UM2045
User manual

Getting started with the X-CUBE-NFC3 near field communication transceiver software expansion for STM32Cube™

Introduction

This document describes how to get started with the X-CUBE-NFC3 software expansion for STM32Cube™.

X-CUBE-NFC3 provides a complete middleware for STM32 to build applications using the ST25R95 or CR95HF near field communication transceivers. It is easily portable across different MCU families, thanks to STM32Cube™. The package contains a sample application to detect different types of NFC tags.

The software provides implementation examples for STM32 Nucleo platforms equipped with the X-NUCLEO-NFC03A1 expansion board, featuring the ST25R95 or CR95HF near field communication transceivers.

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

Contents

List of tables

List of figures

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
BSP	Boot support package
CMSIS	Arm® Cortex® microcontroller software interface standard
HAL	Hardware abstraction layer
LED	Light emitting diode
SPI	Serial peripheral interface
MCU	Micro controller unit
NFC	Near field communication
RFAL	RF abstract layer

2 References

- ST25R95 or CR95HF datasheets (available on *www.st.com*)
- STM32 microcontroller datasheets (available on *www.st.com*)
- Nucleo board 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.

The STM32 microcontrollers are based on Arm®(a) core(s).

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.

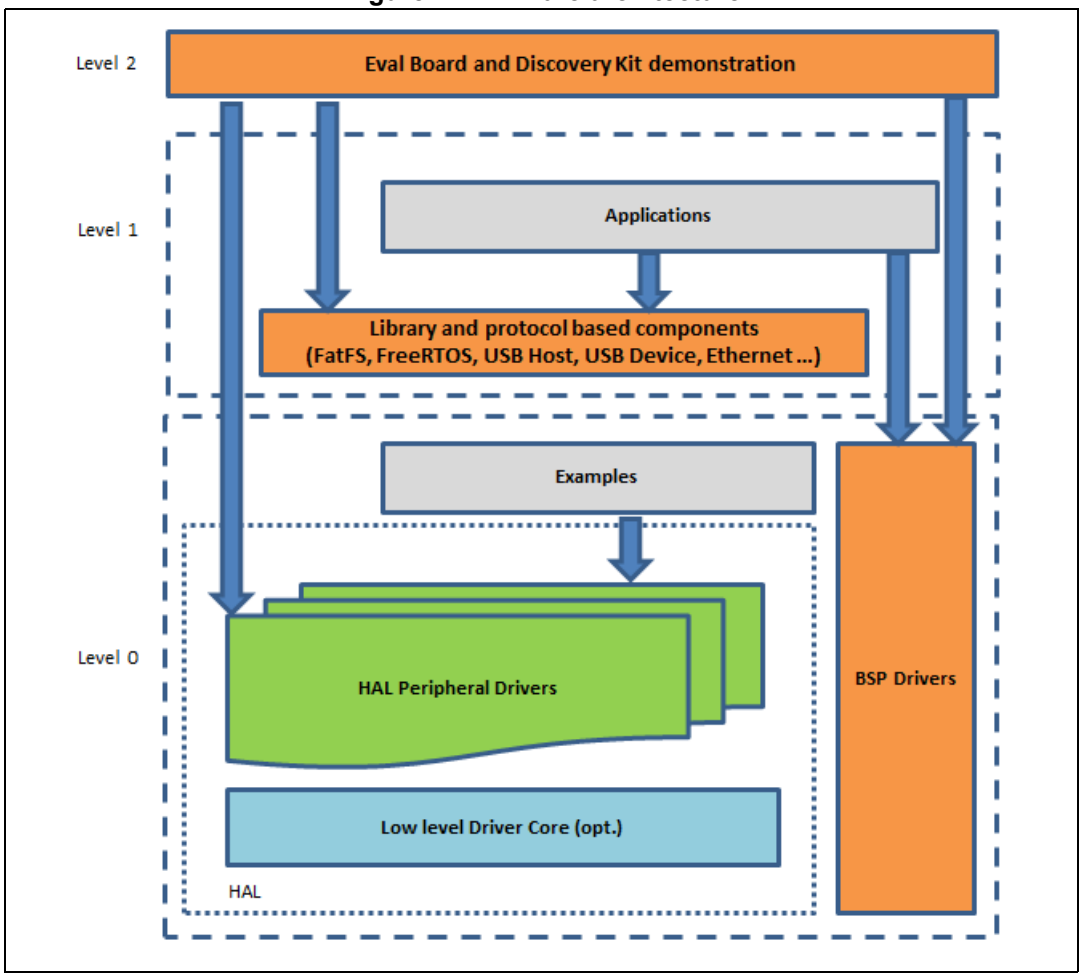
Information about STM32Cube™ are available on *www.st.com*.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

3.2 STM32Cube™ architecture

The STM32Cube™ firmware solution is built around three independent levels that can easily interact with each other's as described in the figure below:

Figure 1. Firmware architecture



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 in the hardware boards (Audio codec, IO expander, Touchscreen, SRAM driver, LCD drivers. etc…) and composed of two parts:
  - **Component**: is the driver relative to the external device on the board and not related to the STM32. The component driver provide specific APIs to the BSP driver external components and could be portable on any other board.
  - **BSP driver**: it permits to link the component driver to a specific board and provides a set of friendly used APIs. The APIs naming rule is BSP_FUNCT_Action(): ex. BSP_LED_Init(),BSP_LED_On().

Level 0 is based on modular architecture allowing to port it 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 a generic, multi instance and functionalities oriented APIs which permit to offload the user application implementation by providing ready to use process. As an example, for the communication peripherals (I2S, UART…), it provides APIs allowing to initialize and configure the peripheral, manage data transfer based on polling, interrupt or DMA process, and manage communication errors that may raise during communication.
  The HAL Drivers APIs are split in two categories:
  – **generic APIs**: it provides common and generic functions to all the STM32 series.
  – **extension APIs**: it provides customized functions for a specific family or a specific part number.
- **Basic peripheral usage examples**: this layer encloses the examples build over the STM32 peripheral using only the HAL and BSP resources.

Level 1: This level is divided into 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 done through specific callbacks and static macros implemented in the library system call interface. As 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:

This layer is global real-time and graphical demonstration. It is based on the middleware service layer, the low level abstraction layer and the basic peripheral usage applications for board based functionalities.

4 X-CUBE-NFC3 software expansion for STM32Cube™

4.1 Overview

X-CUBE-NFC3 is a software package that expands the functionality provided by STM32Cube™.

The key features of the package are:

- Complete middleware to build applications using the ST25R95 or CR95HF near field communication transceivers.
- Easy portability across different MCU families, thanks to STM32Cube™.
- Sample application example to detect NFC tags of different class.
- Free user-friendly license terms.
- Examples implementation available on board X-NUCLEO-NFC03A1 plugged on top of one NUCLEO-L476RG, NUCLEO-F401RE or NUCLEO-F103RB.

This software is gathering ST25R95 or CR95HF drivers for the device, running on STM32.

The software is built on top of STM32Cube™ software technology that ease portability across different STM32 microcontrollers.

The software comes with examples of implementation of such drivers, running on X-NUCLEO-NFC03A1 plugged on top of one NUCLEO-L476RG, NUCLEO-F401RE or NUCLEO-F103RB.

The sample application configures the ST25R95 or CR95HF for wakeup, followed by a polling loop for passive device detection. When a passive tag is detected, the shield signals the detected technology by lighting a corresponding LED.

The demo logs all activities with ST-Link Virtual Com Port to the host system.

The supported NFC technologies in this demo are:

- NFC-A \ ISO14443A (T1T, T2T, T4TA)
- NFC-B \ ISO14443B (T4TB)
- NFC-F \ FeliCa (T3T)
- NFC-V \ ISO15693 (T5T)
- ST25TB (ISO14443-2 Type B with proprietary protocol)

4.2 Architecture

This software expansion for STM32Cube™ lets you develop applications using the ST25R95 or CR95HF near field communication transceiver ICs. It is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller and extends STM32Cube™ with a board support package (BSP) for the X-NUCLEO-NFC03A1 expansion board.

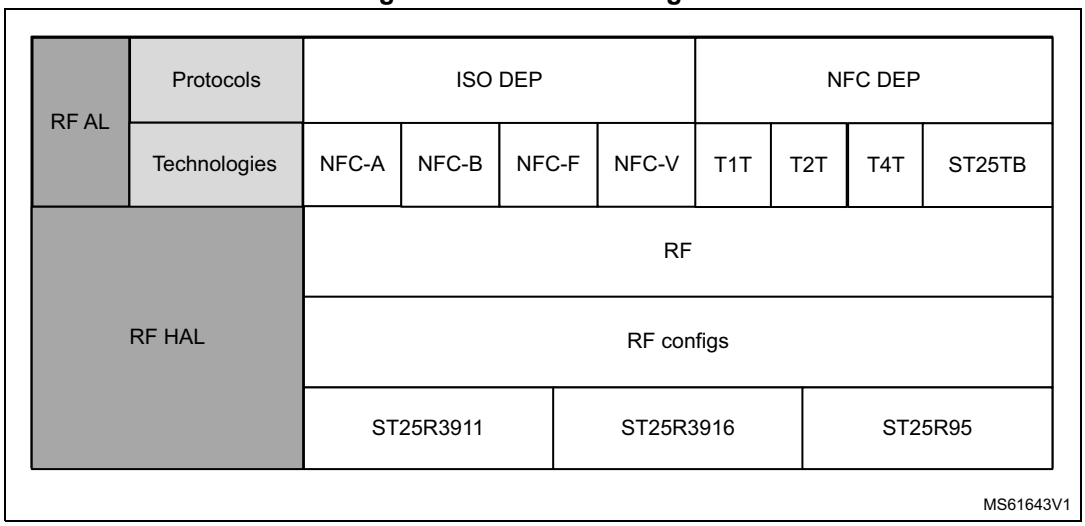
Application software can access and use the X-NUCLEO-NFC03A1 expansion board through the following layers:

- **STM32Cube HAL layer:** provides a simple set of generic, multiinstance APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). These generic and extension APIs are directly built on a common architecture and allow overlying layers like middleware to implement their functions without depending on specific microcontroller unit (MCU) hardware information. This structure improves the library code reusability and guarantees easy portability across other devices.
- **Board support package (BSP) layer**: provides support for the peripherals on the STM32 Nucleo board (apart from the MCU). This set of APIs provides a programming interface for certain board-specific peripherals like the LED, the user button etc. This interface also helps you identify the specific board version.
- **Middleware RF abstraction layer (RFAL)**: RFAL provides several functions for RF/NFC communication. It groups the different RF ICs (for instance ST25R95/CR95HF or ST25R3911) under a common and easy to use interface. The technologies currently supported by RFAL are:
  - NFC-A \ ISO14443A (T1T, T2T, T4TA)
  - NFC-B \ ISO14443B (T4TB)
  - NFC-F \ FeliCa (T3T)
  - NFC-V \ ISO15693 (T5T)
  - ST25TB (ISO14443-2 Type B with Proprietary Protocol)

The RFAL provides support of Data Exchange Protocols. Internally, the RFAL is divided into two sub layers:

- RF HAL- RF hardware abstraction layer
- RF AL - RF abstraction layer

Figure 2. RFAL block diagram



The modules in the RF HAL are chip-dependent, they implement the RF IC driver, configuration tables and specific instructions for the HW to perform the physical RF functions. The interface for the caller is a shared RF header file which provides the same interface for upper layers (for all chips).

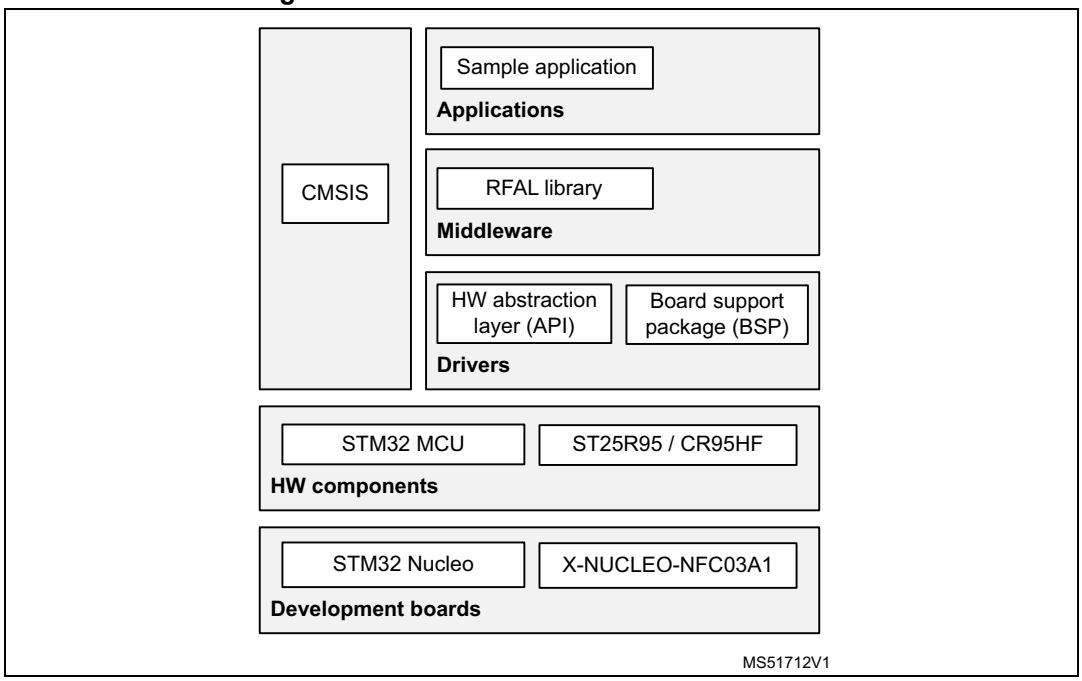
The RF AL can be broken down into two further sub layers:

- Technologies: technology modules which implement all the specifics, framing, timings, etc.
- Protocols: protocol implementation including all the framing, timings, error handling, etc.

On top of these, the application layer uses RFAL functions like NFC Forum Activities, etc.

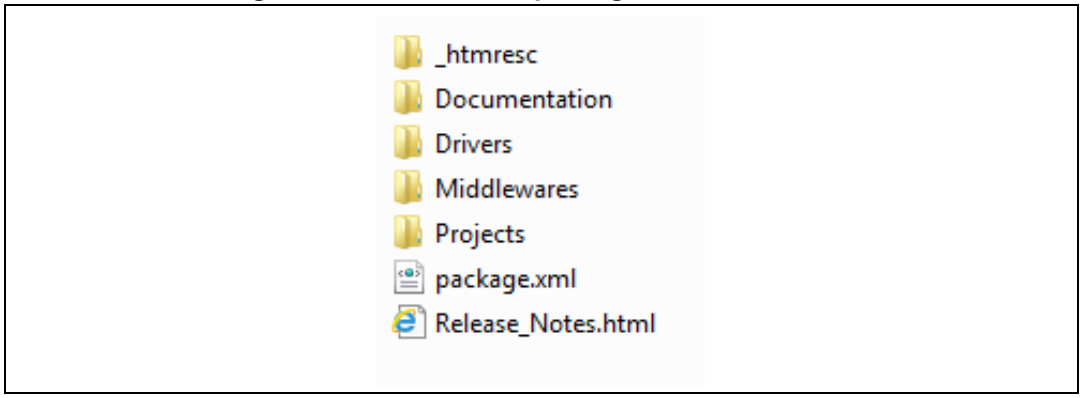
Access to the lowest functions of the ICs is granted by the RF module. The caller can make direct use of any of the RF technology or protocol layers without requiring any specific hardware configuration data.

Figure 3. X-CUBE-NFC3 software architecture



4.3 Folders structure

Figure 4. X-CUBE-NFC3 package folders structure



The following folders are included in the software package:

- **Documentation**: this folder contains a compiled HTML file generated from the source code, which details the software components and 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 vendor-independent hardware abstraction layer for the Cortex-M processor series.
- **Middlewares**: this folder contains RFAL (RF abstraction layer). RFAL provides several functions required to perform RF/NFC communication.

The RFAL groups the different ST25R RF ICs under a common and easy to use interface.

- **Projects**: this folder contains a sample application example Tag Detect, provided for the NUCLEO-L476RG and NUCLEO-F401RE platforms with three development environments (IAR embedded workbench for Arm®, Keil microcontroller development Kit (MDK-ARM), and system workbench for STM32 (SW4STM32).

An RFAL usage example as a Poller device is provided in exampleRfalPoller.c. In this example, different devices are detected and activated, and data is exchanged implementing a presence check mechanism. Once removed or upon error, the device is deactivated and the discovery loop restarts.

4.4 APIs

Detailed technical information about the APIs available to the user can be found in a compiled CHM file located inside the “Documentation” folder of the software package where all the functions and parameters are fully described.

4.5 Sample application description

A sample application using the X-NUCLEO-NFC03A1 expansion board with the STM32 Nucleo development board is provided in the “Projects” directory. Ready-to-build projects are available for multiple IDEs.

In this application, NFC tags of different types are detected by the ST25R95 or CR95HF near field communication transceivers.

See the CHM documentation file generated from the source code for more details regarding the sample application.

After system initialization and clock configuration, LED1, LED2, LED3 and LED4 blink 3 times.

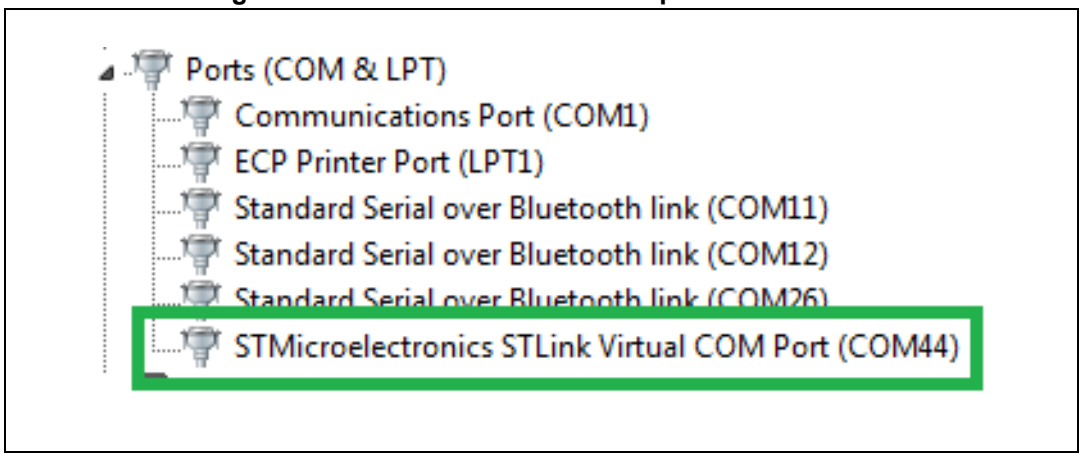
When a tag is detected in the vicinity, a LED is lit on the NFC3 shield according to the table below.

Table 2. LED lit on tag detection

NFC Tag Type	LEDs lit on tag detection
NFC TYPE A	LED1
NFC TYPE B	LED2
NFC TYPE F	LED3
NFC TYPE V	LED4

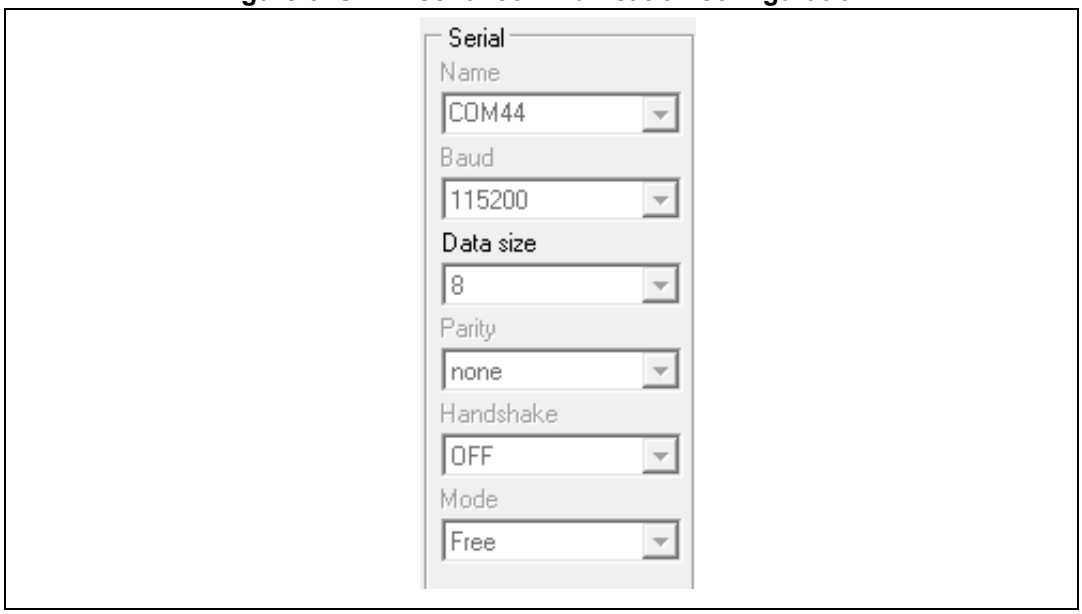
ST’s virtual comport interface is also included: following system initialization, the board is configured and enumerated as an ST Virtual comport.

Figure 5. ST virtual communication port enumeration



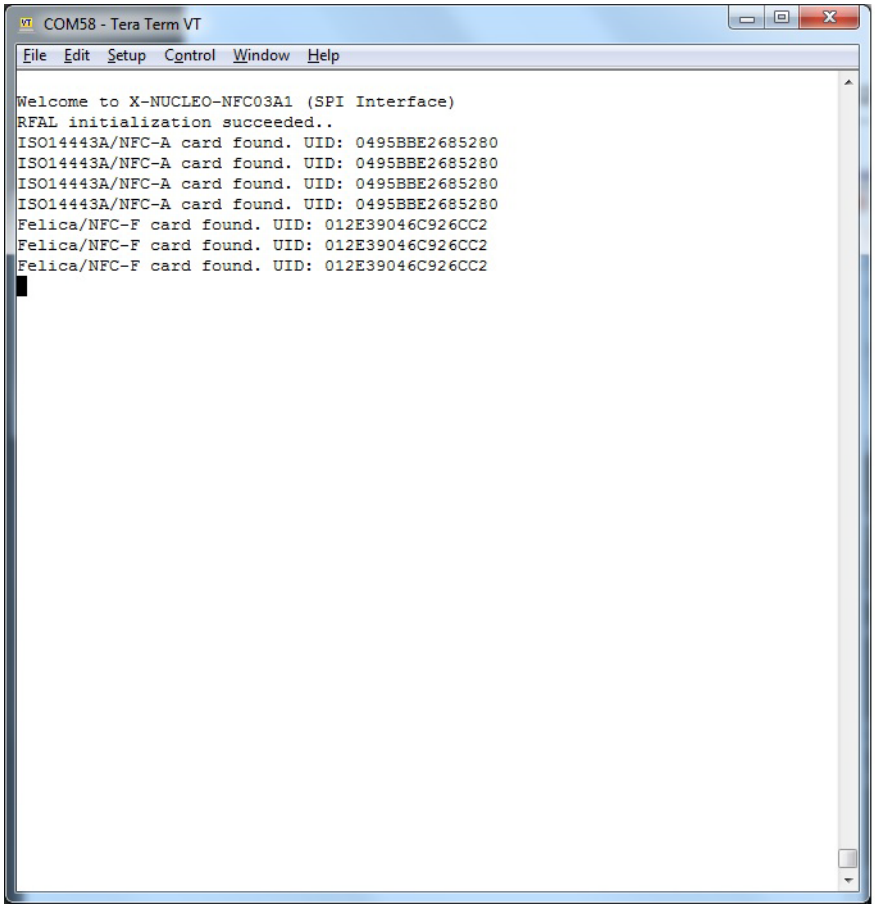
After checking the virtual COM port number, open a connection on Hyperterminal (or similar) with the configuration shown below (enable option: Implicit CR on LF, if available).

Figure 6. UART serial communication configuration



Following successful connection, the user can view the messages on the hyperterminal, as shown below.

Figure 7. UART serial communication displayed on Hyperterminal



5 System setup guide

5.1 Hardware description

This section describes the hardware components required to develop a sensor-based application.

The following sub-sections describe the individual components.

5.1.1 STM32 Nucleo platform

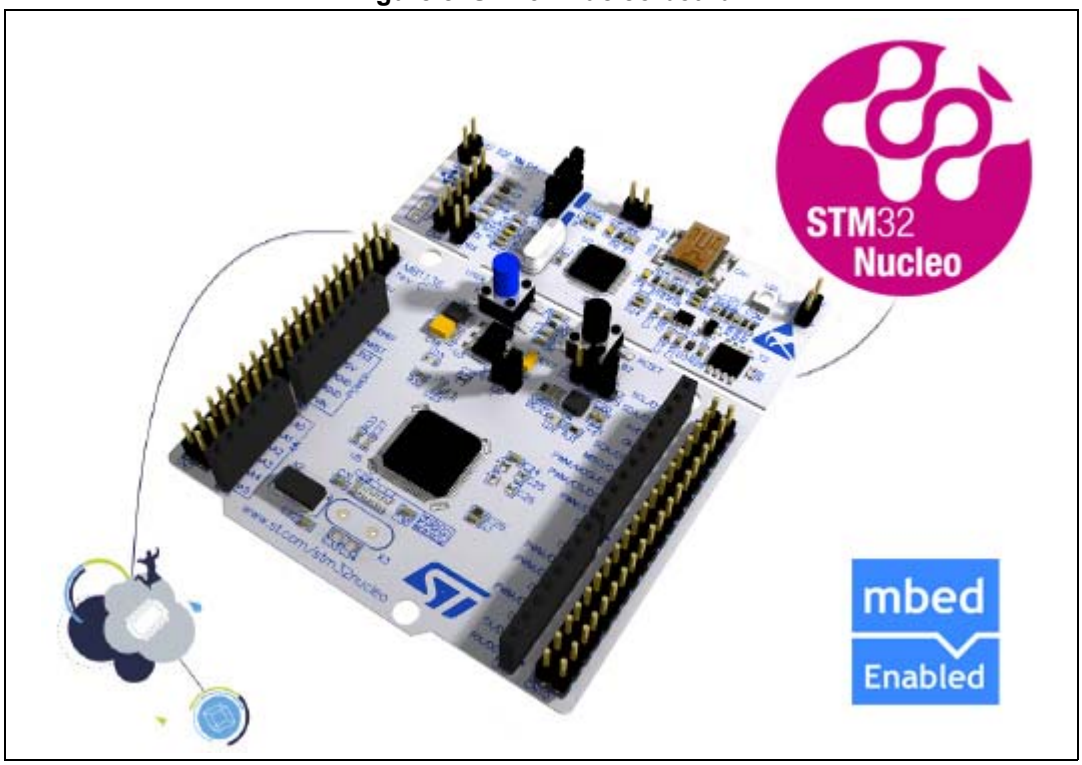
The STM32 Nucleo boards provide an affordable and flexible way for users to test new ideas and build prototypes with any STM32 microcontroller lines.

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 8. STM32 Nucleo board



5.1.2 X-NUCLEO-NFC03A1 expansion board

The X-NUCLEO-NFC03A1 is a contactless transceiver IC expansion board that can be used with the STM32 Nucleo platform. It is also compatible with Arduino™ UNO R3 connector layout, and is designed around the STMicroelectronics ST25R95 or CR95HF near field communication transceiver. The X-NUCLEO-NFC03A1 interfaces with the STM32 MCU via SPI/UART pin.

Figure 9. X-NUCLEO-NFC03A1 expansion board

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

5.2 Software description

The following software components are needed in order to setup the suitable development environment to create applications for the STM32 Nucleo with the NFC expansion board:

- X-CUBE-NFC3: an expansion for STM32Cube™ dedicated to NFC applications development. The X-CUBE-NFC3 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. It also describes the required system setup.

5.3.1 Hardware setup

The following hardware components are needed:

- One STM32 Nucleo Development platform (order code: NUCLEO-L476RG, NUCLEO-F401RE or NUCLEO-F103RB)
- One ST25R95 (CR95HF) near field communication transceiver expansion board (order code: X-NUCLEO-NFC03A1)
- One 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 setup 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 setup different hardware parts before writing and executing an application on the STM32 Nucleo board with the Sensors expansion board.

STM32 Nucleo, ST25R95 (CR95HF) expansion boards setup

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

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

The X-NUCLEO-NFC03A1 expansion board can be easily connected to the STM32 Nucleo motherboard through the Arduino™ UNO R3 extension connector. It is capable of

interfacing with the external STM32 microcontroller on STM32 Nucleo board using SPI/UART transport layer. By default it will run using SPI interface. The user can choose to compile with one of the two interfaces by selecting the right target under Projects properties.

6 Revision history

Table 3. Document revision history

Date	Revision	Changes
26-May-2016	1	Initial release
15-Jan-2019	2	Added ST25R95. Updated: – *Introduction* – *Table 1: List of acronyms* – *Section 4.1: Overview* – *Section 4.2: Architecture* – *Section 4.3: Folders structure* – *Section 4.4: APIs* – *Section 4.5: Sample application description* – *Section 5.3.1: Hardware setup*