Getting started with FP-LIT-BLEMESH1 function pack for IoT nodes with Bluetooth® Low Energy mesh connectivity and lighting mode

Introduction

FP-LIT-BLEMESH1 is an STM32Cube function pack, which lets you connect Bluetooth® Low Energy nodes to a smartphone via Bluetooth® Low Energy, through a suitable Android™ or iOS™ application or use the Ambient Light Sensor on X-NUCLEO-6283A1 to set the HSL values and send the data to the lighting hardware using the Bluetooth® Low Energy mesh lighting model.

The software lets you easily create your own application for extending Bluetooth® mesh networks (by offering a ready-to-use mesh core library), a complete set of compatible APIs, and a lighting demo application running on either X-NUCLEO-IDB05A2 or X-NUCLEO-BNRG2A1, X-NUCLEO-LED12A1 and X-NUCLEO-6283A1 expansion boards connected to a NUCLEO-L476RG development board.

The software runs on the STM32 microcontroller and includes all the necessary drivers to recognize the devices on the STM32 Nucleo development board and the expansion boards.

Related links

Visit the STM32Cube ecosystem web page on www.st.com for further information

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
GATT	Generic attribute profile
BSP	Board support package
HAL	Hardware abstraction layer
SPI	Serial peripheral interface
CMSIS	Cortex® microcontroller software interface standard
HSL	Hue saturation lighting

2 FP-LIT-BLEMESH1 software expansion for STM32Cube

2.1 Overview

The FP-LIT-BLEMESH1 software package expands STM32Cube functionality.

The key features of the package are:

- Complete software to build a mesh network with Bluetooth® Low Energy nodes supporting the Bluetooth® mesh lighting model, defined in Bluetooth® mesh specification V1.0.1
- Hue, saturation, and lightness (HSL) values set by the STBLEMesh Android and iOS app using the lighting model or from Ambient Light Sensor on X-NUCLEO-6283A1, changes the RGB values of the X-NUCLEO-LED12A1 LED expansion board connected to a NUCLEO-L476RG
- Compatible with BLE-enabled smartphones to monitor and control multiple Bluetooth® Low Energy nodes, using the proxy protocol and legacy Bluetooth® Low Energy GATT connectivity
- Two-layer security, thanks to the 128-bit AES CCM encryption and 256-bit ECDH protocol, ensuring protection from multiple attacks, including Replay, Bit-Flipping, Eavesdropping, Man-in-the-Middle, and Trashcan
- Sample implementation available on:
  - the X-NUCLEO-IDB05A2, X-NUCLEO-LED12A1 and X-NUCLEO-6283A1 expansion boards connected to a NUCLEO-L476RG development board
  - the X-NUCLEO-BNRG2A1, X-NUCLEO-LED12A1 and X-NUCLEO-6283A1 expansion boards connected to a NUCLEO-L476RG development board
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms

The function pack software includes the LED1202, which is a 12-channel low quiescent current LED driver (X-NUCLEO-LED12A1) and VD6283TX which is a multispectral ambient light sensor (X-NUCLEO-6283A1). These expansion boards are mounted on top of the STM32 Nucleo.

The package is compatible with the STBLEMesh Android/iOS application available at GooglePlay/iTunes stores, which can be used to set information and send it via Bluetooth® Low Energy. It integrates BlueNRG products with the embedded Bluetooth® Low Energy communication in a powerful, range-extending mesh network with real full-duplex communication. The package flexibility allows you to build your own application.

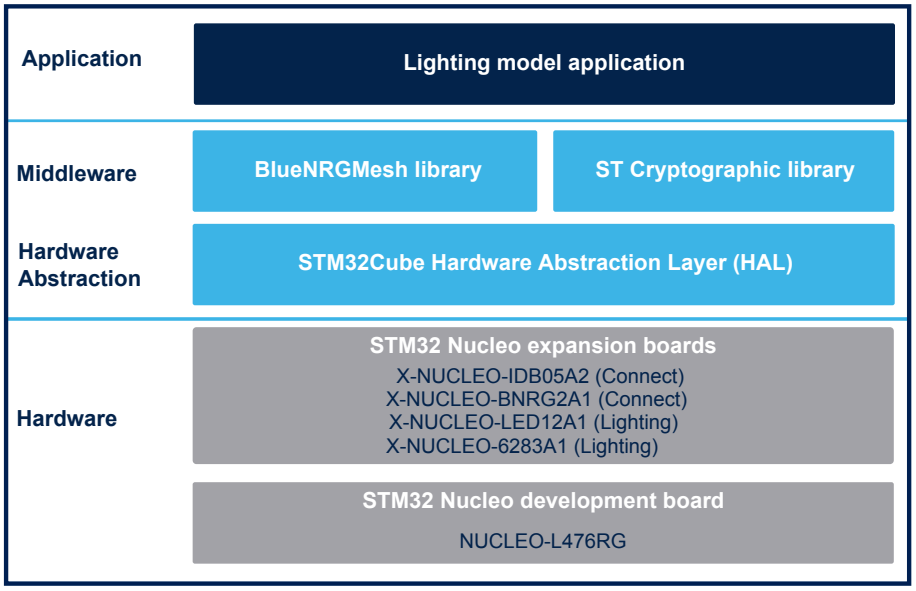
2.2 Architecture

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) to enable development of applications using Bluetooth mesh profile and model specifications.

The software layers used by the application software to access and use the expansion boards are:

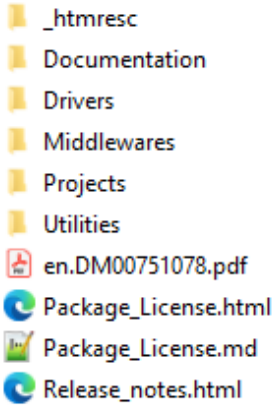
- the **STM32Cube HAL layer**, which provides a simple, generic, multi-instance set of application programming interfaces (APIs) to interact with the upper application, library, and stack layers. It has generic and extension APIs and is directly built around a generic architecture and allows successive layers like the middleware layer to implement functions without requiring specific hardware configurations for a given microcontroller unit (MCU). This structure improves the library code reusability and guarantees an easy portability on other devices.
- the **board support package** (BSP) layer supports all the peripherals on the STM32 Nucleo except the MCU. This limited set of APIs provides a programming interface for certain board-specific peripherals like the LED, the user button, etc. This interface also helps in identifying the specific board version.

Figure 1. **FP-LIT-BLEMESH1 software architecture**



2.3 Folder structure

Figure 2. **FP-LIT-BLEMESH1 package folder structure**



The following folders are included in the software package:

- **Documentation**: contains a compiled HTML file generated from the source code, which details the software components and APIs.
- **Drivers**: contains the HAL drivers and 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 Arm® Cortex®-M processor series.
- **Middlewares**: contains libraries and protocols related to the Bluetooth and Bluetooth mesh profile and model specifications.
- **Projects**: contains a sample application used to update the RGB lights HSL value, provided for the NUCLEO-L476RG platform with three development environments, IAR Embedded Workbench for Arm (IAR-EWARM), RealView Microcontroller Development Kit (MDK-ARM-STM32), and STM32CubeIDE.
- **Utilities**: contains the STM32L4_MAC folder that provides an external MAC address.

2.4 APIs

Detailed technical information with full user API function and parameter description are in a compiled HTML file in the “Documentation” folder.

2.5 Sample application description

2.5.1 Initialization of application callbacks

The "Projects" directory provides an example application using the X-NUCLEO-IDB05A2 or X-NUCLEO-BNRG2A1, X-NUCLEO-6283A1 and X-NUCLEO-LED12A1 expansion boards with the NUCLEO-L476RG development board.

Ready to be built projects are available for multiple IDEs.

This application starts by initializing the callbacks required for the different events and functionalities. The callbacks are used in the BlueNRG-Mesh library to call the functions based on specific events or by the mesh library state machine.

```
__attribute__((aligned(4)))
const MODEL_SIG_cb_t Model_SIG_cb[] =
#ifdef ENABLE_LIGHT_MODEL_SERVER
{
  {
    LightModelServer_GetOpcodeTableCb,
    LightModelServer_GetStatusRequestCb,
    LightModelServer_ProcessMessageCb
  },
};
__attribute__((aligned(4)))
const Appli_Light_cb_t LightAppli_cb =
{
 /* Light Lightness callbacks */
  Appli_Light_Lightness_Set,
  Appli_Light_Lightness_Status,
  Appli_Light_Lightness_Linear_Set,
  Appli_Light_Lightness_Linear_Status,
  Appli_Light_Lightness_Default_Set,
  Appli_Light_Lightness_Default_Status,
  Appli_Light_Lightness_Last_Set,
  Appli_Light_Lightness_Range_Set,
  Appli_Light_Lightness_Range_Status,
  Appli_Light_Ctl_Set,
  Appli_Light_Ctl_Status,
  Appli_Light_CtlTemperature_Set,
  Appli_Light_CtlTemperature_Status,
  Appli_Light_CtlTemperature_Range_Set,
  Appli_Light_CtlTemperature_Range_Status,
  Appli_Light_CtlDefault_Set,
  Appli_Light_CtlDefault_Status,
  Appli_Light_Hsl_Set,
  Appli_Light_Hsl_Status,
  Appli_Light_HslHue_Set,
  Appli_Light_HslHue_Status,
  Appli_Light_HslSaturation_Set,
  Appli_Light_HslSaturation_Status,
  Appli_Light_HslDefault_Set,
  Appli_Light_HslDefault_Status,
  Appli_Light_HslRange_Set,
  Appli_Light_HslRange_Status
};

  /* Callbacks used by BlueNRG-Mesh Models */
 result = BluenrgMesh_SetSIGModelsCbMap(Model_SIG_cb, MODEL_SIG_COUNT);
  if(MOBLE_FAILED(result))
  {
    TRACE_I(TF_INIT,"BluenrgMesh models init failed \r\n");
    while (1)
    {
      Appli_LedBlink();
    }
  }
```

The `Model_SIG_cb` structure is used to initialize the SIG models for the application implementation. The `BluenrgMesh_SetSIGModelsCbMap(Model_SIG_cb, MODEL_SIG_COUNT);` function is used to initialize the different callbacks in the library.

2.5.2 Initialization and main application loop

This procedure develops an application for mesh over Bluetooth® Low Energy on the BlueNRG platforms.

Step 1. Call the `InitDevice()` API, which calls the `SystemInit()` API, to initialize the device vector table, interrupt priorities, and clock.

Step 2. Call the `Appli_CheckBdMacAddr()` API to check the validity of the MAC address.

If the MAC address is not valid, the firmware is stuck in the `while(1)` loop and the LED continuously blinks.

Step 3. Initialize the hardware callback functions for the Bluetooth® Low Energy hardware by updating `MOBLE_USER_BLE_CB_MAP user_ble_cb =`.

```
{
Appli_BleStackInitCb,
Appli_BleSetTxPowerCb,
Appli_BleGattConnectionCompleteCb,
Appli_BleGattDisconnectionCompleteCb,
Appli_BleUnprovisionedIdentifyCb,
Appli_BleSetUUICb,
Appli_BleSetProductInfoCb,
Appli_BleSetNumberOfElementsCb,
Appli_BleDisableFilterCb
};
```

Step 4. To rely on an application interface for the Bluetooth® Low Energy radio initialization and Tx power configuration, initialize the GATT connection and disconnection callbacks for the application interface.

Step 5. Call `BluenrgMesh_BleHardwareInitCallBack(&user_ble_cb)` to complete the initialization of hardware callbacks.

Step 6. Initialize the BlueNRG-Mesh library by calling `BluenrgMesh_Init(&BLEMeshlib_Init_params).`

If an error occurs, a message ("Could not initialize BlueNRG-Mesh library!") pops up on the terminal window, which was opened for the VCOM port created by the board USB connection. This error makes the LED blink continuously.

Step 7. Check whether the device has been provisioned or not.

A provisioned device has network keys and other parameters configured in the internal flash memory. You can check them with the `BluenrgMesh_IsUnprovisioned()` API. If the node is unprovisioned, the `BluenrgMesh_InitUnprovisionedNode()` API initializes it. If the device is already provisioned, the `BluenrgMesh_InitprovisionedNode()` API helps to initialize the device.

Step 8. Print the messages to the terminal window for the nodes that are being initialized.

The message also prints the MAC address assigned to the node.

Step 9. Initialize the BlueNRG-Mesh models using the `BluenrgMesh_ModelsInit()` API.

Step 10. To initialize the node to the unprovisioned state, hold down the user button.

It erases all the network parameters configured in the device internal memory. Once the unprovisioning is complete, reset the board.

Step 11. Initialize the LED drivers and the GPIO mounted on the X-NUCLEO-LED12A1.

The application must call `BluenrgMesh_Process()` in `while(1)` loop as frequently as possible. This function calls `BLE_StackTick()` internally to process Bluetooth® Low Energy communication. The `BluenrgMesh_ModelsProcess()` (model processing) and `Appli_Process() APIs` are also called in `while(1)` loop.
Any application implementation is performed in the state machine by nonblocking functions with frequent calls to `BluenrgMesh_Process()`.

Step 12. Initialize the ALS drivers and the GPIO mounted on the X-NUCLEO-6283A1.

Process `MX_X_CUBE_ALS_Process()` is called in `while(1)` loop after the ALS is initialized and provisioning of node is done.

The refresh rate of ALS is kept as ~5sec.

For other nodes in the mesh network which do not use ALS expansion board X-NUCLEO-6283A1, `MX_X_CUBE_ALS_Process()`, will not be called.

Step 13. After flashing the SW, if the green user LED blinks twice on the NUCLEO-L476RG (with ALS expansion board) it denotes the setup is in HSL mode where the color on LED driver can be changed depending on the light color exposed on ALS sensor.

If the green user LED blinks once on the NUCLEO-L476RG (with ALS expansion board) it denotes the setup is in brightness mode where the intensity of light on LED driver can be changed depending on the light exposed on ALS sensor. Color cannot be changed in this mode.

The mode can be toggled by a single press on the blue user button on NUCLEO-L476RG.

Step 14. Check for user inputs or buttons for any action to take.

2.5.3 GATT connection/disconnection node

Each node in the network can connect to a smartphone through the GATT interface. When this connection is established, the node becomes a proxy, which acts as a bridge between the mesh network commands and the smartphone responses.

You can detect the smartphone connection and disconnection through the following callbacks:

- `Appli_BleGattConnectionCompleteCb;`
- `Appli_BleGattDisconnectionCompleteCb;`

These are initialized during the main loop.

During the provisioning, the GATT connection is established with the node that needs to be provisioned.

Note: If the smartphone moves out of the proxy node range, it establishes a new connection with the available node.

2.5.4 Lighting model

The specification defines the number of light states, messages, and models that are explicitly defined to be nonspecific in their functionality.

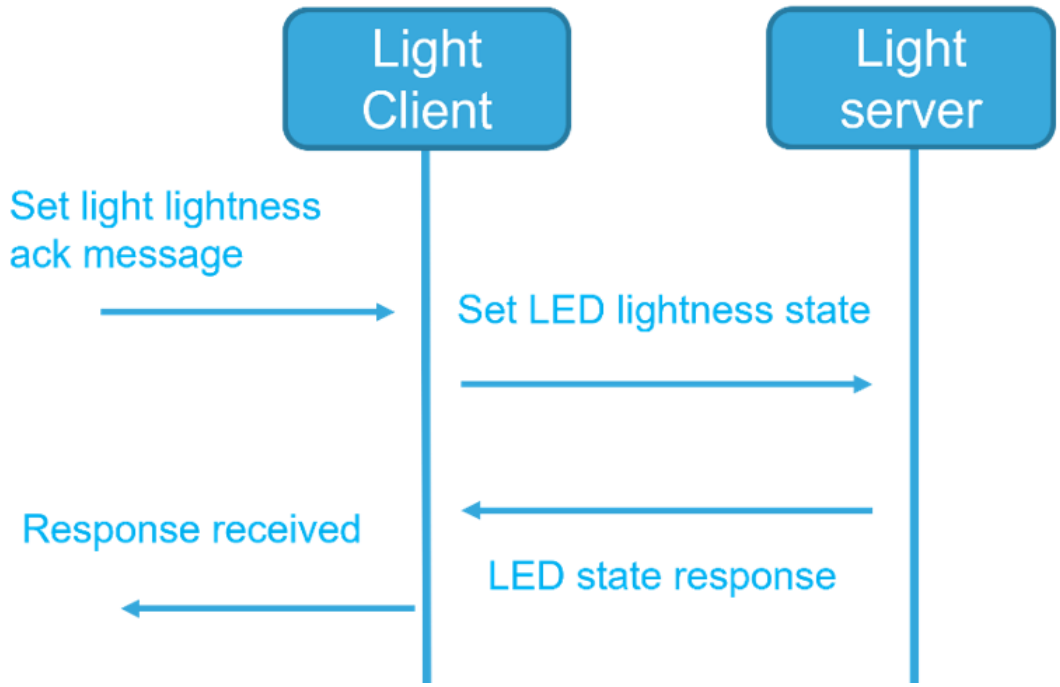
There are different types of light sources with different capabilities. Accordingly, there are different ways to express the state of a light.

A more advanced method of controlling a light is changing the lightness by controlling the light lightness actual state.

If a light is a tunable white, it is possible to control its color temperature through the light CTL.

If a light is a color-changing light, it is possible to control the three dimensions (hue, saturation, and lightness) by controlling each state independently.

Figure 3. **Lighting model message flow**



The number of octets depends on the parameters dedicated for the model. They are different for each lighting model.

The middle layer receives messages from the library. It then checks for the opcode according to the different application of the light model. As an example of the light lightness model, the opcode is checked in the middle layer. The message with the defined data parameters is then transmitted to the light lightness application.

The types of messages are:

- `Set Acknowledged message`, sent by the client to set the desired value to the model on the server. It expects then the response message from the server.
- `Set Unacknowledged message`, sent by the client to set the desired value to the model on the server. It does not expect any response message from the server.
- `Get message`, sent by the client to the server to get the state of the model as a response message from the server.

2.5.5 External MAC address utilities

The "Utilities" folder contains the STM32L4_MAC folder, which provides a hex file of an external MAC address.

To use this address, uncomment the `EXTERNAL_MAC_ADDR_MGMT` macro in the mesh_cfg.h file of the "Middleware" folder.

The demo application firmware and the MAC address are flashed independently. Thus, you do not have to update the firmware if the other firmware has already been flashed.

The MAC address is flashed the first time and at every full chip erase.

3 System setup guide

3.1 Hardware description

3.1.1 STM32 Nucleo

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

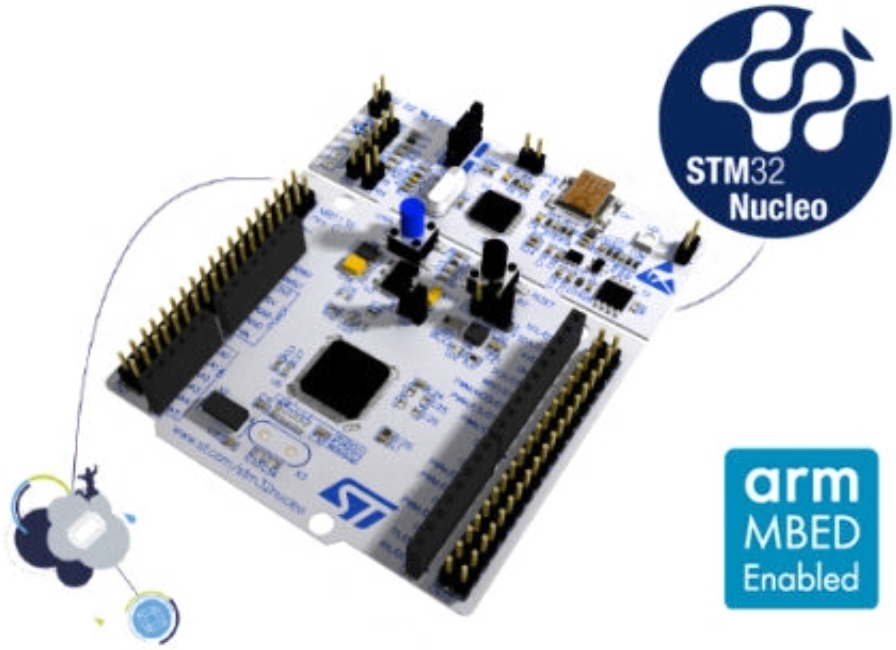
The Arduino connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

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 for different IDEs (IAR EWARM, Keil MDK-ARM, STM32CubeIDE, mbed and GCC/ LLVM).

All STM32 Nucleo users have free access to the mbed online resources (compiler, C/C++ SDK and developer community) at www.mbed.org to easily build complete applications.

Figure 4. STM32 Nucleo board



3.1.2 X-NUCLEO-IDB05A2 expansion board

The X-NUCLEO-IDB05A2 Bluetooth® Low Energy expansion board is based on the BlueNRG-M0 Bluetooth® Low Energy network processor module.

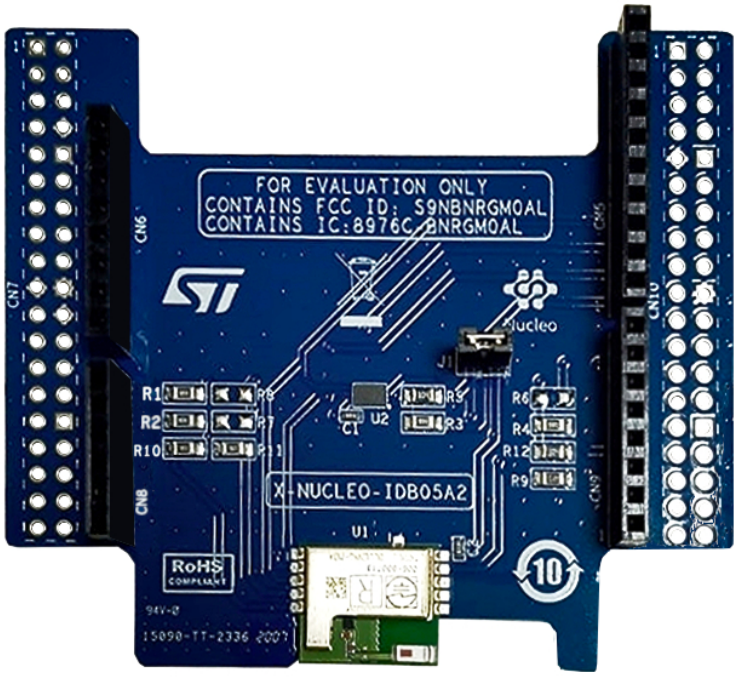
The BlueNRG-M0 is Bluetooth v4.2 compliant, FCC, and IC certified (FCC ID: S9NBNRGM0AL; IC: 8976C-BNRGM0AL). It supports simultaneous master/slave roles and can behave as a Bluetooth® Low Energy sensor and hub device at the same time.

The BlueNRG-M0 provides a complete RF platform in a tiny form factor, with integrated radio, antenna, high frequency, and LPO oscillators.

The X-NUCLEO-IDB05A2 is compatible with the ST morpho (not mounted) and Arduino UNO R3 connector layout.

The X-NUCLEO-IDB05A2 interfaces with the STM32 microcontroller via the SPI pin and allows changing the default SPI clock, SPI chip select, and SPI IRQ by replacing a resistor on the expansion board.

Figure 5. X-NUCLEO-IDB05A2 expansion board



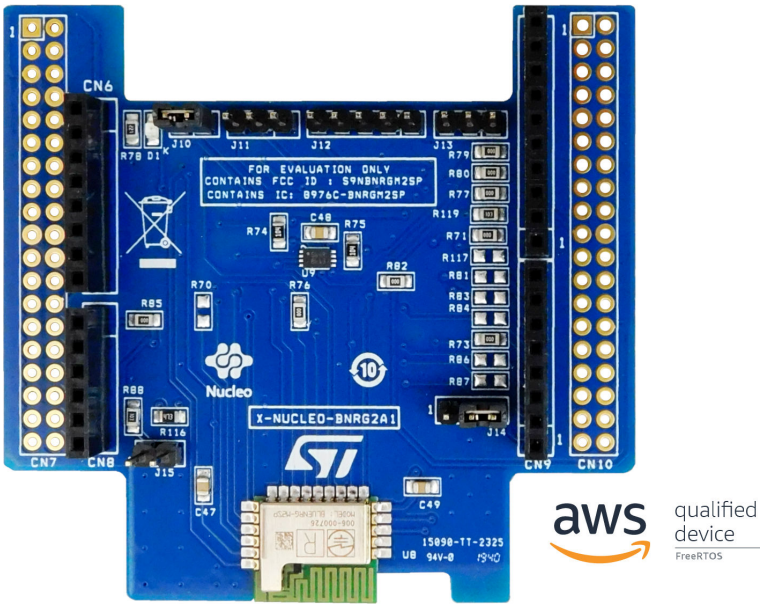
3.1.3 X-NUCLEO-BNRG2A1 expansion board

The X-NUCLEO-BNRG2A1 expansion board provides Bluetooth® Low Energy connectivity for developer applications and can be plugged onto an STM32 Nucleo development board (for example, NUCLEO-L476RG with an ultra-low power STM32 microcontroller) through its Arduino UNO R3 connectors.

The expansion board features the Bluetooth® v5.2 compliant and FCC certified BlueNRG-M2SP application processor module based on the ST BlueNRG-2 System-on-Chip. This SoC manages the complete Bluetooth® Low Energy stack and protocols on its Cortex-M0 core and programmable flash memory, which can accommodate custom applications developed using the SDK. The BlueNRG-M2SP module supports master and slave modes, increased transfer rates with data length extension (DLE), and AES-128 security encryption.

The X-NUCLEO-BNRG2A1 interfaces with the STM32 Nucleo microcontroller via SPI connections and GPIO pins, some of which can be configured through the hardware.

Figure 6. X-NUCLEO-BNRG2A1 expansion board



3.1.4 X-NUCLEO-LED12A1 expansion board

The X-NUCLEO-LED12A1 LED driver expansion board for STM32 Nucleo features four LED1202 devices that can drive up to 48 LEDs.

The LED1202 is a 12-channel low quiescent current LED driver, which guarantees a 5 V output driving capability. Each channel is able to provide up to 20 mA with a headroom voltage of 350 mV (typical) only.

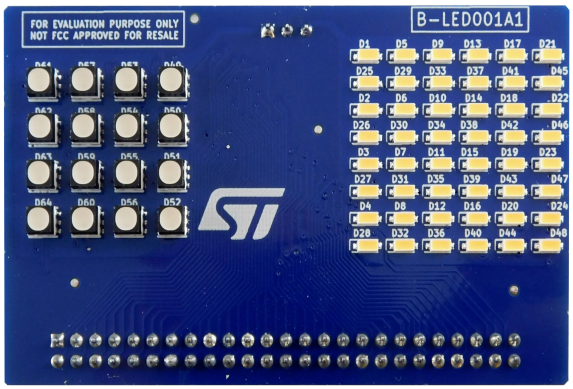
The output current can be adjusted separately for each channel through an 8-bit analog and 12-bit digital dimming control.

The X-NUCLEO-LED12A1 expansion board comes with an additional LED panel board that houses two LEDs matrices: a 6x8 white LED matrix and a 4x4 RGB matrix.

LED matrices can be supplied via an external power supply, which is connected to the J13 connector, and by selecting the right path through the J15 jumper to reach the maximum luminosity available.

Figure 7. X-NUCLEO-LED12A1 expansion board

Figure 8. LED panel board



3.1.5 X-NUCLEO-6283A1 expansion board

The X-NUCLEO-6283A1 is an expansion board for the STM32 Nucleo development boards.

The VD6283TX (1.83 x 1.0 x 0.55 mm) is the smallest 6-channel, ambient light sensor (ALS) on the market.

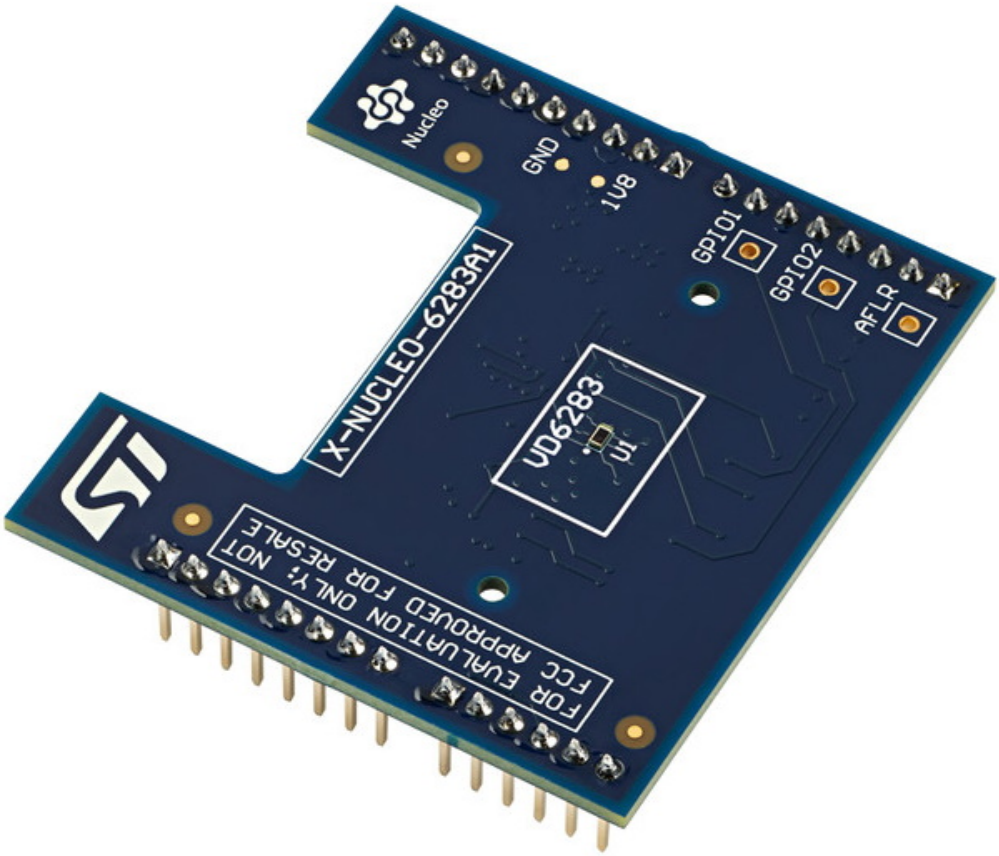
Light measurement is fast and accurate thanks to an individual ADC and readout circuitry for each color channel (Red, Green, Blue, IR, Clear, and Visible).

The VD6283TX uses hybrid color filters with precise responses allowing accurate computation of the correlated color temperature (CCT) and Lux information.

The VD6283TX can be used for display brightness management or scene light correction.

The VD6283TX can extract light flickering frequencies from 100 Hz to 2 kHz, including LED square signals.

Figure 9. X-NUCLEO-6283A1 expansion board



3.2 Hardware setup

To set up a suitable development environment for creating applications for the STM32 Nucleo equipped with the lighting or Bluetooth® Low Energy expansion board, you need the following hardware components:

1. One STM32 Nucleo development board (order code: NUCLEO-L476RG)
2. One Bluetooth® Low Energy expansion board (order code: X-NUCLEO-IDB05A2 or X-NUCLEO-BNRG2A1)
3. One LED expansion board (order code: X-NUCLEO-LED12A1)
4. One ALS expansion board (order code: X-NUCLEO-6283A1)
5. One USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

3.3 Software setup

The following software components are required for the setup of a suitable development environment to create applications for the STM32 Nucleo board with the Bluetooth® Low Energy and the LED expansion board:

- FP-LIT-BLEMESH1: an STM32Cube function pack for IoT node with Bluetooth® Low Energy mesh connectivity and lighting model. The firmware and related documentation are available on www.st.com.
- Development tool-chain and compilers. The STM32Cube expansion software supports the three following environments to select from:
  - IAR Embedded Workbench for Arm® (IAR-EWARM) toolchain + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM-STM32) toolchain + ST-LINK
  - STM32CubeIDE +ST-LINK

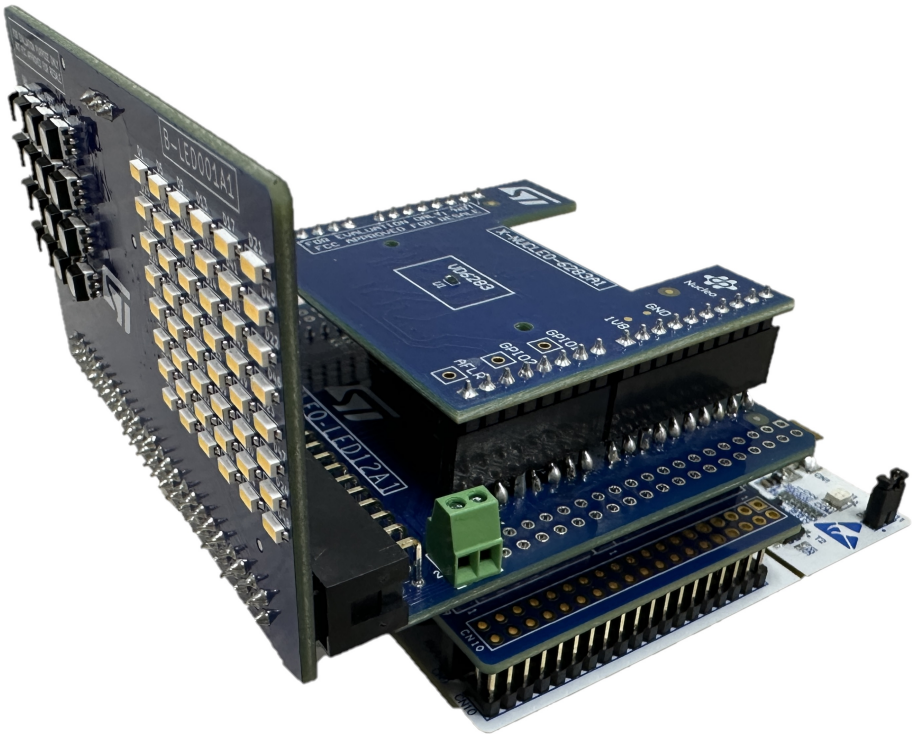
3.4 System setup

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 for the STSW-LINK009 software on www.st.com.

You can easily connect the X-NUCLEO-LED12A1 LED expansion board to the STM32 Nucleo through the Arduino UNO R3 extension connector.

Figure 10. Hardware setup for FP-LIT-BLEMESH1



The X-NUCLEO-LED12A1 can interface with the external STM32 microcontroller on the STM32 Nucleo using the I²C communication protocol.

You can connect the X-NUCLEO-6283A1 ALS expansion board on top of X-NUCLEO-LED12A1 through the Arduino UNO R3 extension connector. Connector CN8 should not be extended on X-NUCLEO-LED12A1.

You can also connect either the X-NUCLEO-IDB05A2 or the X-NUCLEO-BNRG2A1 expansion board to the STM32 Nucleo through the Arduino UNO R3 extension connector.

Appendix A References

1. Mesh over Bluetooth® Low Energy: STSW-BNRG-Mesh
2. Bluetooth mesh networking specifications: https://www.bluetooth.com/specifications/mesh-specifications
3. Bluetooth mesh model specification: https://www.bluetooth.com/specifications/adopted-specifications

Revision history

Table 2. **Document revision history**

Date	Revision	Changes
25-Feb-2022	1	Initial release.
21-Jun-2023	2	Added reference to X-NUCLEO-6283A1. Updated Section Introduction, Section 2.1 Overview, Section 2.2 Architecture, Section 2.3 Folder structure, Section 2.5.1 Initialization of application callbacks, Section 2.5.2 Initialization and main application loop, Section 3.4 System setup. Added Section 3.1.5 X-NUCLEO-6283A1 expansion board.

Contents

List of tables

List of figures