UM2375
User manual

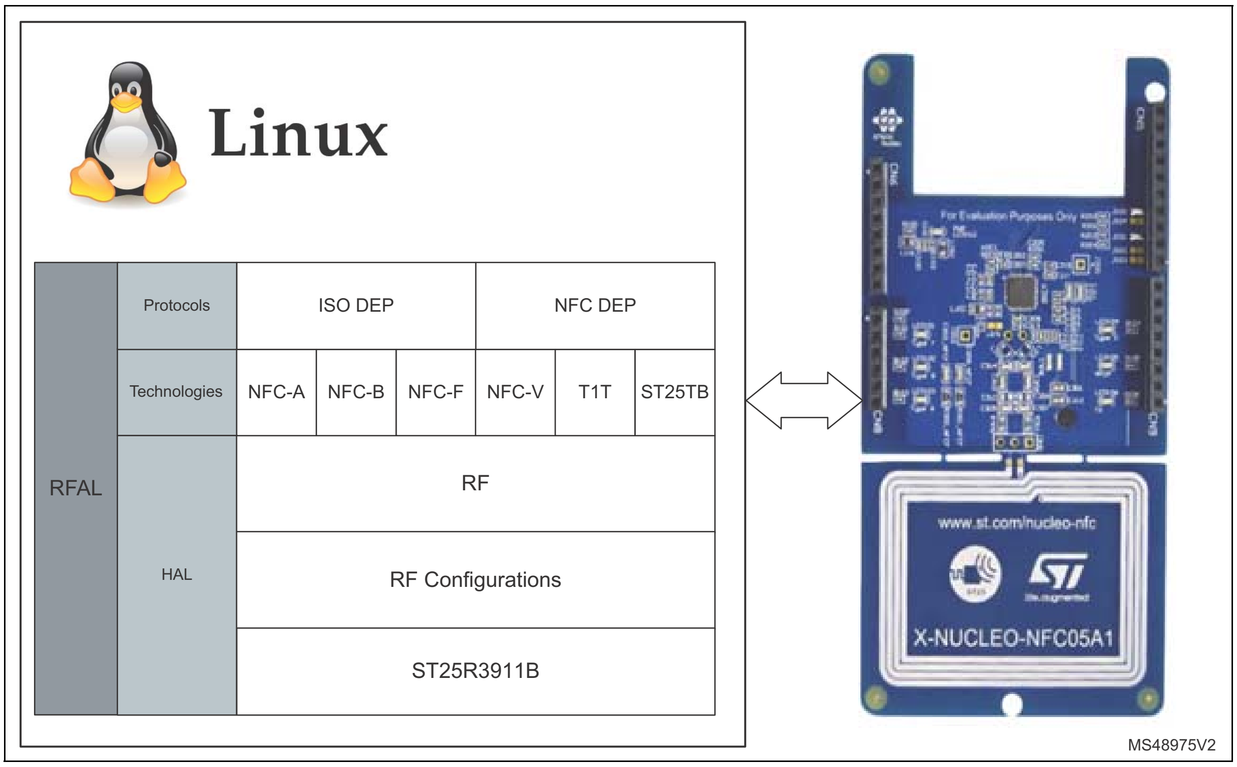
Linux® driver for the ST25R3911B/ST25R391x high performance NFC frontends

Introduction

The STSW-ST25R009 Linux® driver enables the Raspberry Pi 3 to operate with the X-NUCLEO-NFC05A1.
This package ports the RF abstraction layer (RFAL) onto a Raspberry Pi 3 Linux platform to operate with X-NUCLEO-NFC05A1 firmware, which contains the ST25R3911B high-performance NFC frontend. The package contains a sample application that detects different types of NFC tags and mobile phones supporting P2P. The RFAL is the ST standard driver for ST25R NFC/RFID Reader ICs ST25R3911B, ST25R3912, ST25R3913, ST25R3914 and ST25R3915. It is used, for instance, by the ST25R3911B-DISCO (STSW-ST25R002) and X-NUCLEO-NFC05A1 (X-CUBE-NFC5) firmware.

STSW-ST25R009 supports all the ST25R3911B lower-layer protocols and also some higher layer protocols to abstract RF communication. As the RFAL is written in a portable manner, it can run on a wide range of devices, from 8-bit MCUs up to 64-bit processors running Linux.
This document describes how the RFAL library can be used on a standard Linux system (in this case the Raspberry Pi 3) for NFC/RF communication. All the code here is highly portable and works with minor changes on any Linux platform.

Figure 1. RFAL library on Linux platform



Contents

List of figures

List of tables

1 Overview

1.1 Features

- Complete Linux user space driver (RF abstraction layer) to build NFC enabled applications using the ST25R3911B/ST25R391x high performance NFC frontends with up to 1.4 W output power
- Linux host communication with the ST25R3911B/ST25R391x high performance NFC frontends using SPI interface
- Complete RF/NFC abstraction (RFAL) for all major technologies and higher layer protocols:
  - NFC-A (ISO14443-A)
  - NFC-B (ISO14443-B)
  - NFC-F (FeliCa)
  - NFC-V (ISO15693)
  - P2P (ISO18092)
  - ISO-DEP (ISO data exchange protocol, ISO14443-4)
  - NFC-DEP (NFC data exchange protocol, ISO18092)
  - Proprietary technologies (Kovio, B’, iClass, Calypso, …)
- Sample implementation available with the X-NUCLEO-NFC05A1 expansion board, plugged into a Raspberry PI 3
- Sample application to detect several NFC tag types and mobile phones supporting P2P
- Free user-friendly license terms

1.2 Software architecture

Figure 2 shows the software architecture details of RFAL library on a Linux platform.

The RFAL is easily portable to other platforms by adapting the so-called platform files:

The header file platform.h contains macro definitions, which are expected to be provided and implemented by the platform owner.

It provides platform specific settings like GPIO assignment, system resources, locks and IRQs, which are required for correct operation of the RFAL.

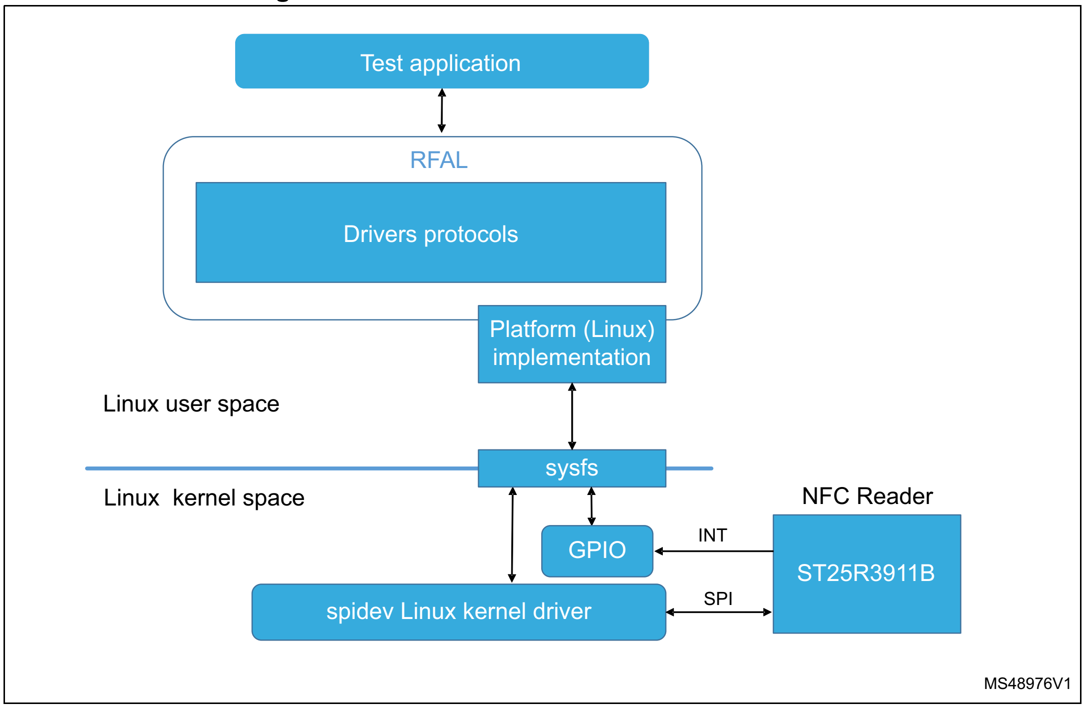
This demo implements the platform functions and provides a port of the RFAL into user space of Linux. A shared library file is generated, which is used by a test/demo application to showcase the functionalities provided by the RFAL layer.

Linux host uses sysfs interface available from Linux user space for performing SPI communication with the ST25R3911B device.

Inside the Linux kernel the SPI sysfs interface uses Linux kernel driver spidev to send/receive the SPI frames to ST25R3911B.

For handling the INT line of ST25R3911B the driver uses the GPIO sysfs to get notified of changes on this line.

Figure 2. RFAL software architecture on Linux



2 Hardware setup

2.1 Platform used

A Raspberry Pi 3 board with Raspbian OS (operating system) is used as Linux platform to build RFAL library and interact with ST25R3911B over SPI.

ST25R3911B enables an application on Linux platform to detect and communicate with NFC devices.

2.2 Hardware requirements

- Raspberry Pi 3 Model B V1.2
- X-NUCLEO-NFC05A1
- 8 GB micro SD card to boot Raspberry Pi
- Bridge board to connect X-NUCLEO NFC05A1 with Raspberry Pi Arduino Adapter for Raspberry Pi, Part No. ARPI600
- SD card reader

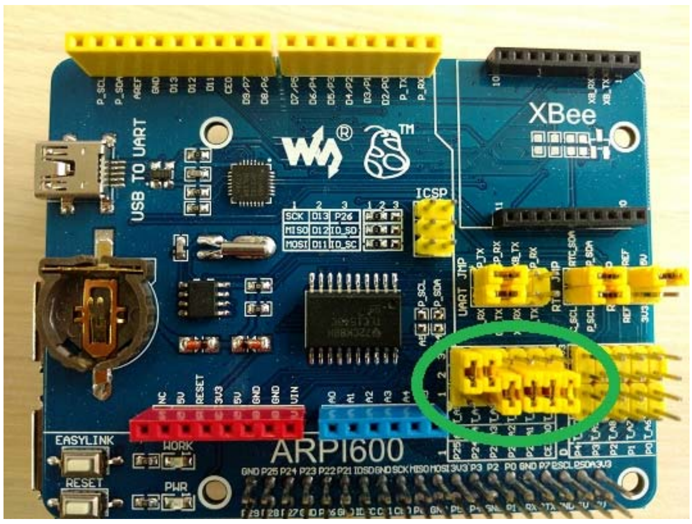
2.2.1 Hardware connections

The ARPI600 Raspberry Pi to Arduino adapter board is used to connect X-NUCLEO-NFC05A1 with the Raspberry Pi. It is required to modify the jumpers of adapter board to connect it with the X-NUCLEO-NFC05A1.

Jumper setting

The jumpers for A3, A2, A1 and A0 shown in *Figure 3* have to be changed to P23, P22, P21 and CE1. With these jumper settings, Raspberry's GPIO pin number 7 is used as interrupt line for X-NUCLEO-NFC05A1.

Figure 3. Position of Jumpers A3, A2, A1 and A0 on adapter board



Currently, this RFAL library port uses the pin GPIO7 as the interrupt line (according to the jumper setting). If there is a requirement to change the interrupt line from GPIO7 to a different GPIO, the platform specific code (in file pltf_gpio.h) needs to be modified to change the definition of macro "PLTF_GPIO_INTR_PIN" from 7 to the new GPIO pin, which should to be used as interrupt line.

With the above jumper settings, the adapter board can be used to connect X-NUCLEO NFC05A1 with Raspberry Pi board as shown in *Figure 4*.

Figure 4. Hardware setup top view

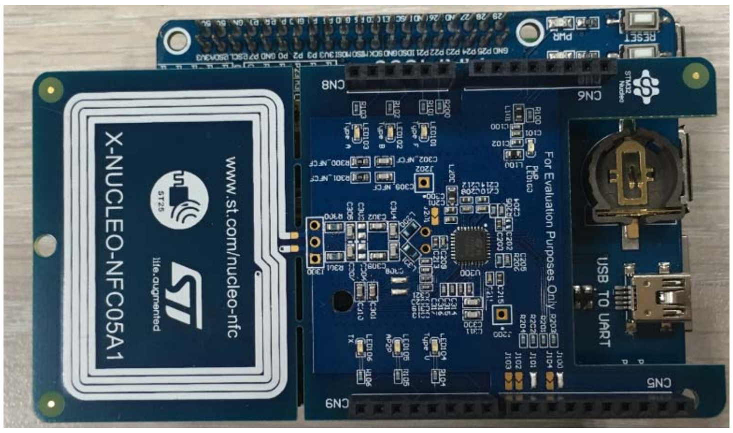
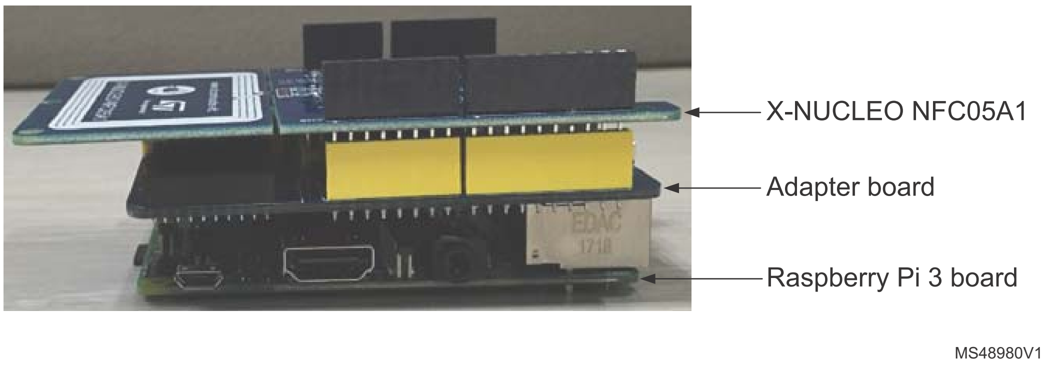


Figure 5. Hardware setup side view



3 Linux environment setup

3.1 Booting of Raspberry Pi

To setup the Linux environment, first step is to install and boot the Raspberry Pi 3 with Raspbian OS as explained below:

Step1.

Download the latest Raspbian image from the link:

- https://www.raspberrypi.org

Choose RASPBIAN STRETCH WITH DESKTOP. For the tests below the following version was used: November 2017 (2017-11-29 with kernel 4.9).

Step2.

Unzip the raspbian image and write in onto the SD card by following the instructions available in the section named “Writing an image to the SD card”.

Step3.

Connect the hardware:

- Connect the Raspberry Pi 3 to a monitor using a standard HDMI cable.
- Connect mouse and keyboard to Raspberry Pi’s USB ports.

It is also possible to work with Raspberry Pi using ssh. In that case it is no required to connect the monitor, keyboard and mouse with Raspberry Pi. The only requirement is to have the PC with ssh inside the same network as the Raspberry Pi.

Step4.

Boot the Raspberry Pi 3 with SD card.

After booting, a debian based Linux desktop appears on monitor.

Note: *Sometimes, it is observed that after booting Raspberry Pi with raspbian, some keyboard keys do not work. To make them work, open the file /etc/default/keyboard and set XKBLAYOUT=”us” and reboot the Raspberry Pi.*

3.2 Enable SPI on Raspberry Pi

The SPI driver inside the kernel communicates with X-NUCLEO-NFC05A1 over SPI. It is important to check if SPI is already enabled in the raspbian/kernel configuration.

Check if /dev/spidev0.0 is visible in the raspberry Pi environment. If it is not visible, enable the SPI interface using the utility “raspi-config” by following the steps described below.

Step1.

Open a new terminal on the Raspberry Pi and run the command “raspi-config” as root:

```
sudo raspi-config.
```

This step opens a graphical interface.

Step2.

Select in the graphical interface the option named “Interfacing Options”.

Step3.

This step lists various options.

Select the option named “SPI”.

A new window appears with following text:

“Would you like the SPI interface to be enabled?”

Step4.

Select <Yes> in this window to enable SPI.

Step5.

Reboot Raspberry Pi.

The above steps will enable the SPI interface in Raspberry Pi environment after a reboot.

4 Build of RFAL library and application

The RFAL demo of Linux is provided in an archive. Let's assume its name is: rfal_v1.3.0_nfc_poller.tar.xz.

To build the RFAL library and application on raspberry Pi, follow the below steps:

Step1.

Unzip the package rfal_v1.3.0_nfc_poller.tar.xz on raspberry Pi using the below command from the home directory:

```
tar -xJvf rfal_v1.3.0_nfc_poller.tar.xz
```

Step2.

Install cmake (if not done before) using below command:

```
apt-get install cmake
```

RFAL library and application build system is based on cmake, for this reason it is required to install cmake for compilation of the package.

Step3.

To build the RFAL library and application, go to the “build” directory:

```
cd ~/rfal_v1.3.0/build
```

and run the below command from there:

```
cmake ..
```

In above command “..” indicates that top level CMakeLists.txt exists in the parent directory (i.e. rfal_v1.3.0).

cmake command creates the makefile that is used in the next step to build the library and application.

Step4.

Run the make command to build the RFAL library and application:

```
make
```

The make command first builds the RFAL library and then builds the application on top of it.

5 How to run the application

Successful build will generate an executable named “nfcPoller” at the following location:

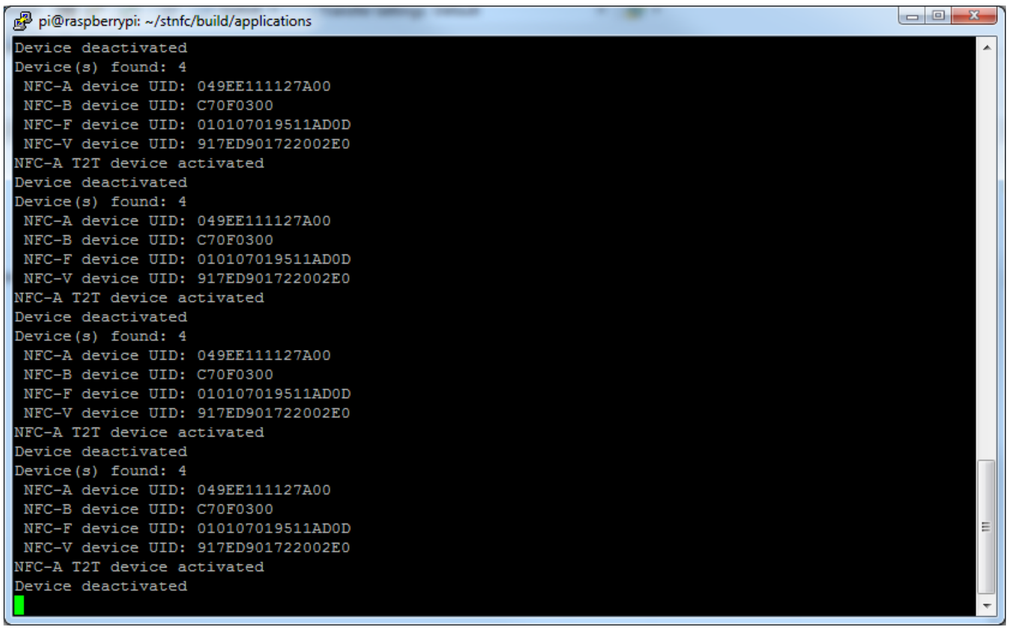
/build/applications.

By default the application needs to be run with root rights from the path: ~/rfal_v1.3.0//build/applications/:

```
sudo ./nfcPoller
```

The application starts to poll for NFC tags and mobile phones. It will display the found devices with their UID as shown in *Figure 6*.

Figure 6. Displays of found devices



pi@raspberrypi: ~/stnfc/build/applications

```
Device deactivated
Device(s) found: 4
 NFC-A device UID: 049EE111127A00
 NFC-B device UID: C70F0300
 NFC-F device UID: 010107019511AD0D
 NFC-V device UID: 917ED901722002E0
NFC-A T2T device activated
Device deactivated
Device(s) found: 4
 NFC-A device UID: 049EE111127A00
 NFC-B device UID: C70F0300
 NFC-F device UID: 010107019511AD0D
 NFC-V device UID: 917ED901722002E0
NFC-A T2T device activated
Device deactivated
Device(s) found: 4
 NFC-A device UID: 049EE111127A00
 NFC-B device UID: C70F0300
 NFC-F device UID: 010107019511AD0D
 NFC-V device UID: 917ED901722002E0
NFC-A T2T device activated
Device deactivated
Device(s) found: 4
 NFC-A device UID: 049EE111127A00
 NFC-B device UID: C70F0300
 NFC-F device UID: 010107019511AD0D
 NFC-V device UID: 917ED901722002E0
NFC-A T2T device activated
Device deactivated
```

To terminate the application press Ctrl + c.

6 Revision history

Table 1. Document revision history

Date	Revision	Changes
09-Mar-2018	1	Initial release.