AN11480

Quick Start-up Guide for EXPLORE-NFC working with Raspberry Pi

Rev. 2.5 — 13 July 2016
279725

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	PN512; Raspberry Pi, NFC, P2P, MIFARE, ISO/IEC 14443, EXPLORE-NFC
Abstract	This document gives a description on how to get started with the provided software packages in conjunction with the EXPLORE-NFC board and the Raspberry Pi model A, B, B+ and Raspberry Pi 2/3 model B. It provides a step by step guide to the installation procedure of the software and the hardware.

Revision history

Rev	Date	Description
2.5	20160713	Updated references to Raspberry pi system image. Added reference to Raspberry Pi 3
2.4	20150907	Added description for write option and copy option in the Basic example. Supplemented option descriptions in all examples. Updated section 10.3 Licenses
2.3	20150407	Updated description of the Basic example. Added description of the Card Emulation example. Added description of the nxppy Python wrapper.
2.2	20150309	Added description of the WiFi example.
2.1	20150217	Corrected two typos in the installation chapter.
2.0	20150212	New approach of communicating and using the EXPLORE-NFC. The Architecture is fully compatible with NeardAL API from the Linux-NFC project.
1.0	20131217	First release

Contact information

For more information, please visit: http://www.nxp.com

1. Introduction

This document gives a description on how to get started with the provided software packages in conjunction with the EXPLORE-NFC board and the Raspberry Pi model B+. All steps can be applied in the same way on the Raspberry Pi model A and Raspberry Pi 2/3 model B.

This document provides a step by step guide to the installation procedure of the software and the hardware.

Fig 1. EXPLORE-NFC and MIFARE Ultralight card

1.1 What is EXPLORE-NFC?

EXPLORE-NFC is a high performance fully NFC compliant expansion board for the Raspberry Pi. Based on the NXP PN512 solution, EXPLORE-NFC meets compliance with Reader mode, P2P mode and Card emulation standards. The board features an integrated high performance antenna and offers a flexible SPI interface. Software packages supporting all these features are described within this manual.

1.2 What is a Raspberry Pi?

The Raspberry Pi is a credit card sized computer. The initial idea behind it was to develop a small and cheap computer to be used by kids all over the world to learn programming. In the end it became very popular among developers all over the world.

To get started quickly, the Raspberry Pi Foundation provides several preconfigured Linux distributions.

For the project described in this guide we are using the “Raspbian Jessie” [7.].

For more information about the Raspberry Pi please visit [8.].

1.3 NXP NFC Reader Library and Linux

The software described in this guide is based on the NXP NFC Reader Library. The NXP NFC Reader Library is written in C language enabling the customers to create their own software stack for their contactless reader.

To get the NXP NFC Reader Library to work on Linux, some of the relevant parts have been ported to Linux. At the moment, only SPI protocol is ported for the communication between reader IC and the Raspberry Pi.

The intention of this project is to give an exemplary implementation of the NXP NFC Reader Library on Linux. It should easily be possible to apply/install the Reader Library of this project on other Linux systems.

2. Software installation

2.1 Required items

- Compatible SD card [1.] with a size of at least 4GB.
- Card reader

2.2 Downloading the system image

Please download Win32DiskImager and the Raspbian “Jessie” image from the download page of the Raspberry Pi foundation [6.].

2.3 Preparing the SD card

1. Unzip the archive containing the system image.
2. Insert the SD card into a card reader connected to the PC.
3. Start Win32DiskImager.
4. Chose the unzipped *.img file.
5. Ensure that the correct device is chosen.
6. Click onto the Write button.

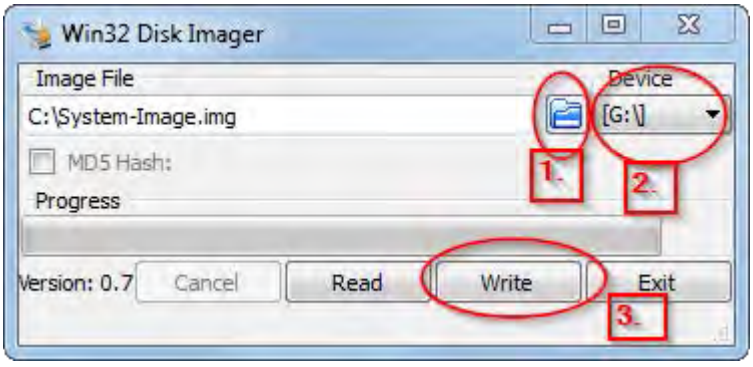

Fig 2. Win32 Disk Imager

2.4 Downloading the software

1. Download the software either from the element14 web site [4.] or from the NXP web site [3.].
2. Copy the zip compressed project onto an USB Stick.

3. Hardware installation

Required items

- Raspberry Pi Model B or B+ or Raspberry Pi 2/3
- Contactless reader board EXPLORE-NFC
- Compatible SD card [1.]
- Micro USB power supply (5V / 1A) [2.]
- Keyboard
- Mouse
- HDMI cable to connect to a Monitor / TV

Please connect the hardware in the following order:

1. Connect the EXPLORE-NFC board.
   Caution: Please take care to connect the extension-board in the right way! See the next picture for more information.
2. Put the SD card into the card reader of the Raspberry Pi.
3. Connect the Raspberry Pi to a Monitor.
4. Connect the Keyboard and the Mouse to the USB ports of the Raspberry Pi.
5. Connect the USB power adapter to the Raspberry Pi and plug it into a socket.

Please wait until the Raspberry Pi is booted and the login screen appears.

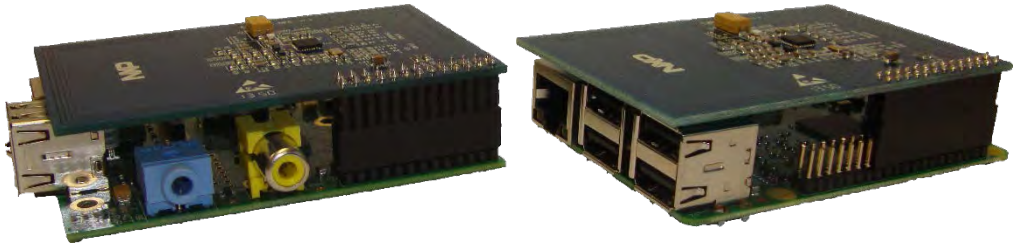

Fig 3. EXPLORE-NFC and Raspberry Pi stacked together
Right: Raspberry Pi Model B
Left: Raspberry Pi 2/3 Model B

4. Login to the Raspberry Pi and starting the software

This section describes how to install and run the software for the EXPLORE-NFC demo board. All needed software components are being delivered as installable Debian packages to be installed directly on the Raspberry Pi.

4.1 Login to the Raspberry Pi

1. Type in username and password, whereas the **username** is **pi** and the **password** is **raspberry.**
2. After the command line appears type in **startx** and hit [Enter]. Wait until the graphical desktop environment has finished loading.

4.2 Activate SPI

If this is the first start, the configuration menu for the Raspberry Pi will appear automatically. Otherwise type the following command to open that menu:

```
sudo raspi-config
```

The option to activate SPI can be found as follows:
Advanced Options → SPI → <Yes>.

Now reboot your Raspberry Pi by typing the following command:

```
sudo reboot
```

4.3 Installing and starting the NFC application

1. Copy the zip compressed software onto an USB Stick.
2. Connect the USB Stick to the Raspberry Pi and unzip the project to the home folder of the pi user.
3. Start the terminal emulator with double clicking on the symbol **LXTerminal** on the desktop.
4. Change into the directory that contains the extracted files with the command:

   ```
   cd ~/<directory of extracted files>/
   ```

5. Install the files with the command:

   ```
   sudo dpkg -i libneardal0_<version>_armhf.deb neard-
   explorenfc_<version>_armhf.deb
   ```

   Where “<version>” needs to be substituted with the real version strings contained in the file names.
6. Start the desired example project with the command

   ```
   explorenfc-<example name>
   ```

5. Included example projects

5.1 Basic example

The Basic example shows how to read a tag or receive a message over peer to peer and display information about it. The message needs to be NDEF encoded. It also shows how to write a tag or push a message to a device and how to duplicate a message.

To run the read example please type the following command:

```
explorenfc-basic
```

To run the write example please type the following command:

```
explorenfc-basic -w '<your message>' -t <type>
```

To copy a message to following tags or devices please type:

```
explorenfc-basic -c -k
```

Options:

-h, --help	Display the help to all options of the example
-k, --keep-polling	Keep the program running (does not exit on first tag detection)
-a, --adapter	Use specified adapter (defaults to /org/neard/nfc0)
-w, --write	Write specified message into tag or push to device
-t, --type	Message type (Text, URI or SmartPoster are supported) (only for use with write option)
-l, --language	(small letter L) Language of record (defaults to en) (only for use with write option)
-c, --copy	Copy first record to subsequent tags or devices (option -k is compulsory!)

Example output writing a tag:

```
pi@raspberrypi ~ $ explorenfc-basic -w 'NXP Semiconductors' -t Text -l en
Waiting for tag or device...
on_name_lost(): :org.neardal
Found record /org/neard/nfc0/tag0/record0
Record type:    URI
URI:    http://nxp.com
Title:
Action:
Language:
Encoding:       UTF-8
Tag found
Record was written successfully
```

Example output reading the previous tag:

```
pi@raspberrypi ~ $ explorenfc-basic
Waiting for tag or device...
on_name_lost(): :org.neardal
Found record /org/neard/nfc0/tag0/record0
Record type:    Text
URI:
Title:  NXP Semiconductors
Action:
Language:       en
Encoding:       UTF-8
Tag found
ISO14443A ATQA:         4400
ISO14443A SAK:  00
ISO14443A UID:  0401C4495B2381
```

5.2 Browser example

The Browser example will read a tag or receive an NDEF message containing an URL over peer to peer and open it in the default browser. You need to have a graphic interface session running for this to work.

To run the example please type the following command:

```
explorenfc-browser
```

Options:

-h, --help	Display the help to all options of the example
-k, --keep-polling	Keep the program running (does not exit on first tag detection)
-a, --adapter	Use specified adapter (defaults to /org/neard/nfc0)

Example output:

pi@raspberrypi ~ $ explorenfc-browser

Waiting for tag or device...

on_name_lost() : :org.neardal

Opening http://nxp.com

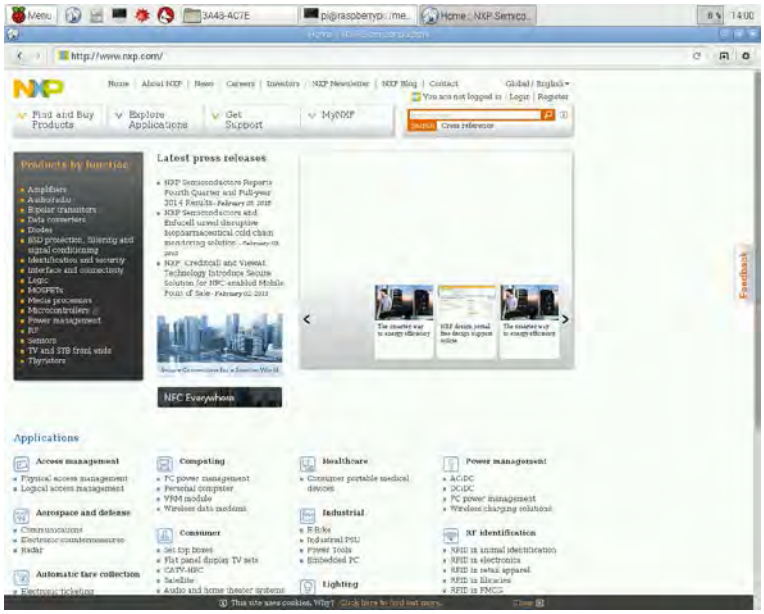

Fig 4. Browser example

5.4 WiFi example

This example shows how to read a tag or receive an NDEF message containing WiFi connection data over peer to peer and pass relevant data to WPA Supplicant so that your Raspberry Pi can connect to the network. If you are not using the Rasperry Pi 3 you need to have a WiFi adapter plugged in and configured.

An application to store WiFi information to a tag is for example the Android application TagWriter from NXP.

Note: In some cases the WPA Supplicant might not work correctly with the WiFi dongle plugged in. In our case it often helped to add the statement
wpa-driver wext
to the configuration file /etc/wpa_supplicant/wpa_supplicant.conf

Note: In Raspbian WPA Supplicant's DBUS interface is not activated by default, when explorenfc-wifi-connect now tries to connect to a network, another WPA Supplicant instance would be spawned.
The solution is to prevent the wlan* interface from being managed by ifup/ifdown scripts and handle the connection process in explorenfc-wifi-connect.
This can be configured by removing any definition to the wlan* interface that should be used in /etc/network/interfaces.

An example config could look like:

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp


iface default inet dhcp
```

Options:

-h, --help	Display the help to all options of the example
-k, --keep-polling	Keep the program running (does not exit on first tag detection)
-a, --adapter	Use specified adapter (defaults to /org/neard/nfc0)
-i, --interface	Use a specific network interface (defaults to wlan0)

Example output:

```
pi@raspberrypi ~ $ sudo explorenfc-wifi-connect

Waiting for tag or device...

on_name_acquired() : :org.neardal
```

SSID: AndroidAP

MAC: C4:53:8F:AD:BF:C2

Authentication: WPA2-PSK

Encryption: AES

Key: 791c3232e4f9

Tag found

wlan0: scanning

wlan0: associating

wlan0: associated

wlan0: 4way_handshake

wlan0: completed

Killing existing DHCP clients...

Running DHCP client...

Internet Systems Consortium DHCP Client 4.2.2

Copyright 2004-2011 Internet Systems Consortium.

All rights reserved.

For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlan0/80:1f:02:af:48:c8

Sending on LPF/wlan0/80:1f:02:af:48:c8

Sending on Socket/fallback

DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 6

DHCPREQUEST on wlan0 to 255.255.255.255 port 67

DHCPOFFER from 192.168.43.1

DHCPACK from 192.168.43.1

bound to 192.168.43.63 -- renewal in 1418 seconds.

Connected!

5.5 Source code of the delivered software

All source code is available for download either from the element14 web site [4.] or from the NXP web site [3.].

For usage instructions please consult the Readme.md which is packaged within the sources package (neard-explorenfc_<version>.orig.tar.gz).

6. Software architecture

6.1 Introduction

Neard - ExploreNFC is a Linux daemon that provides a DBUS API for easy access to the EXPLORE-NFC's board functionality. This DBus Daemon provides a DBUS API compatible with Neard but uses the NXP NFC Reader Library (NXPRdLib) as a backend.

Things you can do with Neard - ExploreNFC:

- Read NFC tags
- Write NFC tags
- Push NDEF messages to another NFC device (P2P)
- Receive NDEF messages from another NFC device (P2P)

6.2 Overall Architecture

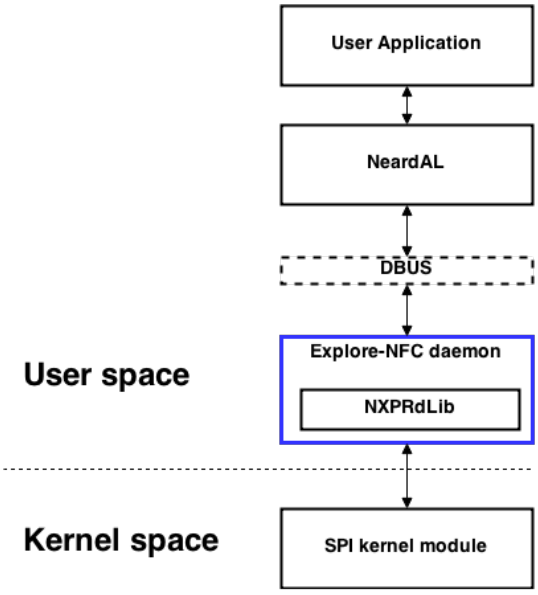

Fig 5. Neard-ExploreNFC in context

The ExploreNFC software has been implemented in order to maximise compatibility with the Neard DBUS API.
NeardAL is a library that provides access to the Neard DBUS API and it has been ensured that it is compatible with the Explore-NFC implementation.

The Explore-NFC software makes use of the NXPRdLib for low-level NFC operations. In order to communicate with the PN512 NFC transceiver using GPIOs and SPI the appropriate kernel modules are used.

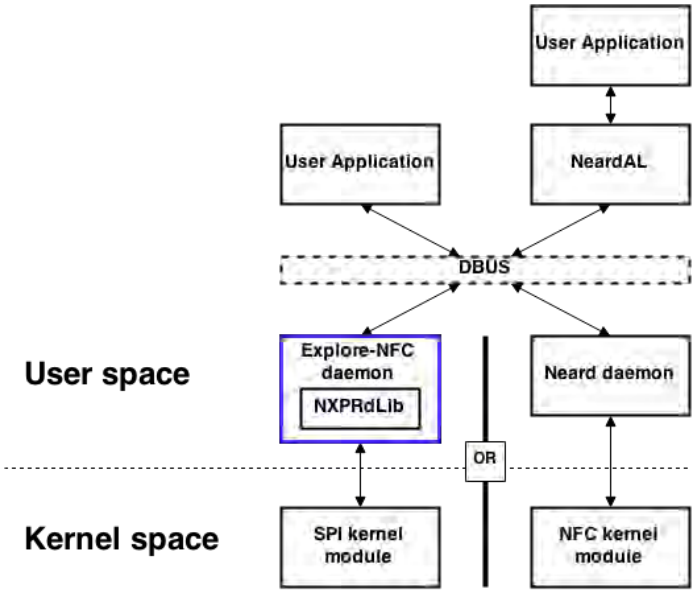

Fig 6. Compatibility with Neard and NeardAL

6.3 Explore-NFC daemon's architecture

The daemon is split in different modules running in two different threads.
The first thread handles the DBUS server and the second thread handles the interaction with the reader library (and hence the NFC hardware).
Both threads run a GLib loop.

6.3.1 Threads perspective

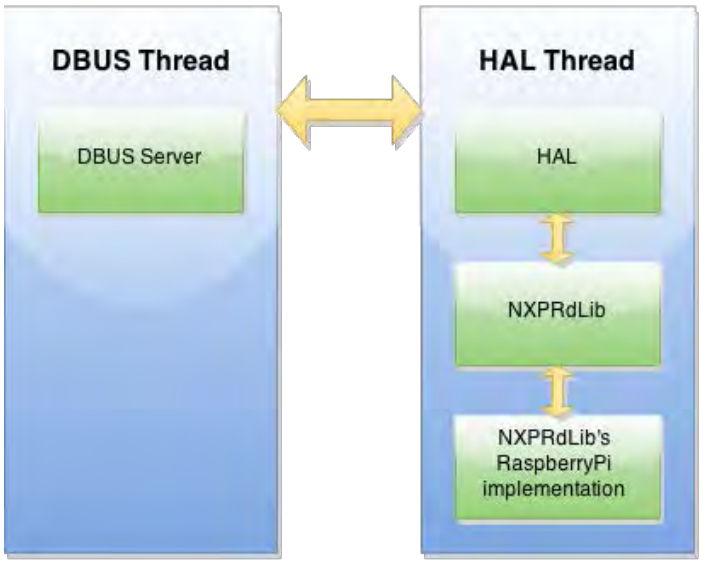

Fig 7. Threads architecture

6.3.1.1 DBUS Thread

The DBUS server implements the Neard DBUS API using GLib's DBUS features.
It starts by obtaining ownership of the "org.neard" DBUS name. It then exposes a standard DBUS interface that can be used to discover interfaces offered within this namespace. It also registers an Adapter interface, which exposes the appropriate DBUS API and communicates with the HAL module. As needed, the DBUS thread exposes Tag and Devices interfaces over DBUS and connects to appropriate Handover Agents that handle the connectivity to other networks such as Wi-Fi.

6.3.1.2 HAL Thread

This module configures the NXPRdLib for proper operation and runs the polling loops needed to discover tags and devices. If a tag is detected it reads its content, and does regular presence checks. Additionally it can write content to the tag.
If a device is detected it brings up the LLCP layer as needed, attempts a connection to a remote SNEP server and waits for a SNEP connection on the local server. NDEF messages can then be exchanged.

The NXPRdLib relies on a specific OS abstraction layer based on GLib and a hardware abstraction layer based on wiringPi.

6.3.1.3 Interaction between threads

Two types of interactions are used to communicate between the two threads.
For properties reading, mutexes are used to protect concurrent access from both threads.
To pass on a message, one thread will allocate some memory that contains the parameters corresponding to the message and will queue the execution by the other thread's GLib loop of a method that handles these parameters.

6.3.2 DBUS Module

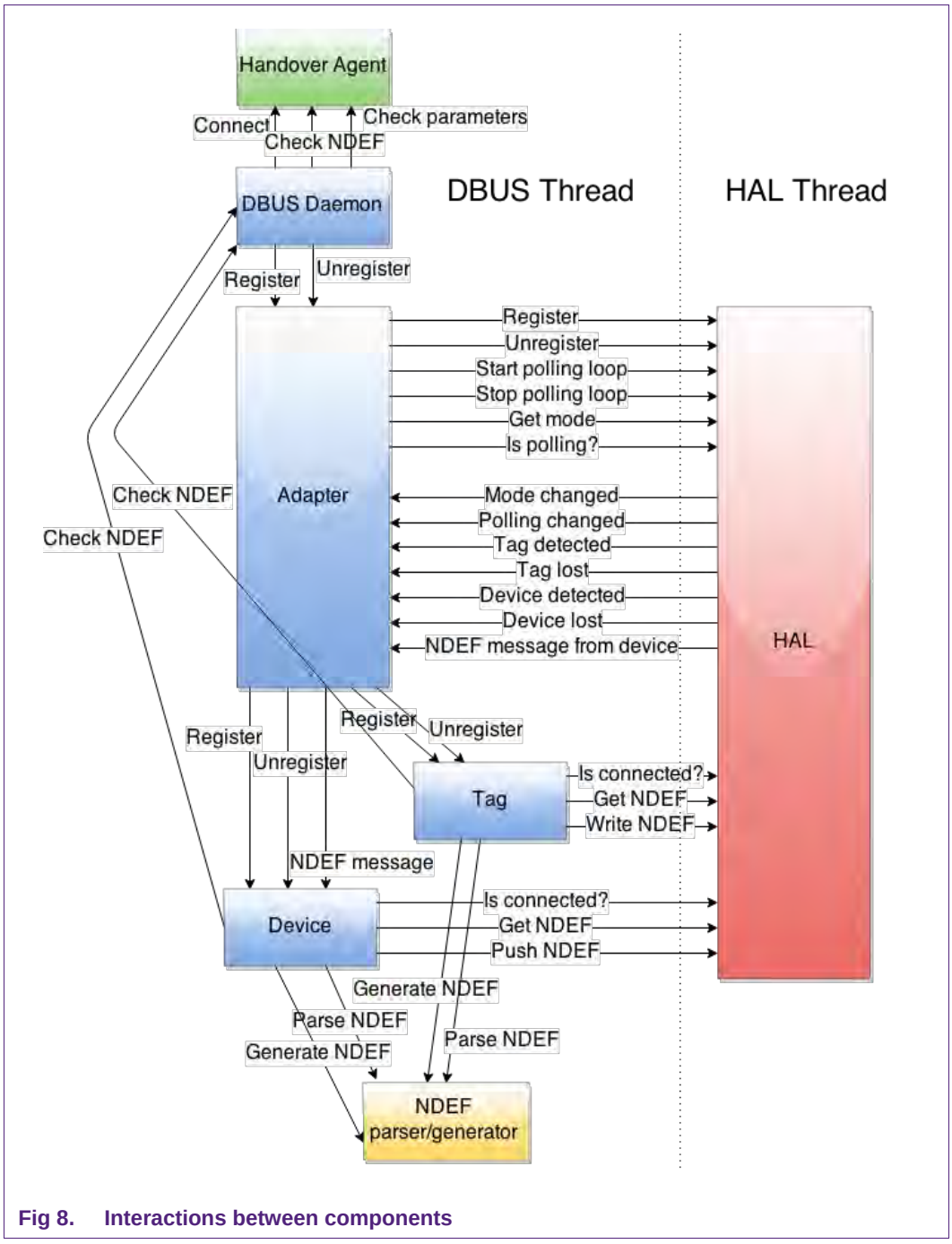

Fig 8. Interactions between components

As seen in the figure above, the DBUS server module is actually made of multiple sub-modules.
Blue boxes represent modules that expose a DBUS API (servers) and the green box represent a module that connects to a remote DBUS server.

6.3.2.1 DBUS Daemon

This module is the DBUS thread's entry point and exposes a DBUS API that allows discovery of other DBUS modules. It tries to obtain the "org.neard" name at startup. If successful, it instantiates and registers an Adapter instance. This module also handles registration requests from handover agents.

6.3.2.2 Adapter

This module exposes the Neard's Adapter DBUS API. It sends commands to the HAL module, reads its status and registers a set of callbacks to receive event messages back from the HAL module. When a tag or device is detected it instantiates and registers the corresponding module.

6.3.2.3 Tag

When this module is instantiated it receives a raw NDEF message that is parsed and exposed on the DBUS API. The Tag module also handles requests to write to a tag and generates the appropriate NDEF message before doing so. When the NDEF message is parsed, it is passed to the DBUS Daemon module. A handover agent needs to handle it.

6.3.2.4 Device

This module is very similar to the Tag module. The significant difference is that NDEF messages can be received after the device is detected.

6.3.2.5 Handover Agent

This module connects to a DBUS server exposing the handover agent interface. When a NDEF message is received, the module checks if it can handle it and if so it is passed to the remote handover agent which can use the NDEF message to setup a wireless connection such as Wi-Fi.

7. Additional software

7.1 Card Emulation example

The Card Emulation example emulates an NFC Forum Type 2 Tag. The emulated tag can be used to be read or to be written with another device. While the EXPLORE-NFC is in card emulation mode, it does not generate its own RF field. The field needs to be generated by the counterpart like an NFC reader or an NFC enabled mobile device.

Currently the Card Emulation example does not make use of the Neard - ExploreNFC daemon. Because of that, please make sure that the daemon is not running while using the Card Emulation example.

7.1.1 Installation

1. Download the software either from the element14 web site [4.] or from the NXP web site [3.].
2. Copy the zip compressed software onto an USB Stick.
3. Connect the USB Stick to the Raspberry Pi and unzip the project to the home folder of the pi user.
4. Start the terminal emulator with double clicking on the symbol **LXTerminal** on the desktop.
5. Change into the directory that contains the extracted files with the command:

   ```
   cd ~/<directory of extracted files>/
   ```

6. Install the files with the command:

   ```
   sudo dpkg -i libwiringpi2-<version>_armhf.deb explorenfc-
   cardemulation_<version>_armhf.deb
   ```

   Where “<version>” needs to be substituted with the real version strings contained in the file names.
7. Start the Card Emulation example with the command
8. `explorenfc-cardemulation`

7.1.2 Usage

When executing the application with a text message appended, it creates a valid NDEF message containing the text to be emulated. If nothing is appended, it emulates a standard message.

When writing a message to the emulated tag, the application makes a print out of the provided message to stdout. That makes it possible to use the Card Emulation application from another application to exchange messages via NFC.

Options:

-t <”message to emulate”>	Emulates the appended message encoded in NDEF format.

Note: Please make sure the Neard - ExploreNFC daemon is not running while using the Card Emulation example. It can be stopped with the following command:

```
sudo service neard-explorenfc stop
```

After each reboot, the daemon is automatically started.

7.2 nxppy

For people that would rather develop in Python instead of C programming language might be interested in the software project nxppy.

nxppy is a Python wrapper for interfacing with the EXPLORE-NFC. It takes the NXP NFC Reader Library and provides a thin layer for detecting MIFARE tags, reading their UID or performing Read/Write operations. The project can be found at GitHub [9.].

8. Supplementary notes

8.1 Turning off the Raspberry Pi

The Raspberry pi has no power switch. To turn it off, please type **sudo halt** into the command prompt. After the screen turns black, you may unplug the power supply.

8.2 Resizing the file system

In case one uses an SD card larger than 4GB it might be reasonable to resize the root file system to use the whole space of the SD card. To do so, please follow these steps:

1. Start the Raspberry Pi and log in.
2. Type **sudo raspi-config** into the command prompt.
3. In the configuration dialog that opens, choose the entry “expand_rootfs” and press [Enter].

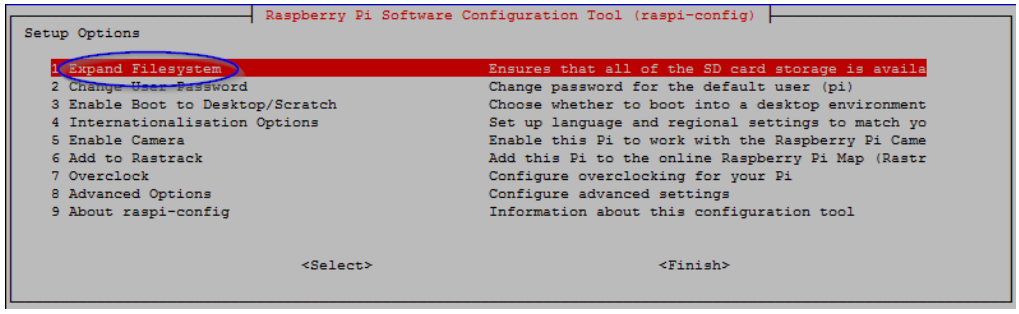

Fig 9. Expand rootfs in Raspi-config

4. After the operation has finished, choose <Finish> by pressing the [TAB] key twice and press [Enter].
5. Answer the question about the reboot with <Yes>.

9. References

[1.] **List of verified SD cards:**
http://www.nxp.com/redirect/elinux.org/RPi_SD_cards

[2.] **List of verified USB power adapters:**
http://www.nxp.com/redirect/elinux.org/RPi_VerifiedPeripherals_Power_adapters

[3.] **Software for the Raspberry Pi at NXP:**
http://www.nxp.com/demoboard/PNEV512R.html#documentation

[4.] **Software for the Raspberry Pi at element14**

www.nxp.com/redirect/element14.com/community/community/designcenter/explorenfc

[5.] **Supported wireless network adapters:**
http://www.nxp.com/redirect/elinux.org/RPi_VerifiedPeripherals_USB_Wi-Fi_Adapters

[6.] **Downloads for the Raspberry Pi**
http://www.nxp.com/redirect/raspberrypi.org/downloads

[7.] **Raspbian Linux**
http://www.nxp.com/redirect/raspbian.org

[8.] **FAQ for the Raspberry Pi**
http://www.nxp.com/redirect/raspberrypi.org/faqs

[9.] **nxppy**
www.nxp.com/redirect/github.com/svvitale/nxppy

10. Legal information

10.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

10.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

10.3 Licenses

Purchase of NXP ICs with NFC technology
Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

10.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

MIFARE — is a trademark of NXP Semiconductors N.V.

MIFARE Plus — is a trademark of NXP Semiconductors N.V.

MIFARE Ultralight — is a trademark of NXP Semiconductors N.V.

DESFire — is a trademark of NXP Semiconductors N.V.

11. List of figures

12. Contents

For more information, visit: http://www.nxp.com

Date of release: 13 July 2016
279725
Document identifier: AN11480