UM2147
User manual

Getting started with X-CUBE-LED1642 16 channel LED driver software expansion based on LED1642GW for STM32Cube

Introduction

The X-CUBE-LED16A1 expansion software package for STM32Cube runs on the STM32 microcontroller with drivers and sample applications for the LED1642GW LED driver.

An SPI interface facilitates communication between the STM32 Nucleo board and the X-NUCLEO-LED16A1 expansion board. It is built on top of STM32Cube software technology that ease portability across different STM32 microcontrollers.

The software includes sample driver implementations for the X-NUCLEO-LED16A1 expansion board on a NUCLEO-F401RE or NUCLEO-L053R8 development board.

Contents

List of figures

1 What is STM32Cube?

1.1 What is STM32Cube?

STMCube™ represents the STMicroelectronics initiative to make developers' lives easier by reducing development effort, time and cost. STM32Cube covers the STM32 portfolio.

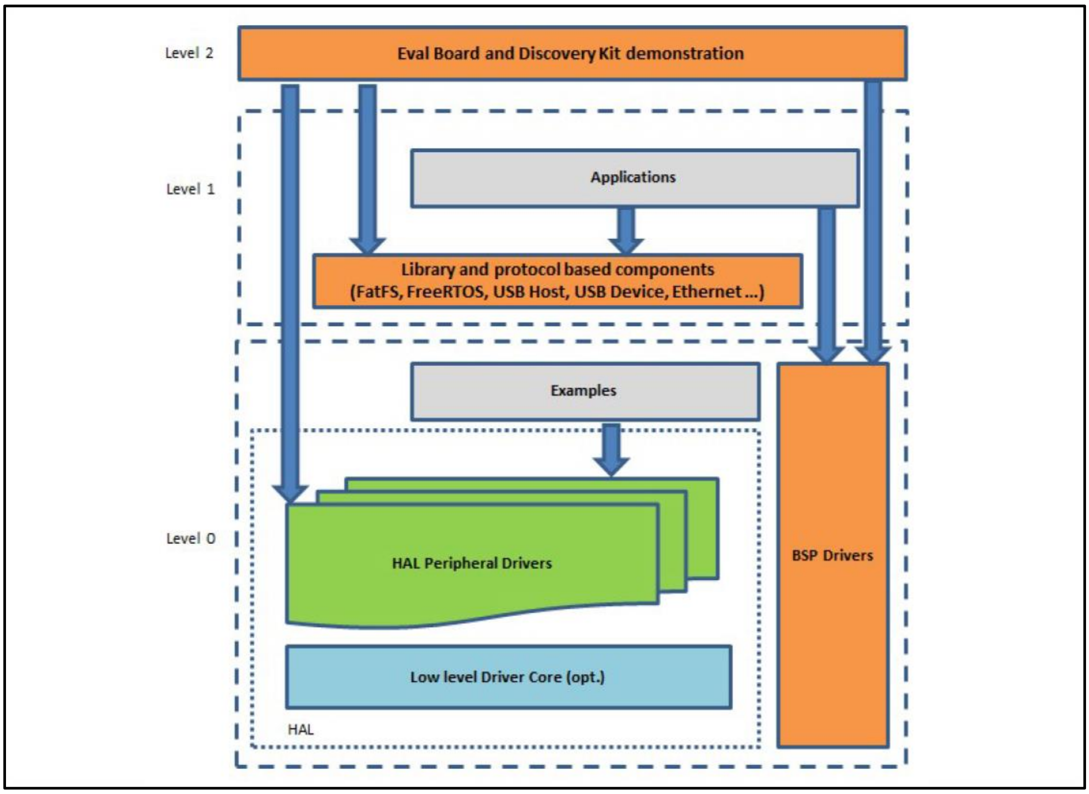
STM32Cube version 1.x includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform specific to each series (such as the STM32Cube for the STM32 series), which includes:
  - the STM32Cube HAL embedded abstraction-layer software, ensuring maximized portability across the STM32 portfolio
  - a consistent set of middleware components such as RTOS, USB, TCP/IP and graphics
  - all embedded software utilities with a full set of examples

1.2 STM32Cube architecture

The STM32Cube firmware solution is built around three independent levels that can easily interact with one another, as described in the diagram below.

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
Low level Driver Core (opt.)
HAL
BSP Drivers

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…); it is based on modular architecture allowing it to be easily ported on any hardware by just implementing the low level routines. It is 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 provides specific APIs to the external components of the BSP driver, and can be ported on any other board.
  - BSP driver: links the component driver to a specific board and provides a set of easy to use APIs. The API naming convention is BSP_FUNCT_Action(): e.g., BSP_LED_Init(), BSP_LED_On().

- 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). It provides generic, multi-instance and function-oriented APIs to help offload user application development time by providing ready to use processes. For example, for the communication peripherals (I²C, UART, etc.) it provides APIs for peripheral initialization and configuration, data transfer management based on polling, interrupt or DMA processes, and communication error management. The HAL Drivers APIs are split in two categories: generic APIs providing common, generic functions to all the STM32 series and extension APIs which provide special, customized functions for a specific family or a specific part number.
- Basic peripheral usage examples: this layer houses the examples built around the STM32 peripherals using the HAL and BSP resources only.

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 interaction among the components in this layer is performed directly by calling the feature APIs, while vertical interaction with low-level drivers is managed by specific callbacks and static macros implemented in the library system call interface. For example, FatFs implements the disk I/O driver to access a microSD drive or USB Mass Storage Class.
- Examples based on the middleware components: each middleware component comes with one or more examples (or applications) showing how to use it. Integration examples that use several middleware components are provided as well.

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

2 X-CUBE-LED1642 software expansion for STM32Cube

2.1 Overview

The X-CUBE-LED1642 software package expands the functionality provided by STM32Cube.

The key features of the package are:

- Complete middleware package to build applications using the LED1642GW LED driver on the X-NUCLEO-LED16A1 expansion board for STM32 Nucleo
- SPI interface between STM32 Nucleo and X-NUCLEO-LED16A1 expansion board
- Multiple stacking of X-NUCLEO-LED16A1 expansion boards allowed
- Easy portability across different MCU families, thanks to STM32Cube
- Free user-friendly license terms
- Sample implementations for X-NUCLEO-LED16A1 expansion board(s) plugged on a NUCLEO-F401RE or NUCLEO-L053R8 development board

2.2 Architecture

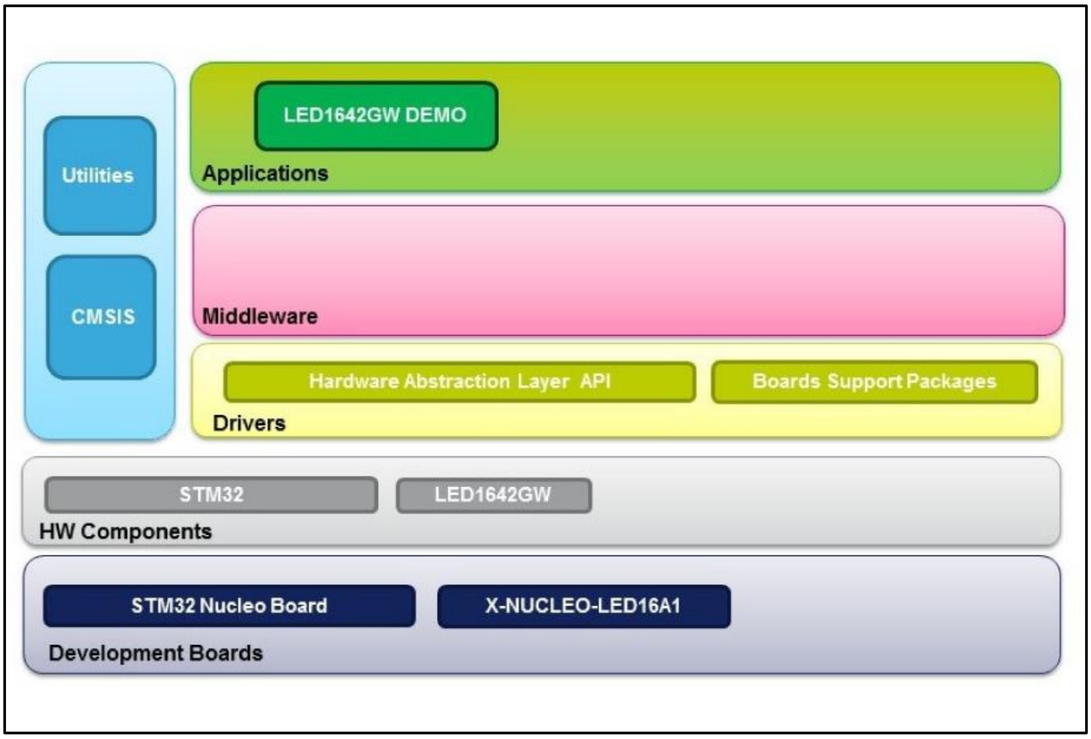
This software is a fully compliant STM32Cube expansion for application development involving DC-DC LED driver devices.

The software is based on the STM32Cube HAL (hardware abstraction layer) for the STM32 microcontroller and extends STM32Cube by providing a board support package (BSP) for the X-NUCLEO-LED16A1 expansion board and a sample application for serial communication with a PC.

The software layers used by the application software to access and use the X-NUCLEO-LED16A1 expansion board are:

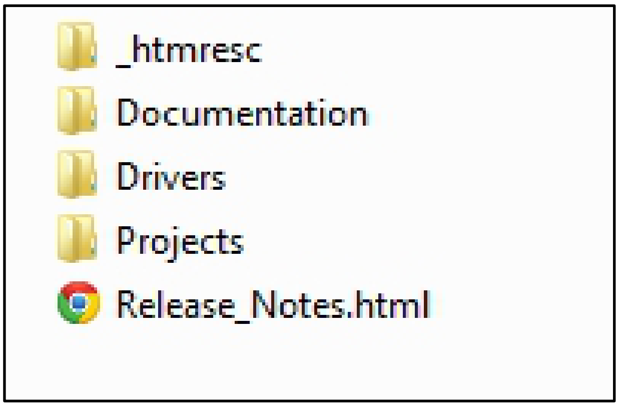
- **STM32Cube HAL layer**: consists of simple, generic and multi-instance APIs (application programming interfaces) which interact with the upper layer applications, libraries and stacks. These generic and extension APIs are based on a common framework so that overlying layers like middleware can function without requiring specific microcontroller unit (MCU) hardware information. This structure improves library code reusability and guarantees easy portability across other devices.
- **Board support package (BSP) layer**: provides software support for the STM32 Nucleo board peripherals, excluding the MCU. These specific APIs provide a programming interface for certain board specific peripherals like LEDs, user buttons, etc., and can also be used to fetch individual board version information. It also provides support for initializing, configuring and reading data.

Figure 2: X-CUBE-LED1642 software architecture



2.3 Folders structure

Figure 3: X-CUBE-LED1642 package folders structure



_htmresc
Documentation
Drivers
Projects
Release_Notes.html

The following folders are included in the software package:

The **Documentation** folder contains a compiled HTML file generated from the source code and detailed documentation regarding the software components and APIs.

The **Drivers** folder contains the HAL drivers, the board-specific drivers for each supported board or hardware platform, including those for the on-board components, and the CMSIS vendor-independent hardware abstraction layer for the Cortex-M processor series.

The **Projects** folder contains a sample application for the NUCLEO-L053R8 and NUCLEO-F401RE platforms; it is provided with three development environments (IAR Embedded Workbench for ARM, RealView MDK-ARM Microcontroller Development Kit and System Workbench for STM32).

2.4 APIs

A compiled HTML file located in the package Documentation folder provides full user-API function and parameter descriptions.

2.5 Sample application description

A sample application using the X-NUCLEO-LED16A1 expansion board with either NUCLEO-F401RE or NUCLEO-L053R8 boards is provided in the Projects directory, ready for multiple IDEs.

This application operates the LED driver expansion board, implementing a state machine to demonstrate various demos including PWM dimming, error detection and different LED patterns.

2.6 Preconfigured demos

Once the proper firmware is installed on the STM32 Nucleo board and the jumpers and connectors are configured correctly, supplying power causes the green OK LED (D2) on the X-NUCLEO-LED16A1 expansion board and predefined LED patterns run every eight seconds:

1. All LEDs ON at full brightness
2. LED train forward – single LED
3. LED train reverse
4. LED brightness – maximum at center LED
5. Error detection – red FAULT LED (D1) lights up
6. LED brightness – low to high
7. Different LED – solid color
8. Random LED – random brightness
9. All LEDs ON with less brightness

You can also cycle through the LED sequences by pressing the blue B1 USER button on the STM32 Nucleo board.

Incorrect jumper configurations causes the red LED (FAULT) to light up.

3 System setup guide

3.1 Hardware description

This section describes the hardware components needed for developing a LED driver application based on LED1642GW.

3.1.1 STM32 Nucleo platform

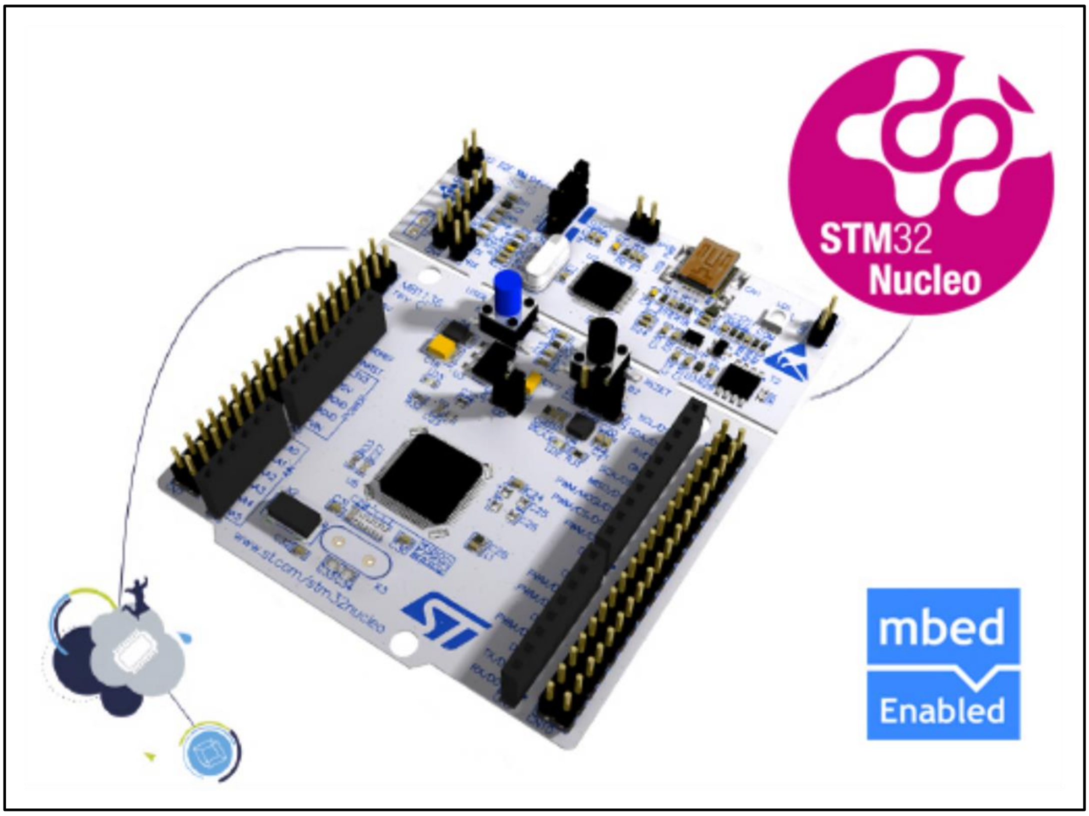
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

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 range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples.

Figure 4: STM32 Nucleo board

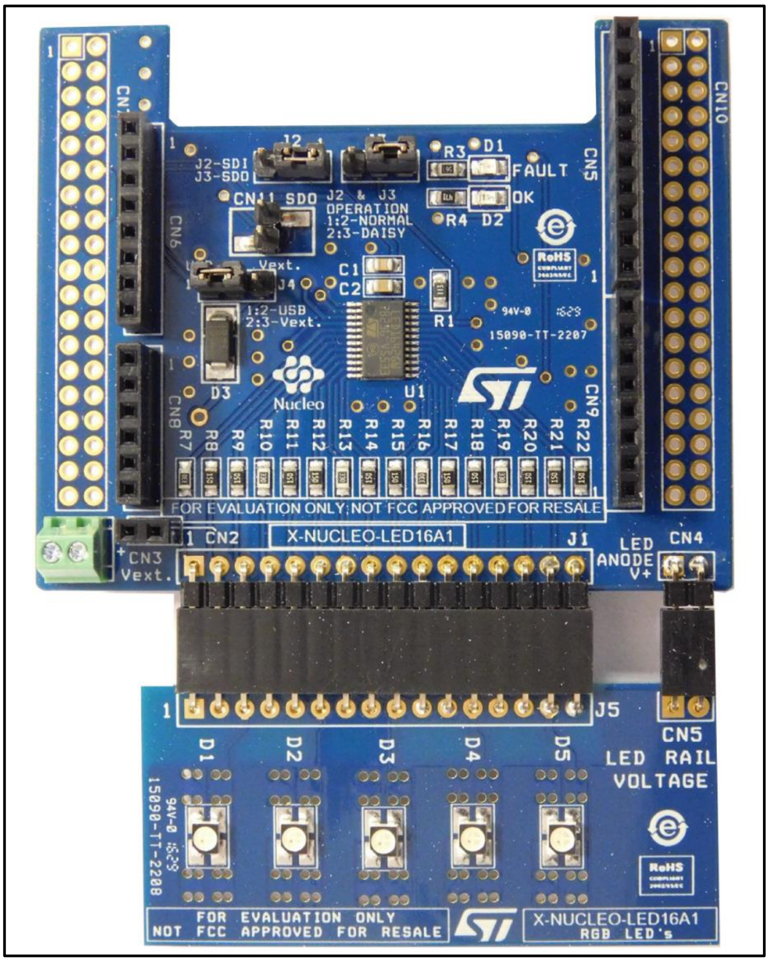


Information regarding the STM32 Nucleo board is available at *www.st.com/stm32nucleo*

3.1.2 X-NUCLEO-LED16A1 expansion board

The X-NUCLEO-LED16A1 expansion board for STM32 Nucleo is based on the LED1642GW 16 channel LED driver. It is compatible with the Arduino UNO R3 platform and interfaces with the STM32 microcontroller via SPI interface.

Figure 5: X-NUCLEO-LED16A1 expansion board



Information regarding the X-NUCLEO-LED16A1 expansion board is available on www.st.com at *www.st.com/x-nucleo*

3.2 Software description

The following software components are required in order to setup a suitable development environment for creating applications with the STM32 Nucleo development board plus X-NUCLEO-LED16A1 expansion board:

1. **X-CUBE-LED1642**: STM32Cube expansion for LED driver application development; the firmware and documentation is available on www.st.com.
2. **Development tool-chain and compiler**: the STM32Cube expansion software supports these environments:
   a. IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-LINK
   b. RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
   c. system Workbench for STM32 (SW4STM32) + ST-LINK

3.3 Hardware and software setup

This section describes the hardware and software setup procedures. It also describes the system setup needed for the above.

3.3.1 Hardware setup

Set up the hardware with the following components and accessories:

- an STM32 Nucleo development platform (order code: either NUCLEO-F401RE or NUCLEO-L053R8)
- the 16 channel LED driver expansion board (order code: X-NUCLEO-LED16A1)
- an optional string of 16 LEDs connected in parallel or X-NUCLEO-LED16A1 expansion board
- a USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC
- an optional external power supply, 4.8-5.2 V/2 A (depending upon the number of stacked X-NUCLEO-LED16A1 boards)

3.3.2 Software setup

This section lists the minimum requirements for the developer for the setup, and customize applications.

Select one of the Integrated Development Environments supported by the STM32Cube expansion software and follow the system requirements and setup information provided by the same IDE provider.

Your PC should have:

- Windows 7 or above
- 40 MB hard disk space and 128 MB RAM available
- 2 x USB ports

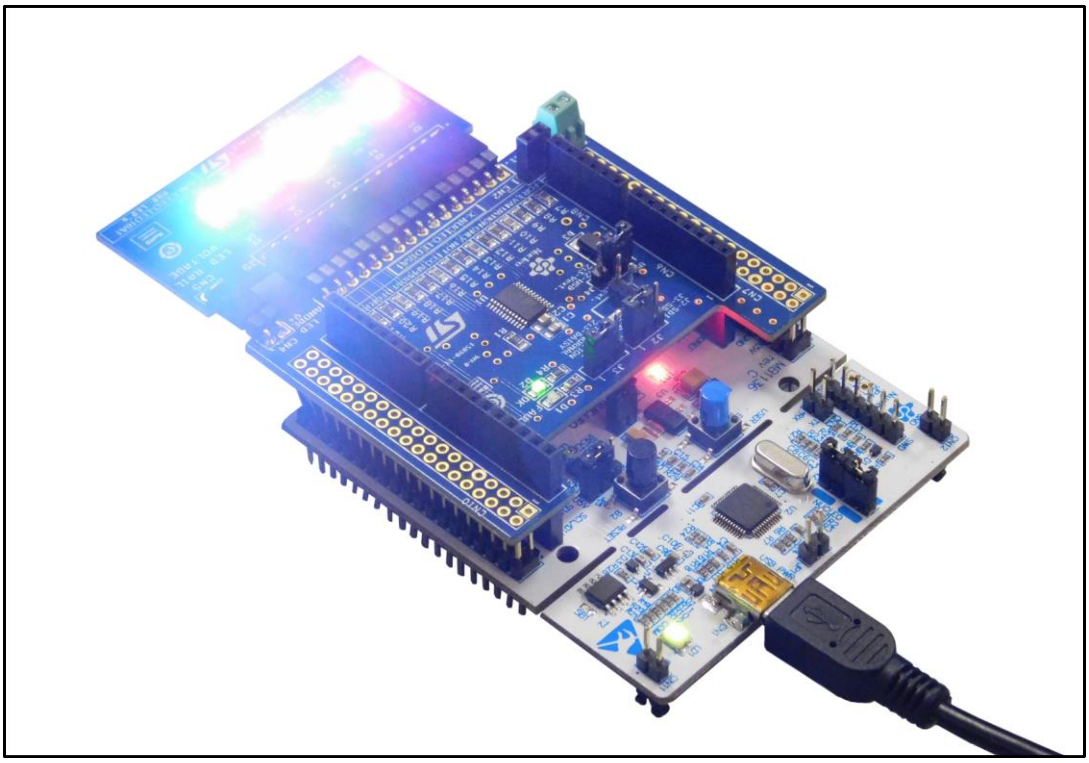
3.4 System setup guide

This section describes how to setup different hardware components before writing and executing an application on the STM32 Nucleo board with the X-NUCLEO-LED16A1 expansion board.

3.4.1 STM32 Nucleo and LED driver expansion boards setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. Developers can download the relevant version of the ST-LINK/V2-1 USB driver by searching STSW-LINK008 or STSW-LINK009 (depending on your Windows version) on www.st.com.

Figure 6: STM32 Nucleo development board plus X-NUCLEO-LED16A1 expansion board



The X-NUCLEO-LED16A1 expansion board can be easily connected to the STM32 Nucleo development board through the Arduino UNO R3 extension connector and can interface with the STM32 microcontroller on STM32 Nucleo via the Serial Peripheral Interface (SPI) transport layer.

4 Revision history

Table 1: Document revision history

Date	Version	Changes
23-Nov-2016	1	Initial release.

IMPORTANT NOTICE – PLEASE 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. 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 acknowledgement.

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 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.

ST and the ST logo are trademarks of ST. 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.