UM2008
User manual

Getting started with the X-CUBE-NFC2 dynamic NFC/RFID tag IC software expansion for STM32Cube

Introduction

This document describes how to start with the X-CUBE-NFC2 software expansion for STM32Cube.

X-CUBE-NFC2 provides a complete middleware solution to build applications using the dynamic NFC/RFID tag IC M24LR04E-R in combination with STM32 microcontrollers. It is easily portable across different MCU series, thanks to STM32Cube.

The software is based on STM32Cube and expands STM32Cube based packages.

This package contains a sample application example to write the NDEF (NFC Data Exchange Format) file with URI (Uniform Resource Identifier) data. The software provides implementation examples for STM32 Nucleo platforms equipped with the X-NUCLEO-NFC02A1 expansion board, featuring the dynamic NFC/RFID tag IC M24LR04E-R.

Contents

List of tables

List of figures

1 List of acronyms

Table 1. List of acronyms

Acronym	Description
NFC	Near Field Communication
NDEF	NFC Data Exchange Format
RFID	Radio Frequency IDentification
URI	Uniform Resource Identifier
AAR	Android Application Record
SMS	Short Message Service
MCU	Micro Controller Unit
BSP	Boot Support Package
HAL	Hardware Abstraction Layer
LED	Light Emitting Diode
CMSIS	ARM® Cortex® Microcontroller Software Interface Standard
IDE	Integrated Development Environment

2 References

- M24LR04E-R datasheet (available on *www.st.com*)
- STM32 microcontrollers datasheets (available on *www.st.com*)
- Nucleo boards user manuals (available on *www.st.com*)

3 What is STM32Cube?

3.1 STM32Cube overview

STMCube™ initiative was originated by STMicroelectronics to ease developers' life by reducing development efforts, time and cost. STM32Cube covers STM32 portfolio.

STM32Cube Version 1.x includes:

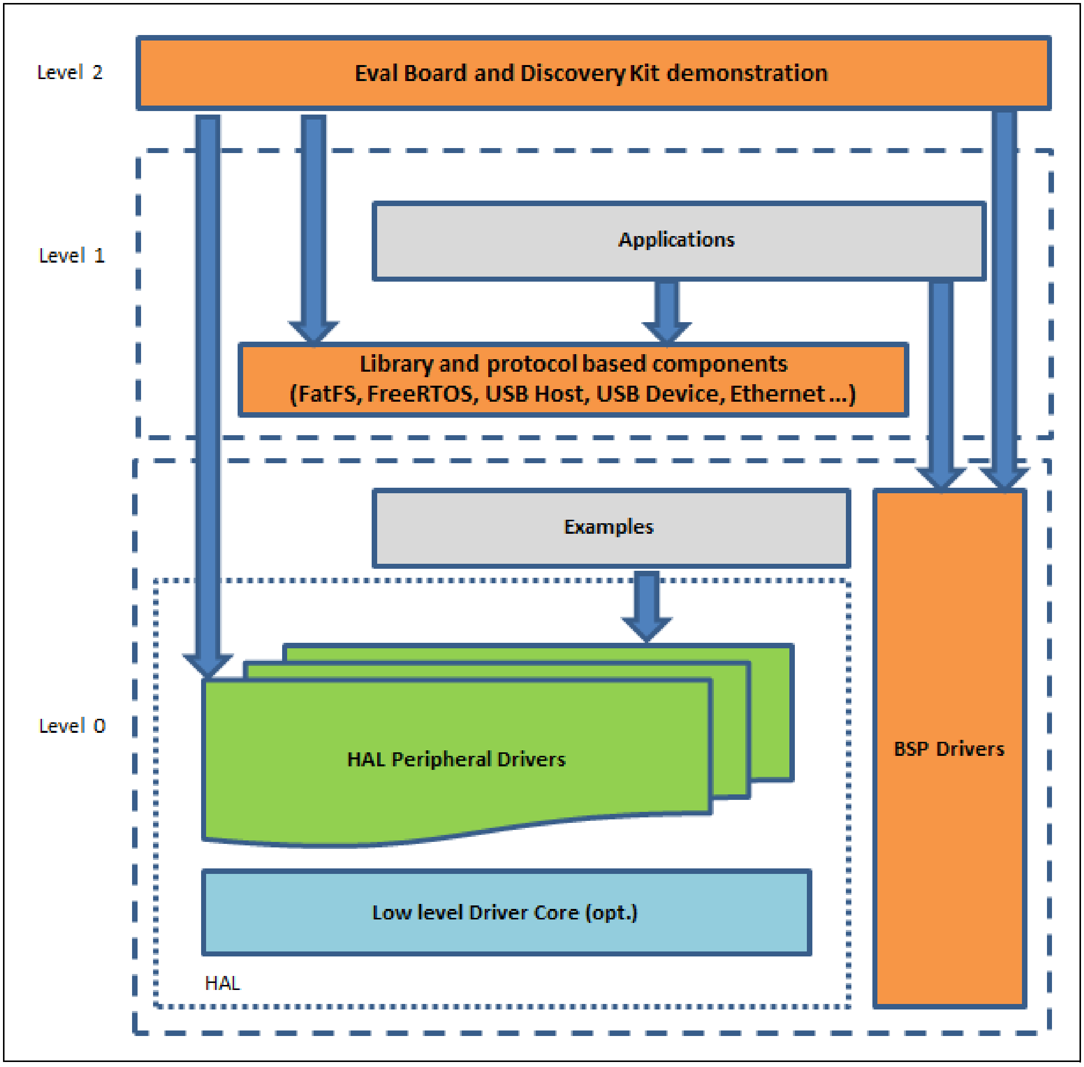
- The STM32CubeMX, a graphical software configuration tool that allows to generate C initialization code using graphical wizards
- A comprehensive embedded software platform, delivered per series (such as STM32CubeF4 for STM32F4 series)
  - The STM32Cube HAL, an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio
  - A consistent set of middleware components such as RTOS, USB, TCP/IP, Graphics
  - All embedded software utilities coming with a full set of examples

Detailed information about STM32Cube is available on *www.st.com.*

3.2 STM32Cube architecture

The STM32Cube firmware solution is built around three independent levels that can easily interact with each other, as described in *Figure 1*.

Figure 1. Firmware architecture


Level 2
Eval Board and Discovery Kit demonstration
Level 1
Applications
Library and protocol based components
(FatFS, FreeRTOS, USB Host, USB Device, Ethernet...)
Examples
Level 0
HAL Peripheral Drivers
BSP Drivers
Low level Driver Core (opt.)
HAL

Level 0: this level is divided into three sub-layers

- Board Support Package (BSP): this layer offers a set of APIs relative to the hardware components of the boards (such as Audio codec, IO expander, Touchscreen, SRAM driver, LCD drivers) and composed of two parts:
  - Component: is the driver relative to the external device on the board, not related to the STM32. The component driver provides specific APIs to the BSP driver external components and can be ported on any other board.
  - BSP driver: needed to link the component driver to a specific board, it provides a set of friendly used APIs. The APIs naming rule is *BSP_FUNCT_Action()*, as in *BSP_LED_Init()*, *BSP_LED_On()*

  BSP is based on modular architecture, so that it can be ported easily on any hardware by just implementing the low level routines.
- Hardware Abstraction Layer (HAL): this layer provides the low level drivers and the hardware interfacing methods to interact with the upper layers (application, libraries and stacks). HAL provides generic, multi instance and functionality-oriented APIs

needed to offload the user application implementation by providing ready-to-use processes. As an example, for the communication peripherals (I2S, UART…) it provides APIs to initialize and configure the peripheral, manages data transfer based on polling, interrupt or DMA processes, and manages communication errors that may arise during communication.

The HAL Drivers APIs are split in two categories:

- generic APIs (providing common and generic functions to all the STM32 MCUs)
- extension APIs (customized functions for a specific family or a specific part number).

- Basic peripheral usage examples: this layer encloses the examples built on the STM32 peripheral using only the HAL and BSP resources.

Level 1: This level is divided in two sub-layers:

- Middleware components: set of Libraries covering USB Host and Device Libraries, STemWin, FreeRTOS, FatFS, LwIP, and PolarSSL. Horizontal interactions between the components of this layer is done directly by calling the feature APIs, while the vertical interaction with the low level drivers is carried out through specific callbacks and static macros implemented in the library system call interface. For example, the FatFS implements the disk I/O driver to access microSD drive or the USB Mass Storage Class.
- Examples based on the Middleware components: each Middleware component comes with one or more examples (called also Applications) showing how to use it. Integration examples that use several Middleware components are provided as well.

Level 2: This level is composed of a single layer: global real-time and graphical demonstration based on the Middleware service layer, the low level abstraction layer, and the basic peripheral usage applications for board based functionalities.

4 X-CUBE-NFC2 software expansion for STM32Cube

4.1 Overview

X-CUBE-NFC2 is a software package that expands the functionality provided by STM32Cube.

The key features of the package are:

- Complete middleware to build applications using dynamic NFC/RFID tag IC (M24LR04E-R)
- Easy portability across different MCU families, thanks to STM32Cube
- Sample application example to write the NDEF (NFC Data Exchange Format) file with URI (Uniform Resource Identifier) data
- Free and user-friendly license terms
- Examples implementation available on board X-NUCLEO-NFC02A1 plugged on top of either NUCLEO-F401RE or NUCLEO-L053R8

This software is gathering dynamic NFC/RFID tag IC drivers for the M24LR04E-R device running on STM32.

The software is built on top of STM32Cube software technology to ease portability across different STM32 microcontrollers, it comes with examples of implementation of such drivers, running on X-NUCLEO-NFC02A1 plugged on top of NUCLEO-F401RE or NUCLEO-L053R8 boards.

The package extends STM32Cube by providing a Board Support Package (BSP) for the X-NUCLEO-NFC02A1 expansion board and some middleware components for NDEF application drivers.

The drivers abstract low-level details of the hardware and allow middleware components and applications to access NDEF data in a hardware-independent fashion.

The package also includes a sample application example that the developer can use to start experimenting with the code. The sample example is developed to write the NDEF file with URI data. Other NDEF application drivers included in the middleware are:

- NDEF AAR (to add AAR (Android Application Record) in the tag)
- NDEF Email (to manage the NDEF file that represents Email)
- NDEF Geo (to manage the NDEF file that represents geo-location)
- NDEF MyApp (to manage the NDEF file of a private application)
- NDEF SMS (to manage the NDEF file that represents SMS)
- NDEF Text (to manage Text NDEF file)
- NDEF Vcard (to manage the NDEF file that represents Vcard)

4.2 Architecture

This software is an expansion for STM32Cube, it fully complies with the architecture of STM32Cube and expands it in order to enable development of applications using dynamic NFC/RFID tag IC.

Refer to *Section 3.2* for an introduction to the STM32Cube architecture.

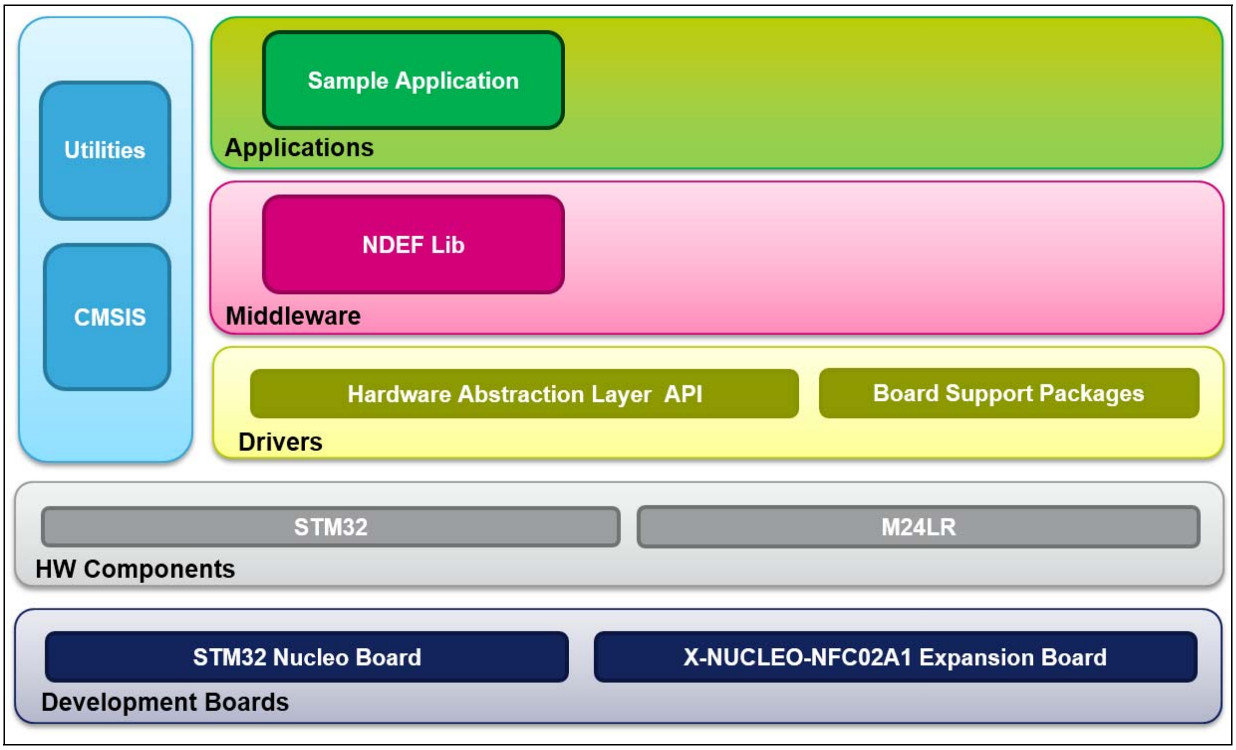
The software is based on the STM32CubeHAL, the hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a Board Support Package (BSP) for the X-NUCLEO-NFC02A1expansion board.

The software layers used by the application software to access and use the X-NUCLEO-NFC02A1 expansion board are:

- STM32Cube HAL Layer: The HAL driver layer provides a generic multi instance simple set of APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). It is composed of generic and extension APIs. It is directly built around a generic architecture and allows the layers that are built upon it (such as the middleware layer) to implement their functionalities without depending from the specific hardware configuration for a given MCU. This structure improves the library code reusability and guarantees an easy portability on other devices.
- Board Support Package (BSP) Layer: The software package needs to support the peripherals on the STM32 Nucleo board apart from the MCU. This software is included in the board support package (BSP). This is a limited set of APIs which provides a programming interface for certain board specific peripherals, e.g. the LED, the user button etc. This interface also helps in identifying the specific board version.

Figure 2 outlines the architecture of the package.

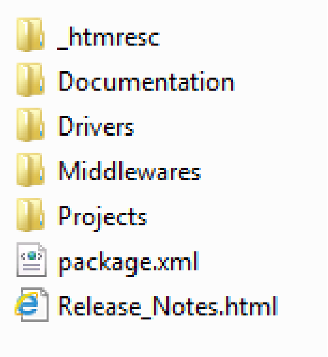
Figure 2. X-CUBE-NFC2 software architecture



4.3 Folders structure

This section provides an overview of the package folders structure, shown in *Figure 3*.

Figure 3. X-CUBE-NFC2 package folders structure



_htmresc
Documentation
Drivers
Middlewares
Projects
package.xml
Release_Notes.html

The following folders are included in the software package:

- **Documentation**: this folder contains a compiled HTML file generated from the source code, and documentation with details on the software components and on the APIs.
- **Drivers**: this folder contains the HAL drivers, the board specific drivers for each supported board or hardware platform, including the on-board components and the CMSIS layer which is a vendor-independent hardware abstraction layer for the Cortex®-M processor series.
- **Middlewares**: this folder contains NDEF application drivers and protocols related to the communication of NFC data.
  - NDEF AAR (to add AAR, Android Application Record, in the tag)
  - NDEF Email (to manage the NDEF file that represents Email)
  - NDEF Geo (to manage the NDEF file that represents geo-location)
  - NDEF MyApp (to manage the NDEF file of a private application)
  - NDEF SMS (to manage the NDEF file that represents SMS)
  - NDEF Text (to manage Text NDEF file)
  - NDEF URI (to manage URI NDEF file)
  - NDEF Vcard (to manage Vcard NDEF file)
- **Projects**: this folder contains a sample application, such as NDEF URI Write, provided for the NUCLEO-L053R8 and NUCLEO-F401RE platforms with three development environments, IAR™ Embedded Workbench® for ARM®, RealView® Microcontroller Development Kit (MDK-ARM™), and System Workbench for STM32 (SW4STM32).

4.4 APIs

Detailed technical information about the available APIs can be found in a compiled HTML file in the *Documentation* folder of the software package, where all the functions and parameters are described in detail.

4.5 Sample application description

An example application using the X-NUCLEO-NFC02A1 expansion board with either NUCLEO-F401RE or NUCLEO-L053R8 boards is provided in the *Projects* directory. Ready to be built projects are available for multiple IDEs.

In this application, NDEF (NFC Data Exchange Format) file with URI (Uniform Resource Identifier) data is written on NFC device M24LR04E-R

- Initialization: *BSP_NFCTAG_isInitialized()*
- Energy harvesting: *Enable_EnergyHarvesting()*
- NDEF URI writing: *NDEF_WriteURI()*
- NFC data writing: *BSP_NFCTAG_WriteData()*

After system initialization and clock configuration, three LEDs (green, blue and orange) are turned ON and OFF sequentially. Then, after initialization of NFC tag, energy harvesting is enabled. On successful enabling of energy harvesting, blue LED is turned ON. Eventually, on successful URI writing on NFC tag, orange LED is turned ON.

5 System setup guide

5.1 Hardware description

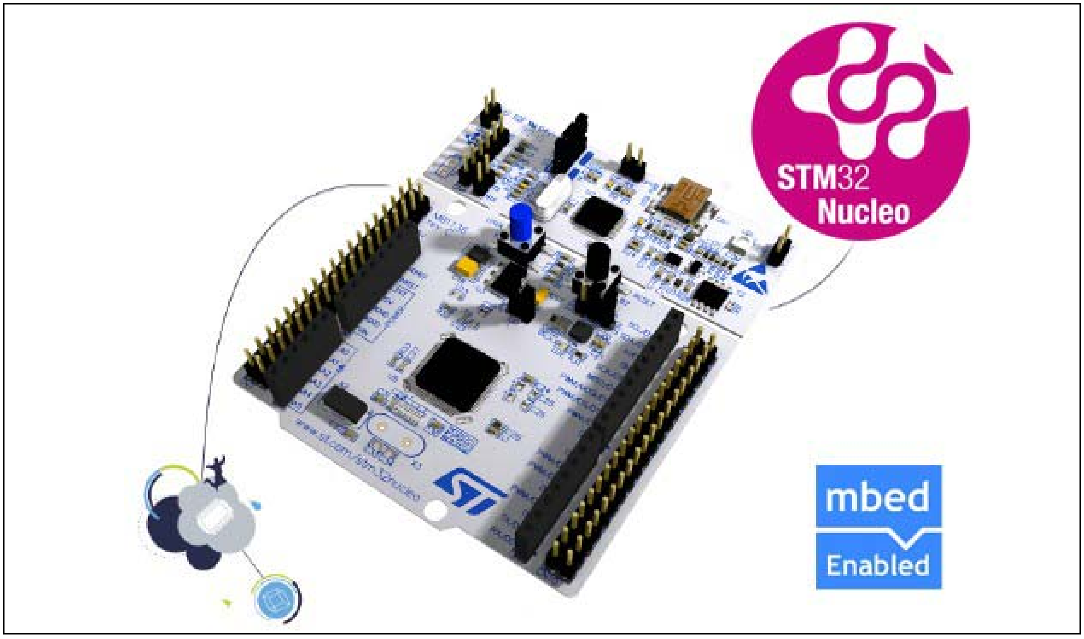
This section describes the hardware components required for developing a sensor-based application. The following subsections describe the individual components.

5.1.1 STM32 Nucleo platform

The STM32 Nucleo boards provide an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller lines. The Arduino™ connectivity support and ST morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide choice of specialized expansion boards. The STM32 Nucleo board does not require any separate probe, as it integrates the ST-LINK/V2-1 debugger/programmer. The STM32 Nucleo board comes with the STM32 comprehensive software HAL library together with various packaged software examples.

Information about the STM32 Nucleo boards is available on STMicroelectronics website *www.st.com*.

Figure 4. STM32 Nucleo board



5.1.2 X-NUCLEO-NFC02A1 expansion board

The X-NUCLEO-NFC02A1 is an NFC/RFID tag IC expansion board usable with the STM32 Nucleo platform. It is also compatible with Arduino™ UNO R3 connector layout, and is designed around the STMicroelectronics dynamic NFC/RFID tag IC (M24LR04E-R). The X-NUCLEO-NFC02A1 interfaces with the STM32 MCU through the I2C pin.

Figure 5. X-NUCLEO-NFC02A1 NFC/RFID tag IC expansion board

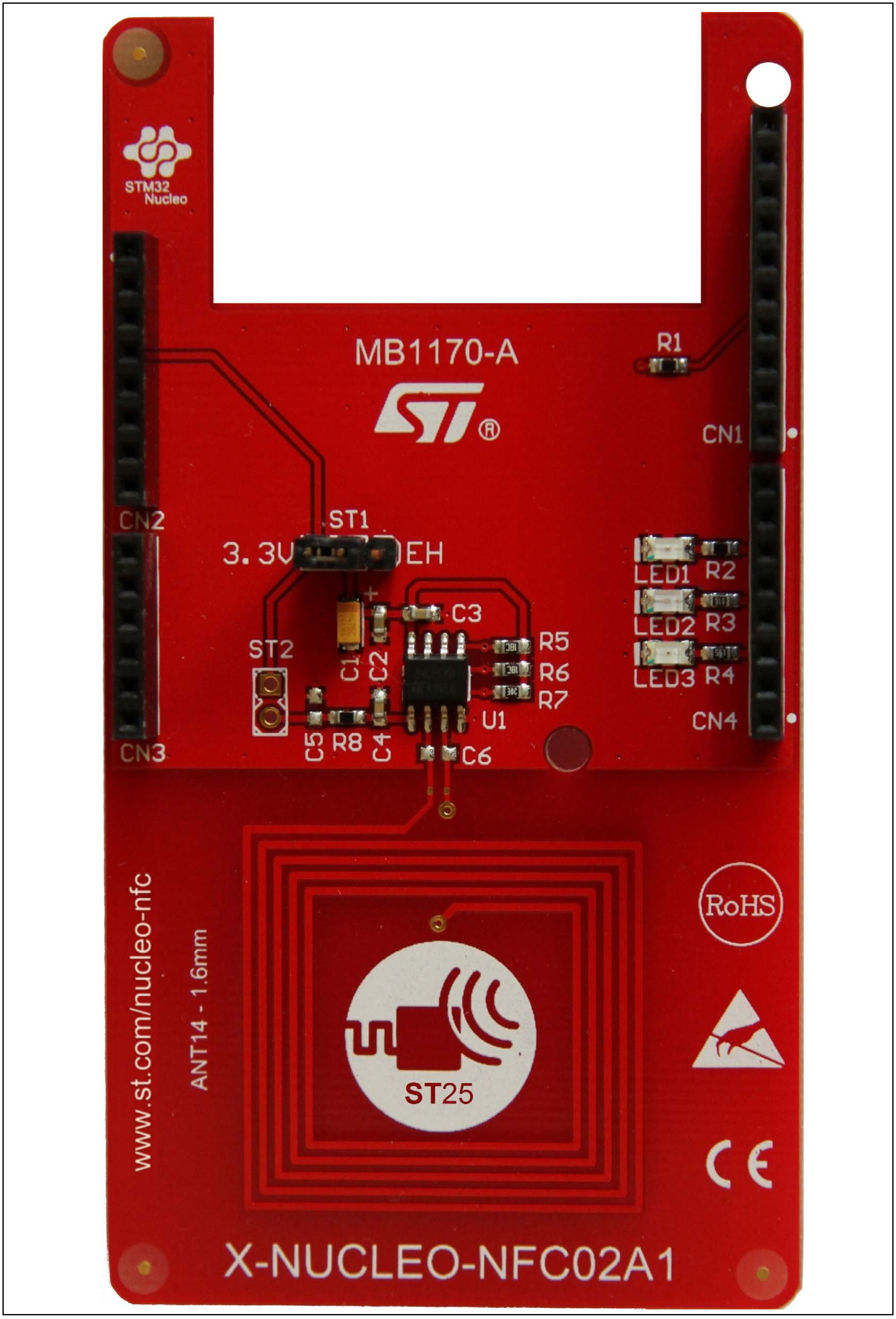
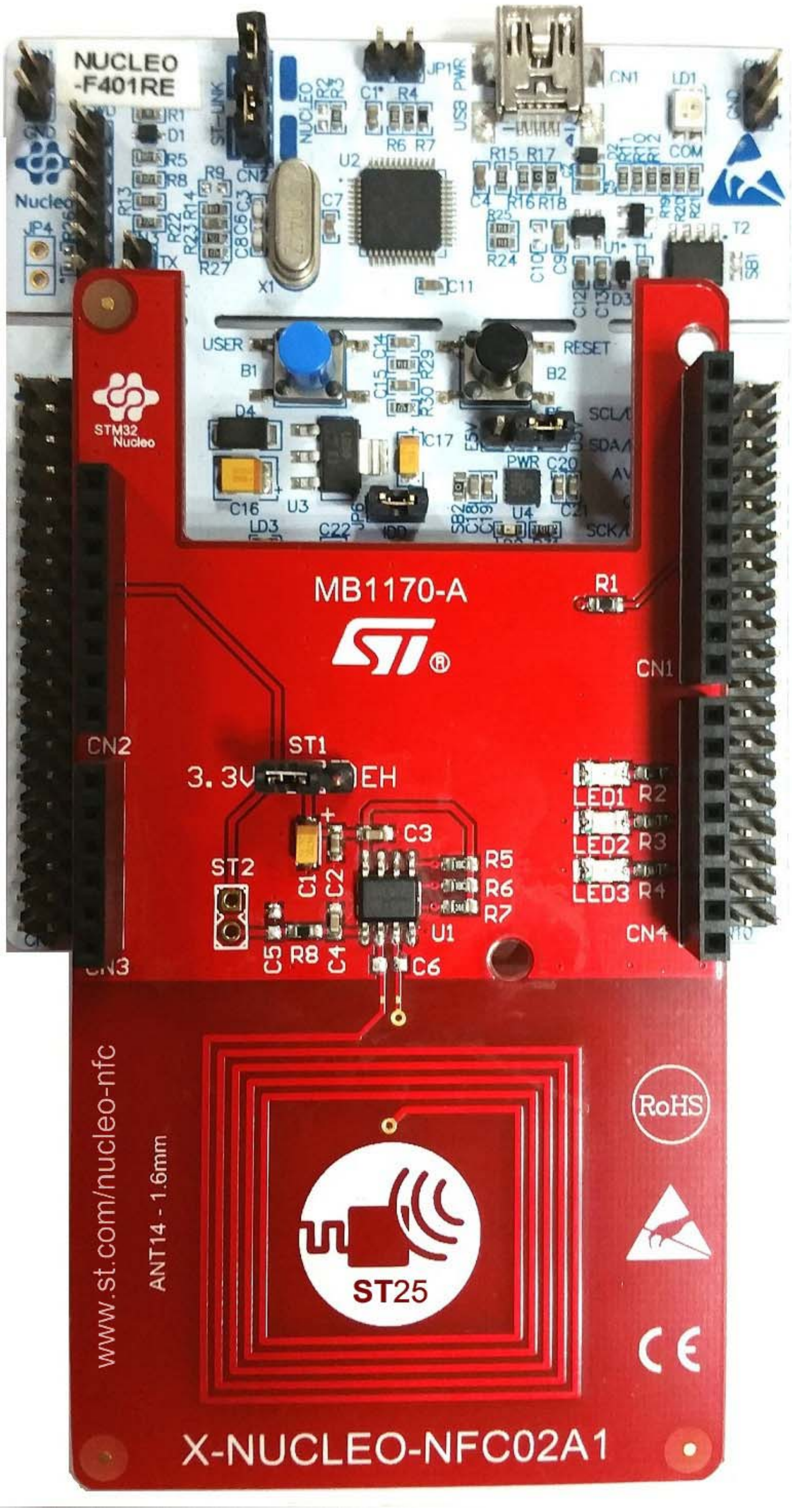


Figure 6. X-NUCLEO-NFC02A1 expansion board connected to STM32 Nucleo board



Information about the X-NUCLEO-NFC02A1 expansion board is available on *www.st.com*.

5.2 Software description

The following software components are needed to set up a suitable development environment to create applications for STM32 Nucleo with the NFC expansion board:

- X-CUBE-NFC2: an expansion dedicated to NFC applications development. The firmware and its related documentation are available on *www.st.com*.
- Development tool-chain and Compiler: the STM32Cube expansion software supports the three following environments:
  - IAR™ Embedded Workbench® for ARM® (EWARM) toolchain + ST-LINK
  - RealView® Microcontroller Development Kit (MDK-ARM™) toolchain + ST-LINK
  - System Workbench for STM32 (SW4STM32) + ST-LINK

5.3 Hardware and software setup

This section describes the hardware and software setup procedures, and the required system setup.

5.3.1 Hardware setup

The following hardware components are needed:

- a STM32 Nucleo Development platform (order code: either NUCLEO-F401RE or NUCLEO-L053R8)
- an NFC/RFID tag IC expansion board (order code: X-NUCLEO-NFC02A1)
- a USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

5.3.2 Software setup

This section lists the minimum requirements for the developer to set up the SDK.

Development Tool-chains and Compilers

Select one of the Integrated Development Environments supported by the STM32Cube expansion software.

Read the system requirements and setup information provided by the selected IDE provider.

5.3.3 System setup guide

This section describes how to set up different hardware parts before writing and executing an application on the STM32 Nucleo board with the Sensors expansion board.

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer.

The developer can download the ST-LINK/V2-1 USB driver in the STSW-LINK009 software available on *www.st.com*.

The X-NUCLEO-NFC02A1 expansion board can be easily connected to the STM32 Nucleo motherboard through the Arduino™ UNO R3 extension connector (*Figure 6*). It is capable of interfacing with the external STM32 microcontroller on STM32 Nucleo using Inter-Integrated Circuit (I2C) transport layer.

Further documentation to get started with these boards is available on *www.st.com*.

6 Revision history

Table 2. Document revision history

Date	Revision	Changes
28-Jan-2016	1	Initial release.