

# Getting started with the Linux® demo application for ST25R100 and ST25R200

## Introduction

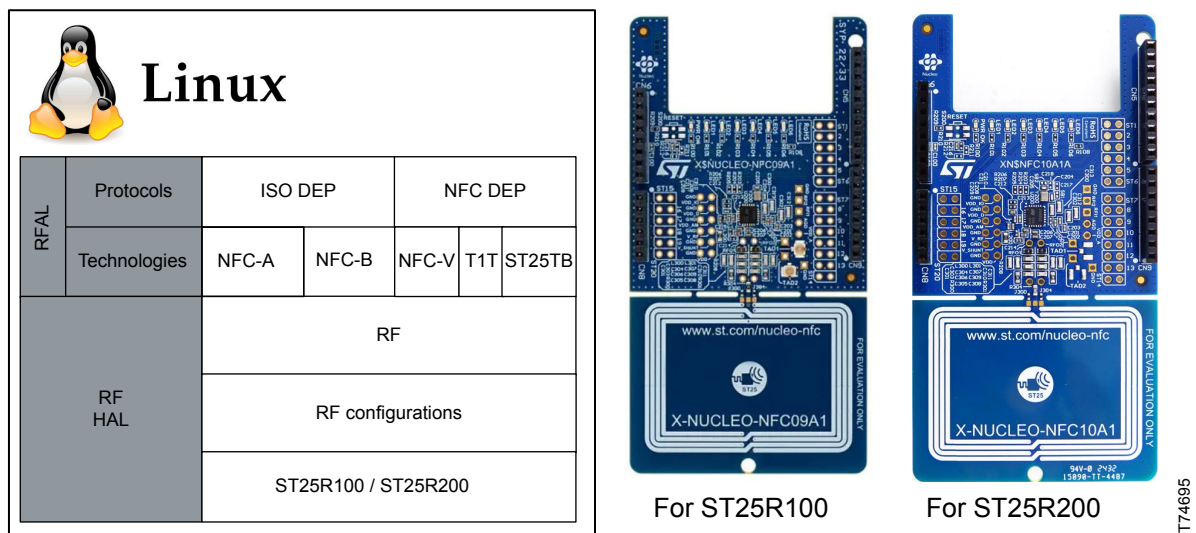
This document explains how to use the RFAL library on a standard Linux system, such as the Raspberry Pi 4, for NFC and RF communication. The code is highly portable and requires only minor modifications to run on any Linux platform.

STSW-ST25R021 Linux® driver enables the Raspberry Pi® 4 to operate with the X-NUCLEO-NFC09A1 board, which contains the ST25R100 device or with the X-NUCLEO-NFC10A1, which contains the ST25R200.

This package ports the RF abstraction layer (RFAL) onto a Raspberry Pi 4 Linux platform, to operate with the board firmware, and provides a sample application detecting different types of NFC tags. The RFAL is the ST standard driver for ST25R100 and ST25R200, multipurpose NFC transceivers. It is used, for instance, by the STEVAL-25R200A firmware.

STSW-ST25R021 supports all the ST25R100 and ST25R200 lower-layer and some higher-layer protocols for communication. The RFAL is written in a portable manner, enabling it to run on a wide range of devices based on Linux.

**Figure 1. RFAL library on Linux platform**



# 1 Overview

## 1.1 Features

- Complete Linux user-space driver (RF abstraction layer) for building NFC-enabled applications using the ST25R100 and ST25R200 device
- Linux host communication with the readers through an SPI interface
- Complete RF/NFC abstraction (RFAL) for all major technologies and higher layer protocols:
  - NFC-A (ISO14443-A)
  - NFC-B (ISO14443-B)
  - NFC-V (ISO15693)
  - ISO-DEP (ISO data exchange protocol, ISO14443-4)
  - Proprietary technologies (Kovio, B', iClass, Calypso®)
- Sample implementation available with the X-NUCLEO-NFC09A1 and X-NUCLEO-NFC10A1 expansion boards, plugged into a Raspberry Pi 4
- Sample application to detect several NFC tag types and mobile phones
- 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.

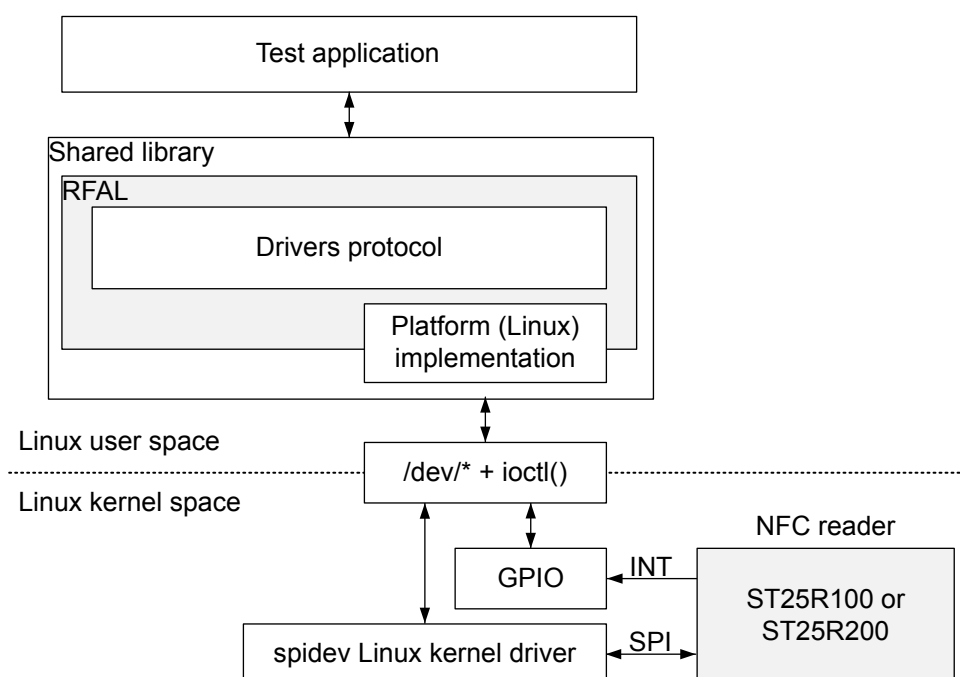
The header file *rfal\_platform.h* contains macro definitions, to be provided and implemented by the platform owner. It provides platform specific settings like GPIO assignment, system resources, locks, and IRQs, which are required for correct operation of the RFAL.

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

The Linux host uses the SPI character device interface, *spidev*, available from the Linux user space to perform SPI communication with the device.

Inside the Linux kernel, the *spidev* driver handles sending and receiving SPI frames to and from the devices.

**Figure 2. Software architecture on Linux**



DT79238V1

## 2 Hardware setup

### 2.1 Platform used

A Raspberry Pi 4 board with Raspberry Pi OS is used as a Linux platform to build the RFAL library and interact with the ST25R100 and ST25R200 over SPI.

The devices enable an application on the Linux platform to detect and communicate with NFC devices.

### 2.2 Hardware requirements

- Raspberry Pi 4
- 8 GB microSD™ card to boot Raspberry Pi OS (with its latest requirements)
- SD card reader
- X-NUCLEO-NFC09A1 or X-NUCLEO-NFC10A1 board
- Bridge to connect the board with Raspberry Pi Arduino® adapter for Raspberry Pi (part number ARPI600)

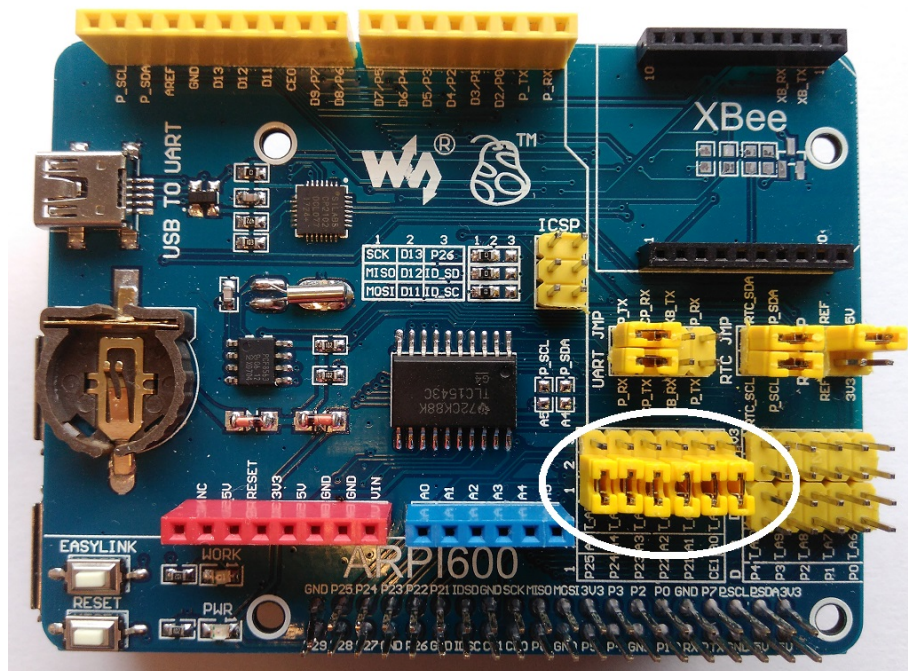
#### 2.2.1 Hardware connections

The ARPI600 Raspberry Pi to Arduino® adapter is used to connect the boards with the Raspberry Pi. The jumpers of the adapter board must be modified to connect it with the X-NUCLEO-NFC09A1 board or X-NUCLEO-NFC10A1 board.

##### Jumper setting

The jumpers for A5, A4, A3, A2, A1, and A0 shown in Figure 2 must be changed, respectively, to P25, P24, P23, P22, P21, and CE1. With this setting, Raspberry GPIO pin number 7 is used as the interrupt line for X-NUCLEO-NFC09A1 and X-NUCLEO-NFC10A1.

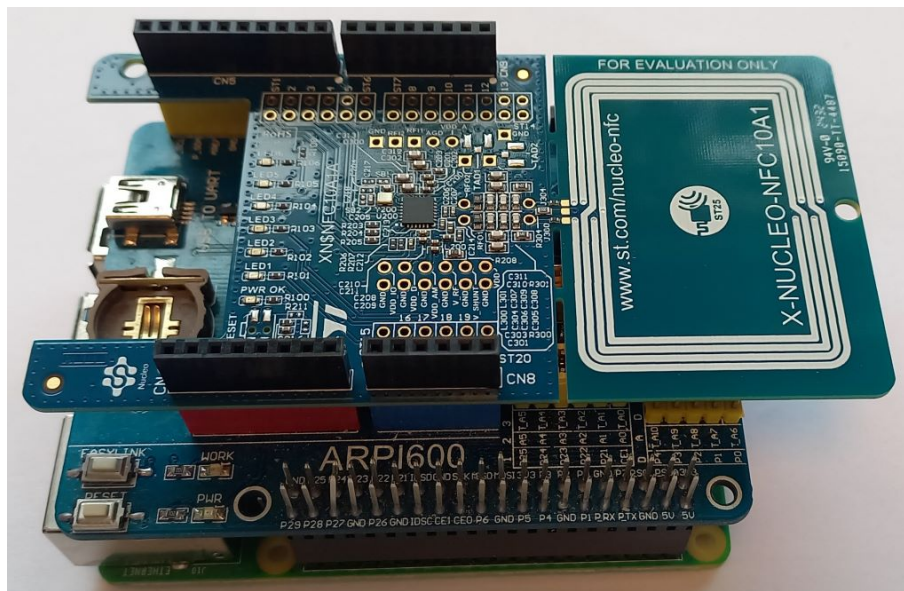
**Figure 3. Position of jumpers A5, A4, A3, A2, A1, and A0 on the adapter board**



Currently, this RFAL library demo uses the pin GPIO7 as the interrupt line (according to the jumper settings). 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*) must be modified to change the definition of macro `ST25R_INT_PIN` from 7 to the new GPIO pin, to be used as the interrupt line.

With the above jumper settings, the adapter board can be used to connect the X-NUCLEO-NFC09A1 and X-NUCLEO-NFC10A1 with Raspberry Pi board, as shown in the following figures.

**Figure 4. Hardware setup side view (X-NUCLEO-NFC10A1)**



## 3 Linux environment setup

### 3.1 Booting Raspberry Pi

To set up the Linux environment, first install and boot the Raspberry Pi with Raspberry Pi OS by following the steps outlined below:

#### Step 1

Download the latest Raspberry Pi OS image from the official website, then select **Raspberry Pi OS Desktop**. For the tests described below, the version **2024-10-22-raspios-bookworm-arm64-full.img.xz** (October 2024) was used.

#### Step 2

Unzip the Raspberry Pi OS image and write it onto the SD card by following the instructions available in the section named **Writing an image to the SD card**.

#### Step 3

Connect the hardware:

- Connect the Raspberry Pi 4 to a monitor using a standard HDMI cable.
- Connect mouse and keyboard to Raspberry Pi USB ports.

It is also possible to work with the Raspberry Pi using SSH. In this case, connecting a monitor, keyboard, and mouse to the Raspberry Pi is not required. The only requirement is that the PC running SSH is on the same network as the Raspberry Pi, with the IP address configured accordingly.

#### Step 4

Boot the Raspberry Pi 4 with the SD card. After booting, a Debian-based Linux desktop appears on the monitor.

### 3.2 Enable SPI on a Raspberry Pi

The SPI driver within the kernel communicates with the X-NUCLEO-NFC09A1 and X-NUCLEO-NFC10A1 boards via SPI. It is important to verify whether SPI is already enabled in the Raspberry Pi OS kernel configuration.

To check, confirm if `/dev/spidev0.0` is present in the Raspberry Pi environment. If it is not visible, enable the SPI interface using the **raspi-config** utility by following the steps described below.

#### Step 1

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.

#### Step 2

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

#### Step 3

Select the option named “SPI”. A new window appears asking: “**Would you like the SPI interface to be enabled?**”.

#### Step 4

Select **<Yes>** in this window to enable SPI.

#### Step 5

Reboot the Raspberry Pi. The SPI interface is enabled in the Raspberry Pi environment after the reboot.

## 4 Build RFAL library and application

The RFAL demonstration for Linux is provided as an archive file, for example, **en.STSW-ST25R021.xz**. To build the RFAL library and application on the Raspberry Pi, follow the steps below:

### Step 1

Unzip the package on the Raspberry Pi using the following command from the home directory

```
tar -xJvf en.STSW-ST25R021.xz
```

This results in the following directory structure on disk:

- common/: Common helper files
- linux\_demo/: Application and platform-specific customization files
- rfal/: RFAL source code
- st25r\_demos/: Poller demo source code
- readme.txt
- readme\_cross\_compilation.txt
- SLA0051\_my\_liberty.pdf

### Step 2

Install CMake (if not done before) using the command

```
apt-get install cmake
```

RFAL library and application build system are based on CMake, for this reason it is required to install CMake to compile the package.

### Step 3

To build the RFAL library and application, go to the build directory

```
cd ST25R200_RFAL_v3.0.0_Linux_demo_v1.0
```

From there, run the command

```
cmake ..
```

In the command above, `..` indicates that the top-level `CMakeLists.txt` file is located in the parent directory. This command generates the makefile, which is used in the following step to build the library and application.

### Step 4

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

```
make
```

This command first builds the RFAL library, and then the application on top of it.



## 5 How to run the application

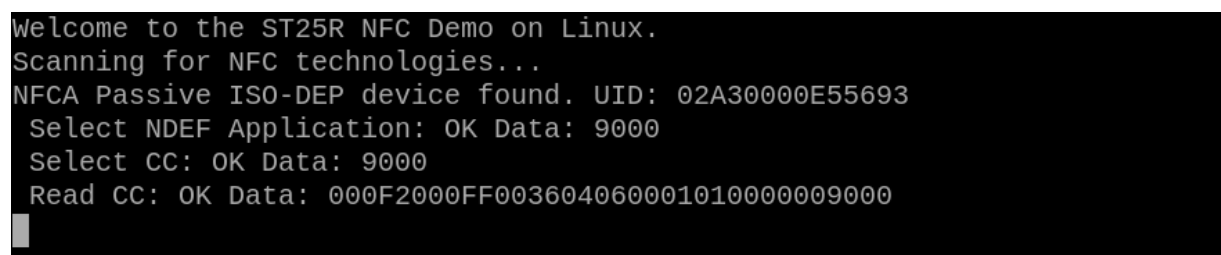
A successful build generates an executable named `nfc_demo_st25r200` located in the `/build/demo` directory.

By default, the application must be run with root privileges from this path:

```
sudo ./nfc_demo_st25r200
```

The application starts to poll for NFC tags and mobile phones, then displays the found devices with their UID, as shown in [Figure 5](#).

**Figure 5. Display of found devices**



```
Welcome to the ST25R NFC Demo on Linux.  
Scanning for NFC technologies...  
NFCA Passive ISO-DEP device found. UID: 02A30000E55693  
Select NDEF Application: OK Data: 9000  
Select CC: OK Data: 9000  
Read CC: OK Data: 000F2000FF003604060001010000009000
```

To terminate the application press Ctrl + C.

## Revision history

**Table 1. Document revision history**

Date	Revision	Changes
24-Sep-2025	1	Initial release.



## Contents

<b>1</b>	<b>Overview .....</b>	<b>2</b>
1.1	Features .....	2
1.2	Software architecture .....	2
<b>2</b>	<b>Hardware setup.....</b>	<b>3</b>
2.1	Platform used.....	3
2.2	Hardware requirements .....	3
2.2.1	Hardware connections .....	3
<b>3</b>	<b>Linux environment setup .....</b>	<b>5</b>
3.1	Bootting Raspberry Pi .....	5
3.2	Enable SPI on a Raspberry Pi.....	5
<b>4</b>	<b>Build RFAL library and application .....</b>	<b>6</b>
<b>5</b>	<b>How to run the application .....</b>	<b>7</b>
	<b>Revision history .....</b>	<b>8</b>
	<b>List of tables .....</b>	<b>10</b>
	<b>List of figures.....</b>	<b>11</b>



## List of tables

Table 1.	Document revision history . . . . .	8
----------	-------------------------------------	---

## List of figures

<b>Figure 1.</b>	RFAL library on Linux platform . . . . .	1
<b>Figure 2.</b>	Software architecture on Linux . . . . .	2
<b>Figure 3.</b>	Position of jumpers A5, A4, A3, A2, A1, and A0 on the adapter board . . . . .	3
<b>Figure 4.</b>	Hardware setup side view (X-NUCLEO-NFC10A1) . . . . .	4
<b>Figure 5.</b>	Display of found devices . . . . .	7

**IMPORTANT NOTICE – READ CAREFULLY**

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved