Getting started with X-CUBE-53L9A1: Direct ToF lidar sensor software expansion for STM32Cube

Introduction

This document describes how to get started with the X-CUBE-53L9A1 software expansion for the STM32Cube.

The X-CUBE-53L9A1 provides a complete software package for the STM32, enabling the development of applications that use the VL53L9CX dToF 3D lidar sensor. Thanks to the STM32Cube, it is easily portable across different STM32 MCU families. The package includes sample applications that demonstrate how to perform ranging acquisition and data processing.

The software provides implementation examples for the STM32 Nucleo boards equipped with the X-NUCLEO-53L9A1 expansion board, which features the VL53L9CX sensor.

It is based on the STM32Cube technology and extends the STM32Cube-based packages.

1 STM32Cube

1.1 Overview

The STM32Cube initiative was created by STMicroelectronics to help developers reduce development effort, time, and cost. The STM32Cube covers the STM32 portfolio.

The STM32Cube version 1.x includes:

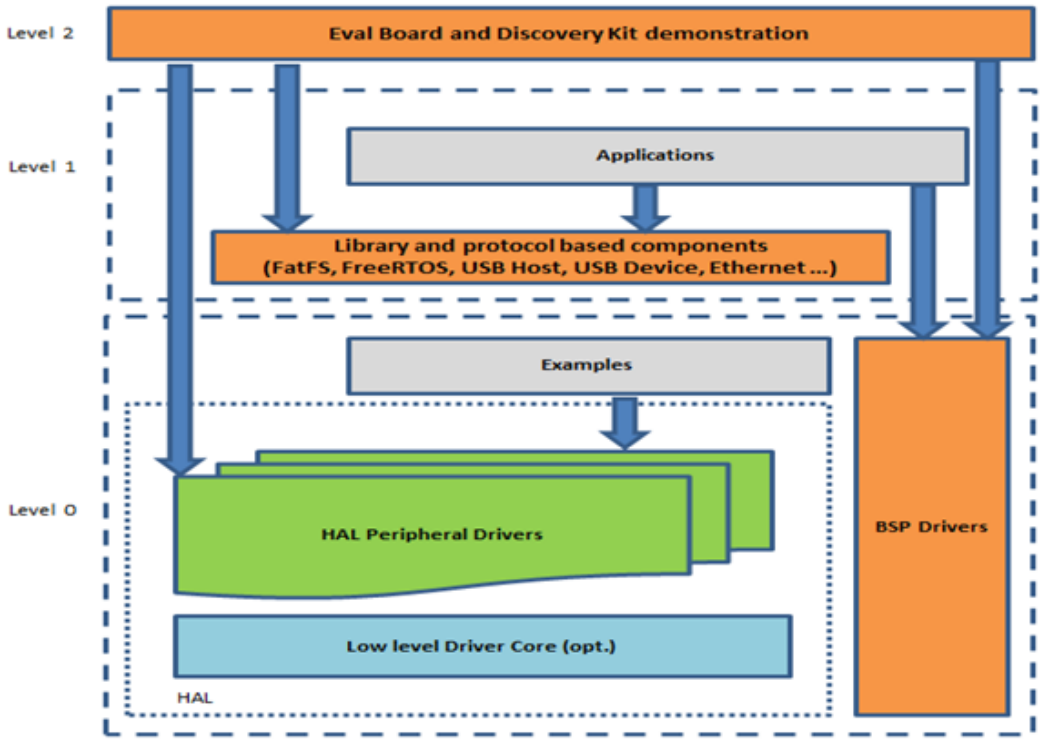
- STM32CubeMX, a graphical configuration tool that generates C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per STM32 series (for example, the STM32CubeF4 for the STM32F4 series), including:
  - STM32 HAL, a hardware abstraction layer that improves portability across the STM32 portfolio.
  - A consistent set of middleware components such as RTOS, USB, TCP/IP, and graphics.
  - Embedded software utilities with a full set of examples.

Refer to www.st.com for information about the STM32Cube.

1.2 Architecture

The STM32Cube firmware solution is structured around three independent levels that interact easily with one another, as described in the figure below. These levels are level 0, level 1, and level 2.

Figure 1. Firmware architecture



Level 0

This level is divided into three sublayers:

- **Board support package (BSP):** This layer provides a set of APIs for the hardware components on the board, such as the audio codec, I/O expander, touchscreen, SRAM driver, and LCD drivers. It consists of two parts:
  - Component: The driver is associated with an external device on the board and not specific to the STM32. The component driver provides specific APIs to the BSP driver for external components and can be ported to any other board.
  - BSP driver: This links the component driver to a specific board and provides a set of easy-to-use APIs. The API naming convention is: BSP_<function>_<action>(). For example, BSP_LED_Init() and BSP_LED_On().

  This modular architecture makes it easy to port the BSP to any hardware by implementing only the low-level routines.
- **Hardware abstraction layer (HAL):** This layer provides low-level drivers and hardware interfacing methods to interact with the upper layers, including application code, libraries, and stacks. It provides generic, multi-instance, function-oriented APIs that simplify application development by offering ready-to-use processes. For communication peripherals such as I2S and UART, it provides APIs for initialization, configuration, data transfer, and error handling using polling, interrupts, or DMA. HAL driver APIs are divided into two categories:
  - Generic APIs, which provide common functions across all STM32 series.
  - Extension APIs, which provide specific or customized functions for a particular family or device.
- **Basic peripheral usage examples:** This layer includes examples built around STM32 peripherals using only HAL and BSP resources.

Level 1

This level is divided into two sublayers:

- **Middleware components:** A set of libraries covering USB host and device libraries, STemWin, FreeRTOS™, FatFS, LwIP, and PolarSSL. Horizontal interactions between components in this layer are handled directly by calling feature APIs. Vertical interaction with low-level drivers is managed through specific callbacks and static macros implemented in the library system call interface. For example, FatFS implements a disk I/O driver to access a microSD™ card or the USB mass storage class.
- **Examples based on middleware components:** Each middleware component comes with one or more examples. These are called applications, and they show how to use the middleware. Integration examples using several middleware components are also provided.

Level 2

This level consists of a single layer that provides global real-time and graphical demonstrations. These demonstrations are based on the middleware service layer, the low-level abstraction layer, and the basic peripheral usage applications for board-level functionalities.

2 X-CUBE-53L9A1 software expansion for STM32Cube

2.1 Overview

The X-CUBE-53L9A1 software package extends the STM32Cube functionality.

Its key features are:

- Complete software for building applications that capture and process data from the VL53L9CX sensor.
- Application examples demonstrating the innovative technology and accurate distance-ranging capabilities.
- A sample implementation available on the X-NUCLEO-53L9A1 board plugged into the NUCLEO-H563ZI board.
- Easy portability across different MCU families thanks to the STM32Cube.
- Free, user-friendly license terms.

2.2 Architecture

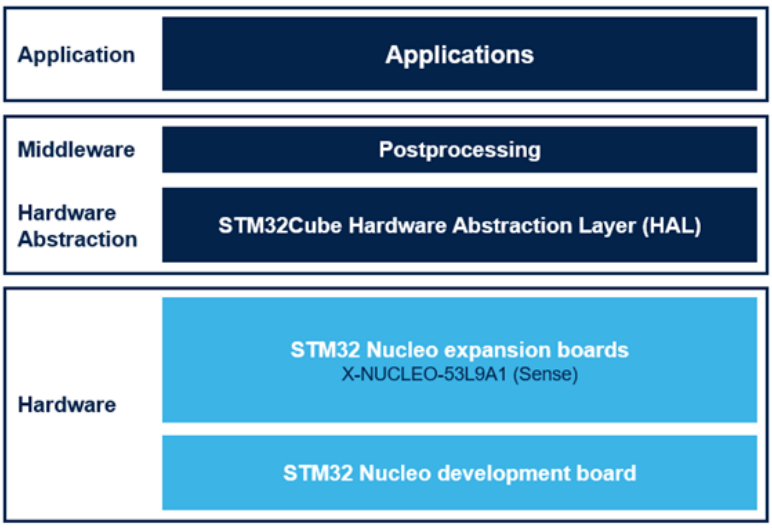
This software extends the STM32Cube and is based on the STM32Cube HAL, the hardware abstraction layer for STM32 microcontrollers. The package extends the STM32Cube by providing a BSP for the sensor expansion board, middleware components for data processing, and sample applications for serial communication with a PC.

The software layers used by the application software to access and use the sensor expansion board are as follows:

- **STM32Cube HAL layer:** This layer provides a simple, generic, multi-instance set of APIs to interact with the upper layers, including the application code, libraries, and stacks. It includes both generic and extension APIs. This generic architecture enables higher layers, such as the middleware layer, to implement their functionalities independently of the specific hardware configuration of a given MCU. This structure improves code reusability and ensures high portability across devices.
- **BSP layer:** This layer provides support for the peripherals on the STM32 Nucleo board, except for the MCU. It offers APIs that provide a programming interface for certain board-specific peripherals, such as the LED and the user button. It also allows the specific board version to be identified.
- **Middleware:** This layer provides advanced postprocessing libraries to apply calibration data, perform normalization, apply optical corrections, remove noise, and compute the confidence level.

The following figure shows the software architecture of the package.

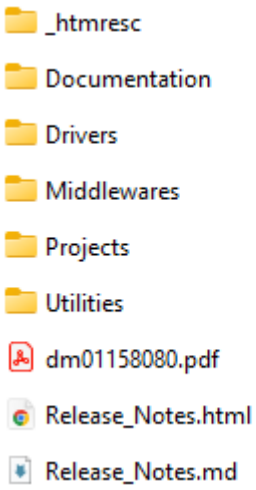
Figure 2. X-CUBE-53L9A1 software architecture



2.3 Folder structure

This section provides an overview of the package folder structure. The following figure outlines the package architecture.

Figure 3. X-CUBE-53L9A1 package folder structure



The software package includes the following folders:

- **Documentation**: Contains a compiled HTML file generated from the source code, as well as detailed documentation for the software components and APIs.
- **Drivers**: Contains the HAL drivers, board-specific drivers for each supported board or hardware platform, and drivers for on-board components. It also includes the CMSIS layer, a vendor-independent hardware abstraction layer for the Cortex®-M processor series.
- **Middleware**: Contains the libraries used to process the data coming from the sensor.
- **Projects**: Contains sample applications for the supported STM32 platforms, demonstrating how to retrieve sensor data using three development environments: IAR Embedded Workbench for Arm, RealView microcontroller development kit (MDK-Arm), and STM32CubeIDE.
- **Utilities**: Contains common software used across several applications and demonstrations.

2.4 APIs

You can find detailed technical information about the APIs available to the user in the compiled HTML file located in the “Documentation” folder of the software package. It fully describes all functions and parameters.

2.5 53L9A1 single postprocessing example

This application demonstrates single-sensor postprocessing with the VL53L9CX ToF 3D lidar. It captures raw measurement frames from the sensor over the I3C, applies the transform pipeline using the device calibration data, and generates a processed depth stream on the STM32 host.

The acquisition data are read asynchronously using DMA, which helps reduce CPU load. A double-buffering mechanism is used so that the previous frame can be processed while the current frame is being transmitted.

The example can be customized to output other available streams. At application startup, the serial port displays the available controls and stream capabilities of the postprocessing pipeline. For a complete description of the library capabilities, refer to the *vl53l9-transform-c* documentation.

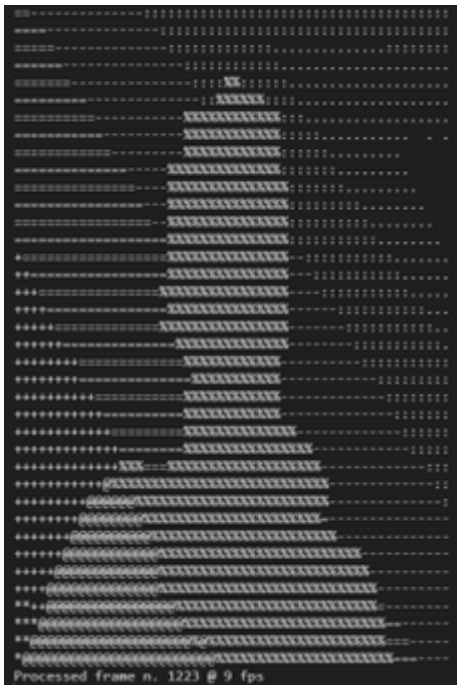
Some symbols are defined and can be overridden to customize the application behavior:

- CONF_DEVICE_ID: Defines which VL53L9 instance to use. It should be updated only in case of multiple devices (see vl53l9_device.c).
- CONF_PRINT_FRAME: Enables printing of the frame mapping depth value using ASCII characters. Please note that this affects performance.
- CONF_USECASE: Selects the ranging profile to be applied (see vl53l9_utils.h). Ranging profiles are predefined device configurations, such as exposure time, binning factor, and power mode. It is still possible to override some parameters through the component driver, depending on your application's needs.

The serial port must be configured with the following settings:

- Baud rate: 115200
- Data bits: 8
- Parity: None
- Stop bits: 1

Figure 4. Serial console output



3 Setup and configuration

3.1 Hardware description

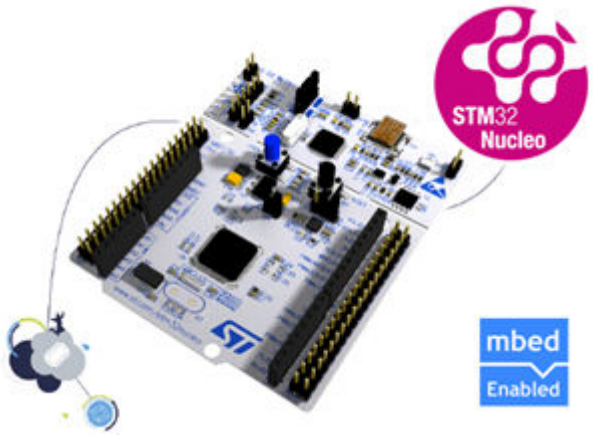
This section provides an overview of the hardware components required for developing a sensor-based application. The following subsections describe each component in detail.

3.1.1 STM32 Nucleo board

The STM32 Nucleo development boards provide an affordable and flexible way to test solutions and build prototypes with any STM32 microcontroller line. They support Arduino™ connectivity and ST morpho connectors, making it easy to expand the STM32 Nucleo open development platform with a wide range of specialized expansion boards.

The STM32 Nucleo board does not require a separate probe, because it integrates the ST-LINK/V2-1 debugger/ programmer. It also includes the STM32 HAL library and several packaged software examples.

Figure 5. STM32 Nucleo board



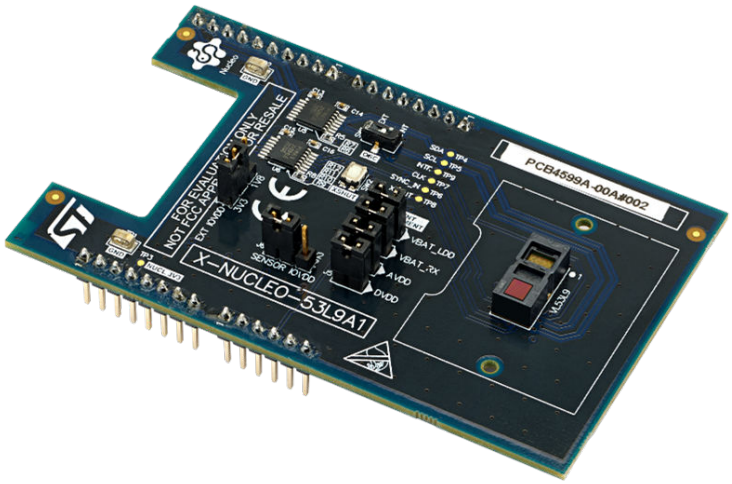
Refer to www.st.com for information about STM32 Nucleo boards.

3.1.2 X-NUCLEO-53L9A1 expansion board

The X-NUCLEO-53L9A1 is a sensor expansion board designed for use with the STM32 Nucleo system. It is also compatible with the Arduino® UNO R3 connector layout. The board is based on the STMicroelectronics VL53L9CX dToF 3D lidar.

The X-NUCLEO-53L9A1 interfaces with the STM32 MCU through the SDA and SCL connections. The device can be shut down through a dedicated pin.

Figure 6. **X-NUCLEO-53L9A1 sensor expansion board**



Refer to www.st.com for information about the X-NUCLEO-53L9A1 expansion board
The following table summarizes the hardware connections.

Table 1. **Hardware connections**

Pin	Name	Description	Notes
D15	SCL	I²C/I3C SCL line	—
D14	SDA	I²C/I3C SDA line	—
D0	INTR	Interrupt pin	Falling edge
D1	XSHUT	Shutdown pin	Reversed polarity
D11	CLK_IN	Clock input pin	When SW1 is set to EXT
A3	SYNC_IN	Synchronization input pin	When sensor is in follower mode

Perform the following steps to obtain a valid hardware configuration:

- Make sure the EXT IOVDD jumper (J1) matches the STM32 Nucleo VDD setting (JP4 on NUCLEO-H563ZI).

Figure 7. STM32 Nucleo VDD configuration (JP4)

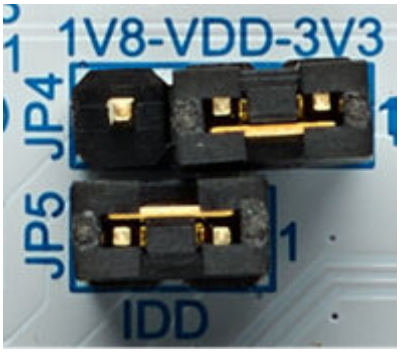


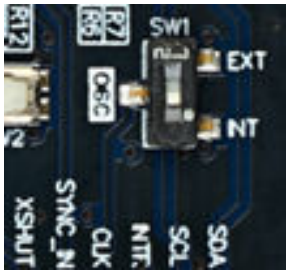
Figure 8. EXT IOVDD jumper configuration (J1)

- Set the SENSOR IOVDD jumper (J6) to 1V8 (top).

Figure 9. SENSOR IOVDD jumper configuration (J6 to 1V8)

- Set SW1 to INT (bottom) to connect AP_CLK to the 12 MHz onboard oscillator.

Figure 10. SW1 configuration to connect AP_CLK to the onboard oscillator



3.2 Software description

The following software components are required to set up the development environment for creating applications for the STM32 Nucleo board equipped with the sensor expansion board:

- **X-CUBE-53L9A1:** An STM32Cube expansion package dedicated to sensor application development. The X-CUBE-53L9A1 firmware and related documentation are available on www.st.com.
- **Development toolchains and compilers:** The STM32Cube expansion software supports the following development environments:
  - IAR Embedded Workbench for Arm (EWARM) with ST-LINK
  - RealView microcontroller development kit (MDK-Arm) with ST-LINK
  - STM32CubeIDE with ST-LINK

3.3 Hardware and software setup

This section describes the hardware and software setup procedures, as well as the system configuration required for the steps described above.

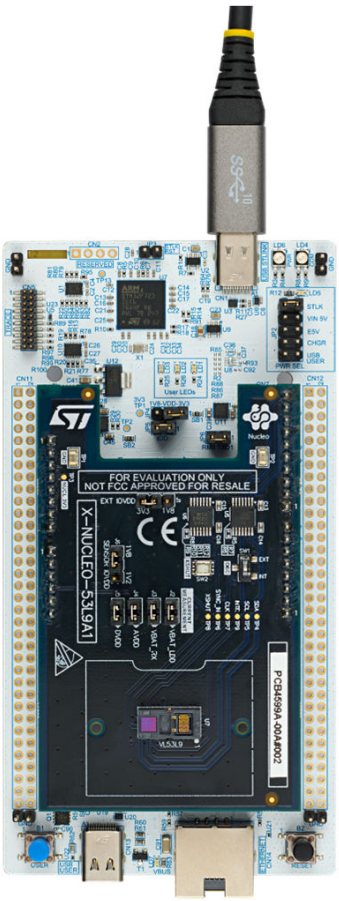
3.3.1 Hardware setup

The following hardware components are required:

1. One STM32 Nucleo development board (suggested order code: NUCLEO-H563ZI)
2. One X-NUCLEO-53L9A1 expansion board
3. One USB cable to connect the STM32 Nucleo board to the PC

Note: ST recommends using a USB 3 Type-C to Type-C cable.

Figure 11. X-NUCLEO-53L9A1 expansion board plugged into NUCLEO-H563ZI



3.3.2 Software setup

This section lists the minimum requirements for developers to customize, build, and flash the sample applications.

3.3.2.1 Development toolchains and compilers

Select one of the integrated development environments supported by the STM32Cube expansion software. Then refer to the system requirements and setup information provided by the selected IDE vendor.

3.3.2.2 Utilities

The utilities folder contains shared code, helpers, and utilities used across several applications. In particular, it includes:

- **A platform abstraction layer** that provides services such as power control, event management, profiling, and communication support for I²C and I3C interfaces.
- **Common device descriptors** for sensor instances (see `vl53l9_device_t`), which allow configuration of the bus interface, static address, power voltage, clock frequency, and GPIO pins used for interrupt and shutdown. When updating the peripheral configuration on the STM32 or using custom hardware, these device descriptors should be adapted to match your setup.
- **Utility helpers** for configuring sensor use cases and ranging profiles, as well as computing output resolution.

It is important that the device descriptor settings are aligned with the hardware configuration of the X-NUCLEO-53L9A1 expansion board, in particular the following parameters:

- **.vdda** must be set to 2V8
- **.vddio** must be set according to the configuration of SENSOR IOVDD (J6)
- **.ext_clock** must be set according to the configuration of SW1
  - **INT**: The onboard oscillator generates the clock at 12 MHz.
  - **EXT**: The host generates the clock (12.5 MHz) — the firmware does not support it yet.

Note: When making hardware changes or using custom hardware, the device descriptor configuration must be aligned accordingly.

3.3.3 Board setup

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

3.3.3.1 STM32 Nucleo and sensor expansion board setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. The ST-LINK/V2-1 USB driver is available in the STSW-LINK009 software package on www.st.com.

The X-NUCLEO-53L9A1 sensor expansion board connects to the STM32 Nucleo board through the Arduino® UNO R3 extension connector. It communicates with the STM32 microcontroller over the I3C transport layer.

4 Acronyms and abbreviations

Table 2. Acronyms and abbreviations

Acronym/abbreviation	Description
API	application programming interface
BSP	board support package
CMSIS	Cortex® microcontroller software interface standard
CPU	central processing unit
DMA	direct memory access
HAL	hardware abstraction layer
I²C	inter-integrated circuit
I2S	inter-integrated circuit sound
I3C	improved inter-integrated circuit
IDE	integrated development environment
I/O	input/output
LCD	liquid crystal display
LED	light-emitting diode
MCU	microcontroller unit
SCL	serial clock
SDA	serial data
SRAM	static random-access memory
ToF	Time-of-Flight
UART	universal asynchronous receiver transmitter
USB	universal serial bus

Revision history

Table 3. **Document revision history**

Date	Version	Changes
28-Apr-2026	1	Initial release
28-Apr-2026	2	Changed document status.
06-May-2026	3	Changed document status.

Contents

1 **STM32Cube** 2
1.1 Overview 2
1.2 Architecture 2
2 **X-CUBE-53L9A1 software expansion for STM32Cube** 4
2.1 Overview 4
2.2 Architecture 4
2.3 Folder structure 5
2.4 APIs 5
2.5 53L9A1 single postprocessing example 6
3 **Setup and configuration** 7
3.1 Hardware description 7
3.1.1 STM32 Nucleo board 7
3.1.2 X-NUCLEO-53L9A1 expansion board 7
3.2 Software description 10
3.3 Hardware and software setup 10
3.3.1 Hardware setup 10
3.3.2 Software setup 11
3.3.3 Board setup 11
4 **Acronyms and abbreviations** 12
Revision history 13