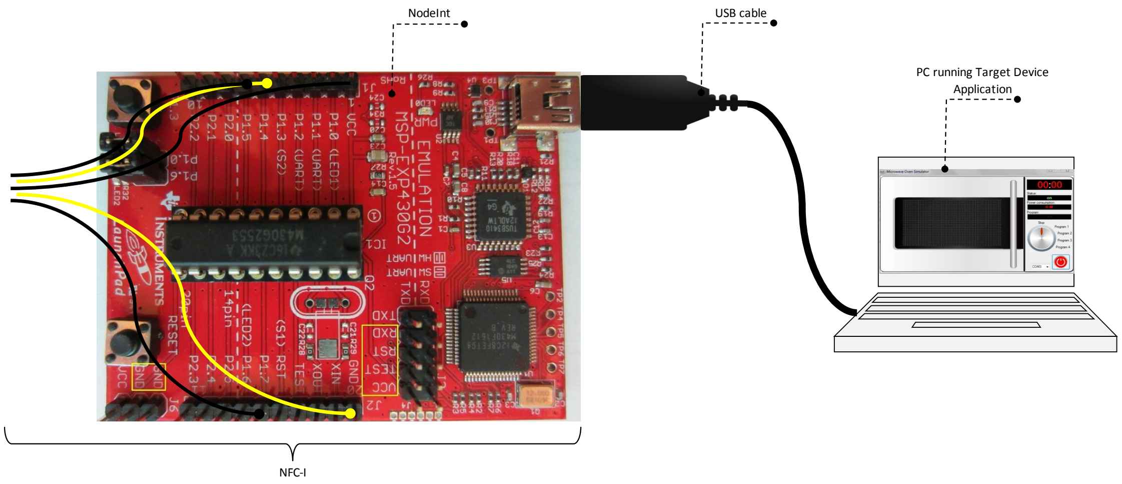


Figure 5.2: NFC-I: Hardware Setup (b).

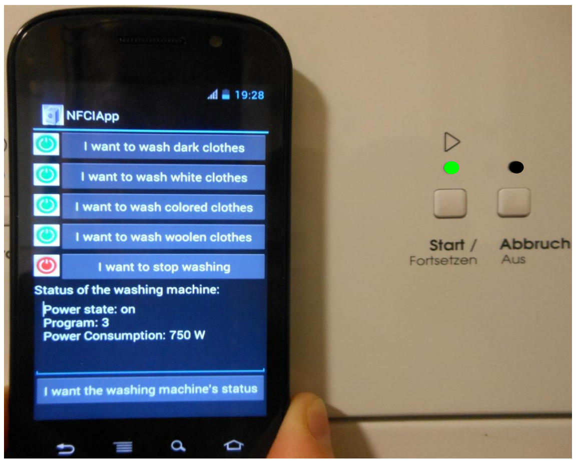

19:28
NFCIApp
I want to wash dark clothes
I want to wash white clothes
I want to wash colored clothes
I want to wash woolen clothes
I want to stop washing
Status of the washing machine:
Power state: on
Program: 3
Power Consumption: 750 W
I want the washing machine's status
Start /
Fortsetzen
Abbruch
Aus

Figure 5.3: Android Smart Phone and NFC-I enhanced Washing Machine.

5.2 Implemented Use Cases

In order to demonstrate the capabilities of the NFC-I enhanced target devices and the Android NFC-enabled smart phone which is used to interact with these devices, two use cases were planned and implemented. These two use cases implement and demonstrate a washing machine and a microwave oven as NFC-I enhanced target devices. Both household appliances are visually and functionally represented by PC application and well-designed user interfaces. The NFC-I is connected to the PC via USB cable. Accordingly, Android application was developed to provide graphical user interfaces and functionalities for interacting with the proposed target devices.

5.2.1 Target Devices

Washing Machine as NFC-I enabled Target Device

In the first use case a washing machine is used as target device. It supports four different combinations of clothing to wash, which can be selected on the Android NFC-enabled smart phone's application. The selection leads to a configuration of the washing program on the target device. Additionally, the washing process can be stopped. Another selection on the Android NFC-enabled smart phone can be used to get status information. This status information consists of the current operational state (on - running one of the washing programs or on - idle), current washing program (if any is running) and the power state (power consumption of the washing machine at the current running washing program). Figure 5.4 depicts the user interface of the washing machine. The red rectangle surrounds the section of the user interface which represents the washing machine's node (control logic unit).
Marker ① points the power on button. Underneath the power on button, the combo box shows all available serial communication (COM) ports on the PC. By selecting the COM port on which the NFC-I is connected to a PC and by pressing the power on button, the connection between the washing machine and the NFC-I is established. As previously mentioned in the implementation chapter, this action is controlled by the user. Upon creating the connection, the power on button is automatically replaced by a power off button. Now, the button is used to terminate the serial communication between the PC and the NFC-I.

Markers ②, ③, ④, ⑤ and ⑥ point to the node elements of the washing machine:

- ② Control element (washing program selector)
- ③ Monitor element (main timer display indicating the remaining time of the current washing program)
- ④ Monitor element (status display showing the current operational state)
- ⑤ Monitor element (shows the power consumption of the current washing program)
- ⑥ Monitor element (status display indicating the current running washing program)

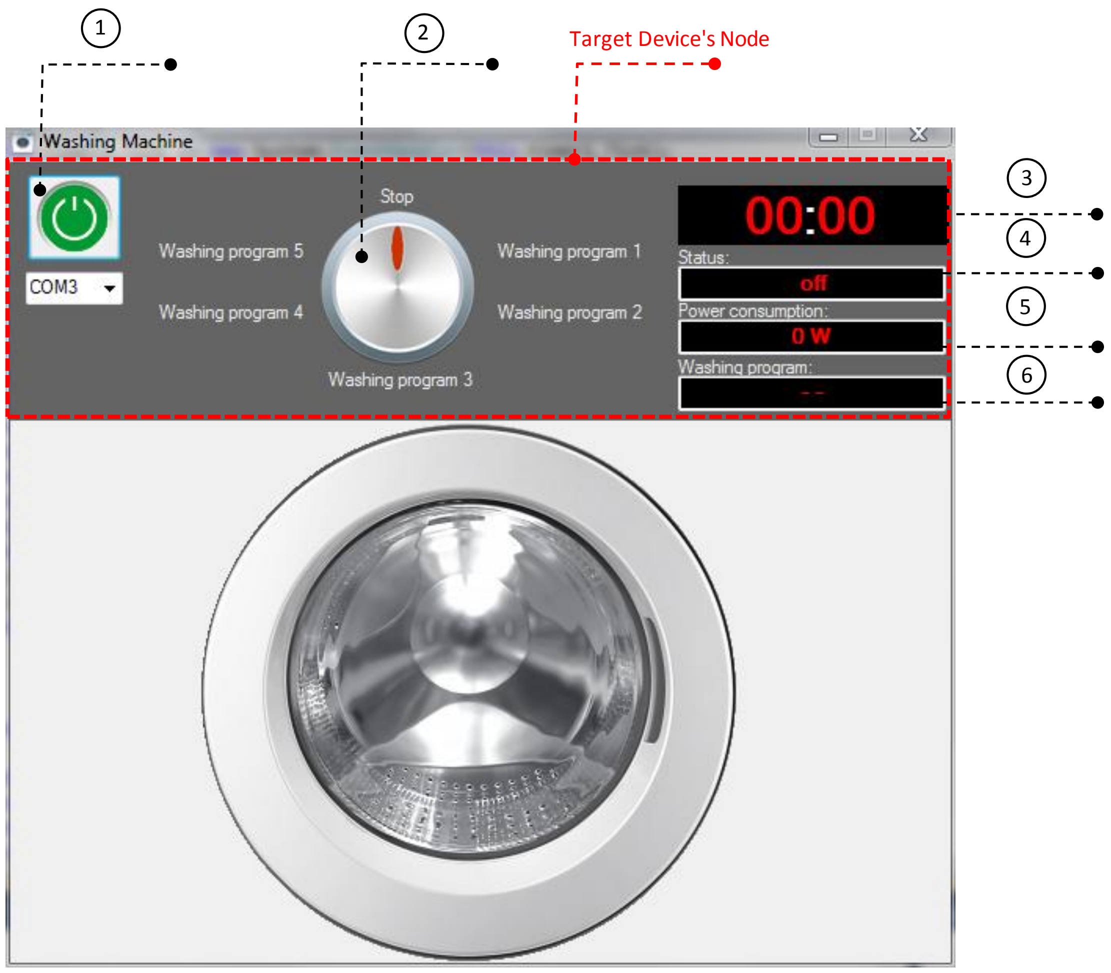


Figure 5.4: Use Case: Washing Machine (User Interface of the Washing Machine).

Microwave Oven as NFC-I enabled Target Device

In the second use case, a microwave oven is used as target device. Similarly to the washing machine, the microwave oven provides four different cooking programs which can be selected by the Android NFC-enabled smart phone's application. Also, the stopping of the cooking program can be selected. From within the Android NFC-enabled smart phone the user can invoke the status information of the microwave oven. The status information, as in the washing machine, consists of current operational state (on - running one of the cooking programs or on - idle), current running cooking program (if any is running) and the power state (power consumption of the microwave oven at the current running cooking program).
Figure 5.5 shows the user interface of the microwave oven. Microwave oven's node (control logic unit) is surrounded with the red rectangle. Contrary to the user interface of the washing machine, the communication between the PC and the NFC-I is already created, and the power on button is replaced with the power off button (see marker ①). Additionally, the status display indicates that the microwave oven is turned on (see marker ⑤)

and is automatically in idle state.

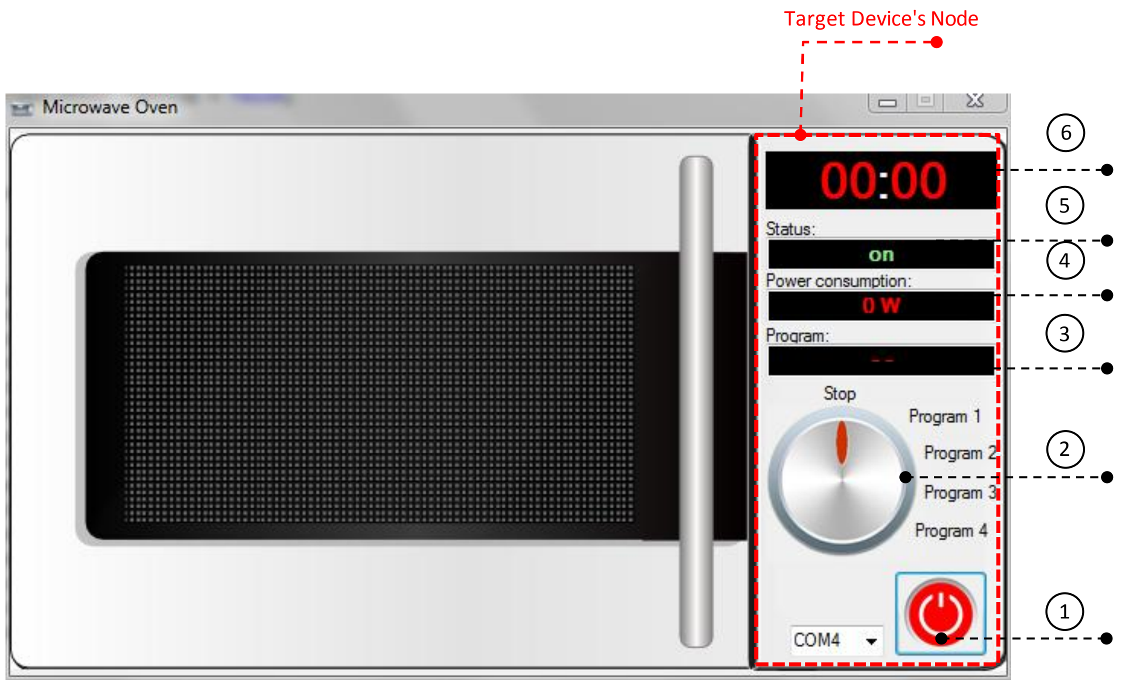


Figure 5.5: Use Case: Microwave Oven (User Interface of the Microwave Oven).

Markers ②, ③, ④, ⑤ and ⑥ point to the node elements of the microwave oven:

- ② Control element (cooking program selector)
- ③ Monitor element (status display indicating the current running cooking program)
- ④ Monitor element (shows the power consumption of the device running current cooking program)
- ⑤ Monitor element (status display showing the current operational state)
- ⑥ Monitor element (main timer display indicating the remaining time of the current cooking program)

The described node elements of both target devices show the response or change their appearance in the user interface according to the performed actions (sent commands) from the Android NFC-enabled smart phone. Power consumption values shown in the user interface do not indicate the actual power consumption of NFC-I enabled target devices. These values serve only for demonstration purposes and were chosen on the basis of average consumption of real household appliances in Watts.

5.2.2 Android Application

For the purposes of interacting with the two previously described target devices, an Android application was developed. The application utilizes *Basic* and *Extended Communication Libraries* to establish the communication with the target devices over the NFC-I and to provide functionalities for setting and getting certain values of target device's node elements. For each target device, a corresponding user interface is created. When user starts the application, and the NFC-I enhanced target device is not within the NFC range of the Android NFC-enabled smart phone, the main user interface, as depicted in the Figure 5.6, is displayed. The second way to start the application is to move the Android NFC-enabled smart phone upon the NFC-I enhanced target device. Android system shows then all available applications which have access to the NFC module and, among them, the implemented use case application.

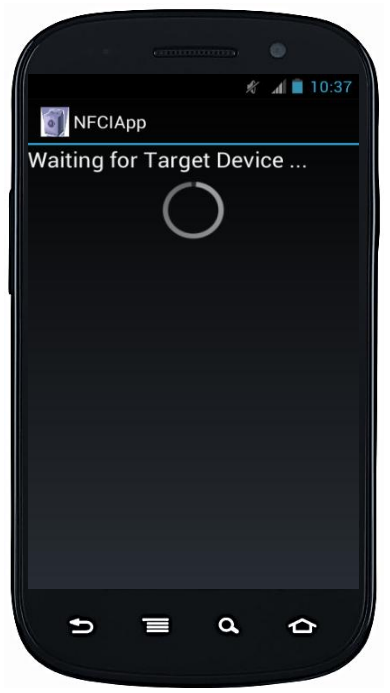
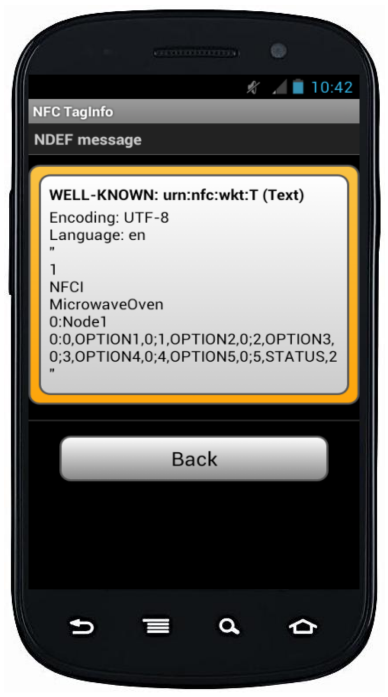


Figure 5.6: Android Application: Main UI - Waiting for Target Device.

Also, when the NFC-I enhanced target device is not activated or disappears from the magnetic field of the NFC module, the main user interface is displayed. Application provides functionalities for activating of the NFC-I enhanced target device by accessing the options menu and pressing 'Activate Target Device'. The activation is successfully made by providing the correct password. Also, options menu provides deactivating of the target device. In both cases, the *Activation flag* of the *General Information Record* is altered.
Only in case when: the target device is detected, the *Activation flag* is 1 and the NFC-I has the right identifier, the application proceeds with reading of target device's nodes and node elements. Figure 5.7 depicts the *General Information Record* for a NFC-I enhanced microwave oven. This process is completed in background and the user does not see the

text representation of the *General Information Record*.



```
WELL-KNOWN: urn:nfc:wkt:T (Text)
Encoding: UTF-8
Language: en
"
1
NFCI
MicrowaveOven
0:Node1
0:0,OPTION1,0;1,OPTION2,0;2,OPTION3,
0;3,OPTION4,0;4,OPTION5,0;5,STATUS,2
"
```

Figure 5.7: Android Application: Reading of the General Information Record.

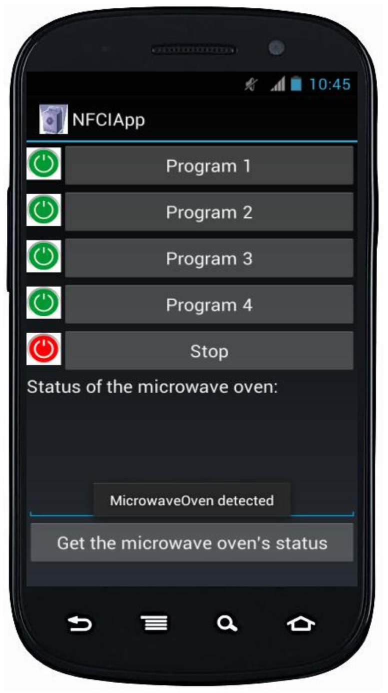
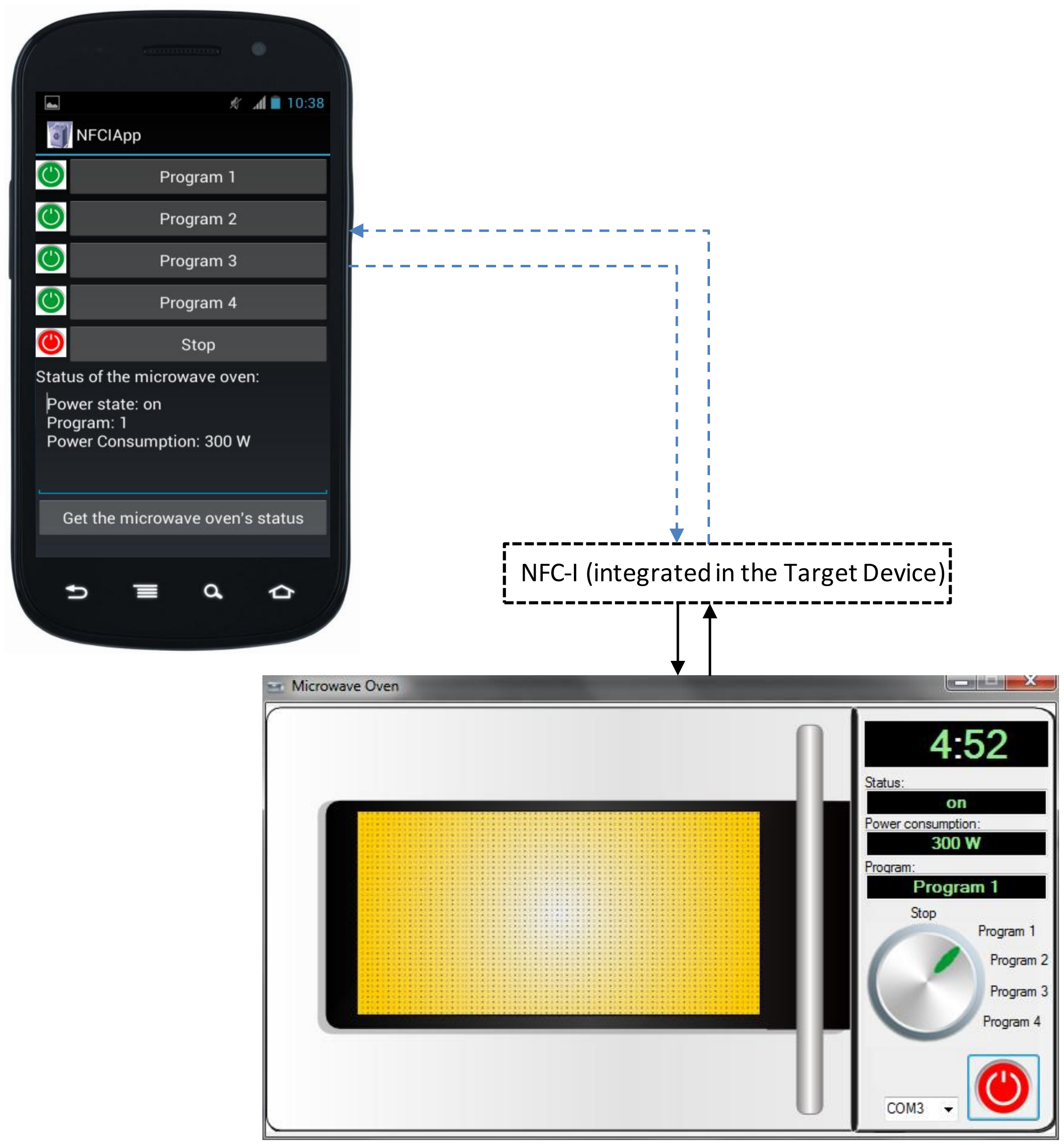


Figure 5.8: Android Application: Microwave Oven User Interface.

Upon finishing the extraction of nodes and node elements, and according to the type of the target device indicated in *General Information Record*, the user interface is switched to one that matches the target device. In this case, the user interface of the microwave oven is displayed as shown in the Figure 5.8. Also, a small pop-up appears telling that the microwave oven is detected.
Figure 5.9 outlines the interaction between the Android NFC-enabled smart phone and the microwave oven as target device.



Status of the microwave oven:
Power state: on
Program: 1
Power Consumption: 300 W

4:52
Status: on
Power consumption: 300 W
Program: Program 1

Figure 5.9: Interaction between the Android NFC-enabled Smart Phone and the Microwave Oven as Target Device: User starts Program 1 and invokes the Status of the Microwave Oven.

By pressing the 'Program 1' button in the Android application, user sends the command to start the 'Program 1' on the microwave oven. Respectively, the appearance of the microwave oven's user interface is changed from idle into running state:

- Timer display indicates the remaining time necessary for 'Program 1' to finish
- Power consumption display shows the current power consumption of the microwave oven
- Cooking program selector knob points to started program and program display shows 'Program 1'

This command or, more precisely, the action carried out by the user performs only setting the value of the microwave oven's node control element (cooking program selector). Therefore, no response is shown in the Android application. Yet when user presses the button 'Get the microwave oven's status', another command is sent to the microwave oven. This command is related to invoking the status of the microwave oven and results in showing status information in the user interface of the Android application. When the time for execution of 'Program 1' has elapsed, the user interface of the microwave oven is changed to its initial (idle) state. Launching other cooking programs (2,3,4 and 5) in the microwave oven is performed in similar way as in case of the 'Program 1'. User launches the desired cooking program in the Android application by pressing one of the buttons. For each program, the microwave oven's user interface is modified in its specific way: timer display shows different time, different power consumption of the device is shown and the status display shows the name of the currently launched program.
Moreover, when user presses the 'Stop' button, the execution of any running program is immediately terminated. The user interface of the microwave oven is also changed to its initial (idle) state.

As already discussed in the design and implementation chapters, Android application consists of user interfaces and NFC-I related functionalities which are implemented with the help of *Basic* and *Extended Communication Libraries*. In particular, the Android application of this use case utilizes the *Basic Communication Library* to implement the launching of cooking programs and their stopping. Getting the status information of the microwave oven required additional functionality which is provided by a new class defined as part of the *Extended Communication Library*. This class and its method are accessed directly from the user interface. When user presses the button for getting status information, a method in the defined class uses the functionalities provided by *Basic Communication Library* to send the command to the microwave oven. The response is also received and parsed within the method, and the status information is shown as response in the user interface.

NodeInt component of the NFC-I features a small green LED lamp which is used to visually present the user when the microwave oven is running one of the cooking programs. The LED glows green when one of the cooking programs is executed. As soon as the program ends or after it is stopped by the user, the LED is turned off. Besides, when the microwave oven is in idle operation state, the LED is turned off.

The visual appearance of the Android application's user interface which utilizes the washing machine as target device is almost the same compared to the user interface of the microwave oven. User actions in the Android application are also one-to-one equally handled by the NFC-I, only the user interface of the washing machine is updated differently. For this reason, the documentation does not contain the scenario of interaction between the Android application and the washing machine as target device.

5.3 NFC-I and Zero Standby Energy

Standby power consumption of household appliances and in general all electronic devices is the subject of many studies and initiatives. The amount of energy consumed by appliances being in standby mode is relatively small, yet if multiple appliances continuously operate in the standby mode, the total power consumption of a household is increased up to 14 percent [PSL+11]. Authors of the papers [CPD07] and [Nat06] report on energy losses caused by household appliances operating in the standby mode. Particularly interesting is the data on the power consumption of microwave oven and washing machine in standby off mode:

	Washing machine	**Microwave oven**
Minimum standby power [W]	0.00	0.00
Maximum standby power [W]	4.50	5.50
Average standby power [W]	0.90	3.00

Table 5.1: Standby Power Consumption of Washing Machine and Microwave Oven [Nat06].

The 1 Watt initiative was launched with the aim to reduce the standby power consumption from usual 3-10 Watts to less than 1 Watt [ML99]. On this issue there are certain results because major consumer electronics manufacturers have embraced the idea. However, the question raises whether the power consumption of household appliances can be totally eliminated. Paper [DMB+12] describes in detail how the standby energy consumption can be eliminated with the help of NFC-I.

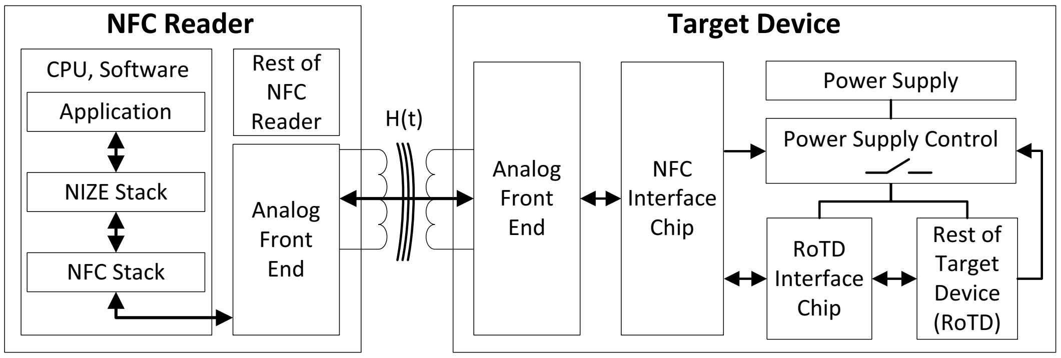


Figure 5.10: Android NFC-enabled Smart Phone and Target Device Enhanced with the Modified Version of NFC-I [DMB+12].

The architecture of the proposed system which eliminates the standby energy consumption, as depicted in the Figure 5.10, consists of Android NFC-enabled smart phone

(NFC Reader in the figure) and the target device enhanced with the modified version of NFC-I (depicted in figure as NFC Interface Chip and RoTD Interface Chip).

Android NFC-enabled smart phone is used to provide electrical power over the NFC module to induce an electrical voltage in the analog front end of the NFC-I. The induced voltage is rectified and electrical charges are buffered within the additional circuity. Such gained electrical power is transferred from the NFC-I to the switch on the target device's power supply, and from that moment, the target device uses its own power supply to perform its tasks. As soon as the target device completes the execution of a certain task, the connection between the power supply and the target device is interrupted by the power supply control unit.

This approach enables zero standby energy consumption of the target device and still, all NFC-I communicational capabilities, as described in the master thesis, are retained. All details on this topic and proof of concept can be obtained from the paper [DMB$^+$12].

Chapter 6

Conclusion

A fast growing market of smart homes and environments increasingly demands new and intelligent solutions whose purpose is to raise the quality of living. In addition, the focus is on optimized solutions which increase efficiency and reduce costs. In such environments, home control systems and home appliances are enhanced with internet connectivity and other wireless technologies. Also, smart phones and smart phone compatible solutions are becoming very popular each day.

This master thesis presents an innovative smart phone and, in general, NFC reader compatible solution that enables wireless communication for home appliances, consumer electronics and other similar applications. During the master thesis project, a Near Field Communication based Interface (NFC-I) is constructed and implemented. The main purpose of the NFC-I is to provide target home appliances or hardware with NFC technology. A special attention was paid to minimizing of construction costs, what was also achieved. Yet saving does not automatically imply reduced functionality or decreased user experience. In contrary, usability and user experience are improved by using Android NFC-enabled smart phone as input device to the NFC-I enhanced appliances or hardware. Android applications are designed and implemented to enable users interaction with NFC-I enhanced target devices via Android NFC-enabled smart phone. This allows users wirelessly connecting to the NFC-I enhanced target devices and activating, monitoring, controlling and configuring of the target device. Further advantage of using Android NFC-enabled smart phones is that additional savings can be achieved. Implemented Android applications provide users, for interacting with target appliances, rich graphical user interfaces which can completely replace the standard user interfaces of appliances.

In order to test the implemented interface, target devices were demonstrated in software on a PC. Such way of testing proved to give best results as it has provided a transparent view into all communication within the system. Although the tests were not performed with real appliances of hardware, the prototype version of the NFC-I is perfectly suited for that. Additionally, the NFC-I provides a number of ways for integration into appliances and hardware. In Results chapter, two implemented use cases present some capabilities of the NFC-I on the example of washing machine and microwave oven used as target devices. Areas where this, innovative and low-budget, NFC enabling interface can be successfully employed are many, ranging from smart homes and environments, industry, advertising, entertainment, etc.

New use cases also emerge from the fact that the NFC-I, in its modified version, can be used to enable zero standby energy consumption. This additional feature is especially applicable in home appliances but also in home control systems and other similar appliances where electronics constantly run and wait for user to provide input or perform certain activity. Savings which can be made in this way are very important, particularly because the total energy consumption in a home is heavily influenced by appliances consuming energy in standby mode.

6.1 Future Work

The prototype version of NFC-I fulfils all set requirements. However, there is a space for further improvements which can be made. The industrial partner in this project, Infineon Technologies Austria AG, has also shown a great interest in further development and improvement of the NFC-I.

6.1.1 Improved Version of the NFC-I

The current prototype version of the NFC-I is not optimized in terms of size. Security cryptocontroller provided by Infineon, which enables NFC communication for the NFC-I, is mounted on a DIP package. The DIP package is mounted on a circuit board which is equipped with antenna circuit. A Launchpad featuring MSP430 microcontroller is large in size and not all its features are required for the realization of the NFC-I. These circuit boards could be replaced with smaller sized ones in order to ease the installation into target appliances and hardware. By replacing the components which were used for the prototype version of the NFC-I, with the optimized components in terms of size and required features, overall construction costs could also be reduced. In mass production, the estimated construction costs for the final and market ready NFC-I could be lowered to just few Euro cents.

6.1.2 Testing with Real Appliances

Testing with real home appliances could primarily provide more accurate results for the implemented NFC-I. Based on realistic examples, where the users would interact with the real appliances over the Android NFC-enabled smart phones, the advantages of the implemented NFC system could be even better presented.

6.1.3 Secure Communication within the System

An important aspect in every wireless communication system is security. Further improvements could be achieved by implementing secured communication within the NFC system. All prerequisites for this are already met by choosing Infineon security cryptocontroller which provides crypto coprocessors for both symmetric and asymmetric cryptography. Also, the Android NFC-enabled smart phone, used for the realization of the project, provides support for a variety of cryptographic algorithms. Particularly, the Elliptic Curve based cryptography is well suited for this NFC system. By implementing secure communication, additional opportunities for the employment of the implemented NFC system would be created. Primarily, the NFC-I could be utilized in payment solutions, for example by providing payment terminals with secure wireless communication.

6.1.4 Support for other NFC-enabled Smart Phone Platforms

In this thesis and in the implemented project, focus is on the Android mobile platform. However, other mobile platforms including BlackBerry, Windows phone and Nokia provide support for NFC technology. Further step in the improvement of the NFC-I would be to adopt it for the mentioned mobile platforms and to provide basic libraries for the implementation of applications.

Appendix A

Android Development Environment Installation

The procedures for the preparation of the Android development environment are presented in the following steps:

1. If Java Development Kit (JDK) is not already installed on the host computer, it is necessary to download and install the JDK6 or the latest JDK7[1].

2. Android developers recommend using Eclipse, a powerful open source multi language IDE, because it features an Android Development Tools (ADT) plug-in, which simplifies the development of Android applications. It is recommended to install a 'classic' version of Eclipse, which is, at the time of writing the document, Eclipse Classic 3.7.1[2].

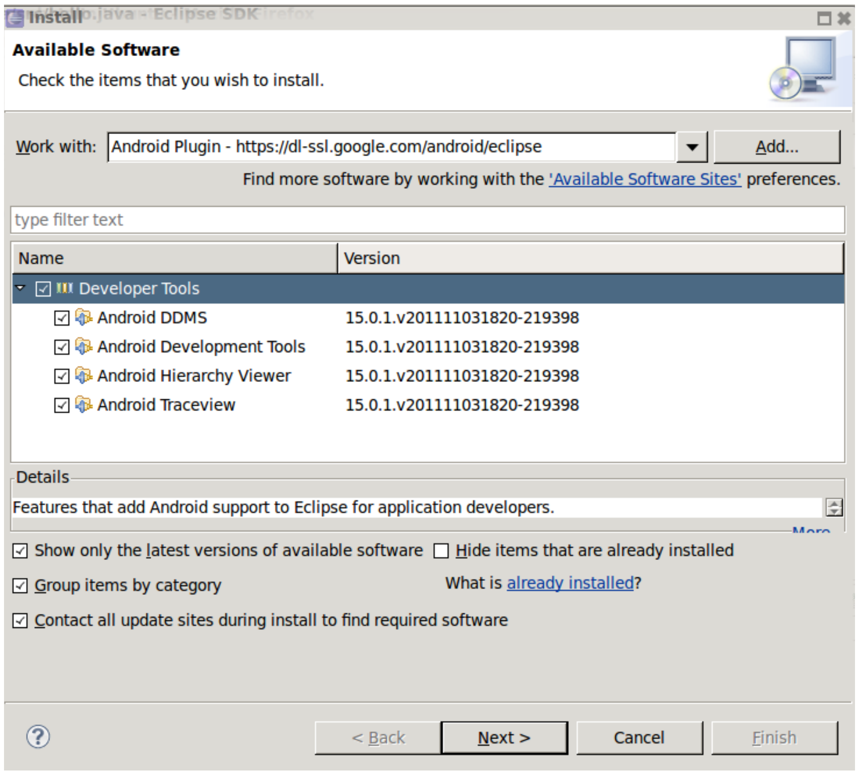
3. After successful installation of JDK and Eclipse, the next step is to install Android SDK on the computer. This step involves downloading and unpacking of the appropriate Android SDK package which can be found on the Android web site[3].

4. The next step is the installation of the Eclipse ADT plug-in. It is necessary to open Eclipse, select **Help -> Install New Software** and as shown in the Figure A.1, enter the remote location of the plug-in and install Android Developer Tools.

5. Eclipse must know the location of the unpacked Android SDK, and that is done by browsing and selecting the appropriate folder: in Eclipse, **Window -> Preferences -> Android**.

6. By accessing Android SDK Manager in Eclipse (**Window -> Android SDK Manager**), user needs to choose and install Android API and tools (Android SDK Tools, Android SDK Platform-tools). It is possible to install all versions of Android APIs, but since the practical part of this thesis involves working with a smart phone where the version of Android is 4.1.2, the choice fell to install and use Android APIs 4.1.2 (Level 14).

[1] `http://www.oracle.com/technetwork/java/javase/downloads/index.html`
[2] `http://www.eclipse.org/`
[3] `http://developer.android.com/sdk/index.html`


Install

Available Software

Check the items that you wish to install.

Work with: Android Plugin - https://dl-ssl.google.com/android/eclipse

Find more software by working with the 'Available Software Sites' preferences.

type filter text

Name	Version
☑ Developer Tools	
☑ Android DDMS	15.0.1.v201111031820-219398
☑ Android Development Tools	15.0.1.v201111031820-219398
☑ Android Hierarchy Viewer	15.0.1.v201111031820-219398
☑ Android Traceview	15.0.1.v201111031820-219398

Details

Features that add Android support to Eclipse for application developers.

☑ Show only the latest versions of available software ☐ Hide items that are already installed

☑ Group items by category

What is already installed?

☑ Contact all update sites during install to find required software

Figure A.1: Eclipse: Installation of the ADT Plug-in.

7. The last step in preparing of the development environment is creating of the Android Virtual Device (AVD) in Eclipse (**Windows -> AVD Manager**), which is, simply said, the configuration for the emulator.

8. If everything is installed and configured correctly, after creating and starting of a sample Android project in Eclipse, (**File -> New -> Android Project -> Create new project from existing sample**), a new window with Android Emulator, as depicted shown in Figure A.2, appears on the screen.

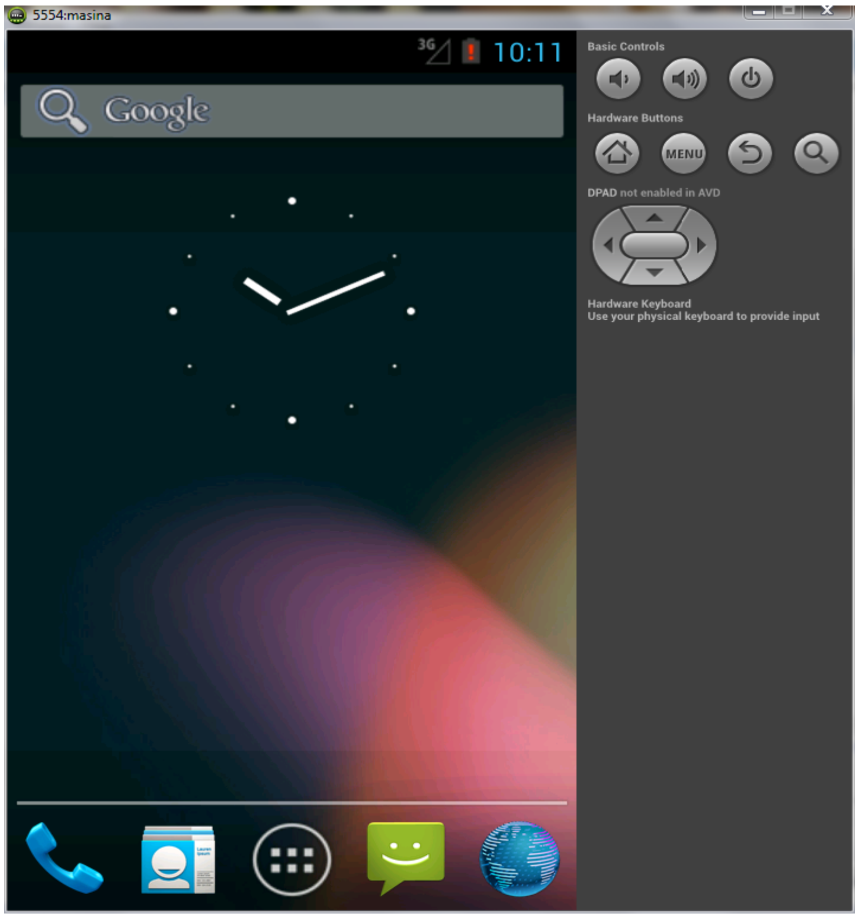


Figure A.2: Eclipse: Android Emulator.

Bibliography

[AITH09] Heikki Ailisto, Minna Isomursu, Tuomo Tuikka, and Juha Häikiö. Experiences from interaction design for NFC applications. In *Journal of Ambient Intelligence and Smart Environments*, volume 1, pages 351–364, December 2009.

[All07] Open Handset Alliance. Open Handset Alliance Releases Android SDK. `http://www.openhandsetalliance.com/press_111207.html`, 2007.

[And11a] Research2guidance AndroidPIT. Android Market Insights, April 2011.

[And11b] Research2guidance AndroidPIT. Android Market Insights, Vol. 6, October 2011.

[And13a] Android. Android Developing. `http://developer.android.com/guide/developing/index.html`, 2013.

[And13b] Android. Android NDK. `http://developer.android.com/sdk/ndk/overview.html`, 2013.

[And13c] Android. Android NFC API. `http://developer.android.com/reference/android/nfc/package-summary.html`, 2013.

[And13d] Android. Android Source Code. `http://source.android.com/index.html`, 2013.

[And13e] Android. What is Android? `http://developer.android.com/guide/basics/what-is-android.html`, 2013.

[Blo10] Patently Apple Blog. NFC iPhone to Control All of Your In-Home Electronics & More. `http://www.patentlyapple.com/patently-apple/2010/04/nfc-iphone-to-control-all-of-your-in-home-electronics-more.html`, 2010.

[Bur08] E. Burnette. *Hello, Android.* Pragmatic Bookshelf Series. Pragmatic Programmers, LLC, 2008.

[But11] M. Butler. Android: Changing the Mobile Landscape. In *IEEE Pervasive Computing Magazine (M-PVC)*, volume 10, pages 4–7, January-March 2011.

[CPD07] K. Clement, I. Pardon, and J. Driesen. Standby Power Consumption in Belgium. In *International Conference on Electrical Power Quality and Utilisation*, pages 1–4, October 2007.

[CPL12] Longbiao Chen, Gang Pan, and Shijian Li. Touch-driven Interaction via an NFC-enabled Smartphone. In *2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 504–506, March 2012.

[DMB$^+$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)*, pages 261–267, October 2012.

[ES06] M. England and M. Seminar. *Elliptic curve cryptography.* PhD thesis, Heriot-Watt University, 2006.

[For11] NFC Forum. Type 4 Tag Operation Specification - Technical Specification. `http://www.nfc-forum.org/specs/spec_list/`, 2011.

[For13] NFC Forum. `http://www.nfc-forum.org/`, 2013.

[Gar11] Gartner.com. Market Share: Mobile Communication Devices by Region and Country, 3Q11. `http://www.gartner.com/it/page.jsp?id=1848514`, 2011.

[Goo08] Google. Google I/O 2008 - Anatomy and Physiology of an Android. `https://sites.google.com/site/io/anatomy--physiology-of-an-android`, 2008.

[Ins12a] Texas Instruments. MSP430 LaunchPad. `http://www.ti.com/LaunchPad`, 2012.

[Ins12b] Texas Instruments. MSP430G2553 - Mixed Signal Microcontroller. `http://www.ti.com/product/msp430g2553`, 2012.

[ISO08] ISO. *ISO/IEC 14443 Parts 1-4: Identification cards - Contactless integrated circuit(s) card - Proximity cards*, 2001-08.

[ISO98] ISO. *ISO/IEC 7816 Parts 1-15: Identification cards - Integrated circuit(s) cards with contacts*, 1998.

[ISO05] ISO. *ISO/IEC 7816 Part 4: Identification cards - Cards with contacts Organization, security and commands for interchange*, 2005.

[ISO06] ISO. *ISO/IEC 7816 Part 3: Identification cards - Cards with contacts Electrical interface and transmission protocols*, 2006.

[JMV01] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). In *International Journal of Information Security*, volume 1, pages 36–63. Springer, 2001.

[Kob87] N. Koblitz. Elliptic Curve Cryptosystems. In *Mathematics of computation*, volume 48, pages 203–209, 1987.

[Lau04] K. Lauter. The advantages of elliptic curve cryptography for wireless security. In *IEEE: Wireless Communications*, volume 11, pages 62–67. IEEE, 2004.

[Li10] N. Li. Research on Diffie-Hellman Key Exchange Protocol. In *2nd International Conference on Computer Engineering and Technology (ICCET)*, volume 4, pages 634–637. IEEE, April 2010.

[LJ10] S. Lee and J.W. Jeon. Evaluating erformance of Android platform using native C for embedded systems. In *International Conference on Control Automation and Systems (ICCAS)*, pages 1160–1163. IEEE, October 2010.

[Mic11] Michaels, Ross & Cole, ltd. The Beginners Guide to Creating Mobile Applications for your Business. `http://www.mrc-productivity.com`, 2011.

[Mil86] V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology-CRYPTO'85 Proceedings*, pages 417–426. Springer, 1986.

[ML99] Alan Meier and Benot LeBot. One Watt Initiative: A Global Effort to Reduce Leaking Electricity. IEA - International Energy Agency, 1999.

[Mui12] M. Mui. NXP Smart Washing Machine Solution. `http://www.nxp.com/documents/other/NXP_total_solutions_in_Smart_Home_Appliance.pdf`, 2012.

[Nat06] National Appliance and Equipment Energy Efficiency Program. Appliance Standby Power Consumption - Store Survey 2005/2006 - Final Report. `http://www.energyrating.gov.au/wp-content/uploads/Energy_Rating_Documents/Library/Standby_Power/Standby_Power/200607-storesurvey.pdf`, August 2006.

[NF08] NFC-Forum. Essentials for Successful NFC Mobile Ecosystems. `http://www.nfc-forum.org/resources/white_papers/NFC_Forum_Mobile_NFC_Ecosystem_White_Paper.pdf`, 2008.

[NXP09] NXP. NFC Forum Type Tags White paper V1.0. `http://www.nfc-forum.org/resources/white_papers/NXP_BV_Type_Tags_White_Paper-Apr_09.pdf`, 2009.

[NXP11a] NXP. P5CD016/021/041/051 and P5Cx081 family Secure dual interface and contact PKI smart card controller. Product data sheet, 2011.

[NXP11b] NXP. PN512 - Transmission module. Product data sheet, 2011.

[NXP13a] NXP. MWC 2013 Prototype Demonstration: Smart Wireless Charging Pad. `http://www.nxp.com/wcm_documents/technologies/documents/smart_wireless_charging_pad.pdf`, 2013.

[NXP13b] NXP. NXP Heralds a Secure, Contactless and Multimedia Future at MWC 2013. `http://www.nxp.com/news/press-releases/2013/02/nxp-heralds-a-secure-contactless-and-multimedia-future-at-mwc-2013.html`, 2013.

[PE11] Shailendra Pandey and Guillermo Escofet. *White paper: Remote Payments Drive Near-Term M-Commerce Revenue Opportunity for Mobile Operators*. `http://www.trustvesta.com/datasheets/remote_payments_informa_2011.pdf`, 2011.

[P.E12] Ross P.E. Phone-y money. In *IEEE Spectrum*, volume 49, pages 60–63, June 2012.

[PK10] K. Paul and T.K. Kundu. Android on Mobile Devices: An Energy Perspective. In *IEEE 10th International Conference on Computer and Information Technology (CIT)*, pages 2421–2426. IEEE, July 2010.

[Poi11] PointAbout. A look into the (Near) Future of NFC. `http://www.latinia.com/static/NFC_future.pdf`, 2011.

[PS12] J. Potts and S. Sukittanon. Exploiting Bluetooth on Android mobile devices for home security application. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–4, March 2012.

[PSL$^+$11] A. Prudenzi, A. Silvestri, R. Lamedica, M. C. Falvo, U. Grasselli, and M. Regoli. Standby Power End Users and Their Demand Management Impact on Households Load Shape. In *Power and Energy Society General Meeting*, pages 1–6. IEEE, July 2011.

[Res11a] Research Industrial Systems Engineering - RISE. NXP Chip PN65. `http://www.nfc.cc/technology/nxp-nfc-chips`, 2011.

[Res11b] Research Industrial Systems Engineering - RISE. NXP FRI. `http://www.nfc.cc/technology/nxp-fri`, 2011.

[RTI12] R.A. Ramlee, D.H.Z. Tang, and M.M. Ismail. Smart Home System for Disabled People Via Wireless Bluetooth. In *2012 International Conference on System Engineering and Technology (ICSET)*, pages 1–4, September 2012.

[Sam11] Samsung. Samsung Nexus S. `http://www.samsung.com/us/mobile/cell-phones/GT-I9020FSTTMB`, 2011.

[Sma11a] SmartOnline. Overview: US Smart phone statistics Q1 2011. `http://www.mobilesmith.com/mobile-2/looking-back-at-2011-the-year-of-mobile/`, 2011.

[Sma11b] SmartOnline. White Paper: A mobile World: how your supporters are using their Smart phones and why you should care., 2011.

[Stu11] John Stultz. Android OS for Servers. `http://www.linaro.org`, 2011.

[Sup12] Wayan Suparta. Application of Near Field Communication Technology for Mobile Airline Ticketing. In *Journal of Computer Science*, volume 8, pages 1235–1246. Science Publications, 2012.

[TV12] Nguyen-Vu Truong and Duc-Lung Vu. Remote monitoring and control of industrial process via wireless network and Android platform. In *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 340–343, November 2012.

[Wan11] R. Want. Near Field Communication. *IEEE Pervasive Computing*, pages 4–7, July-September 2011.

[WGSL12] Rainer Widmann, Stefan Grnberger, Burkhard Stadlmann, and Josef Langer. System Integration of NFC Ticketing into an Existing Public Transport Infrastructure. In *4th International Workshop on Near Field Communication (NFC)*, pages 13–18, March 2012.

[Wik11] Embedded Linux Wiki. Android Binder. `http://elinux.org/Android_Binder`, 2011.