UM2371
User manual

Getting started with the X-CUBE-53L1A1 Time-of-Flight long distance ranging sensor software expansion for STM32Cube

Introduction

This document describes how to get started with the X-CUBE-53L1A1 software expansion for STM32Cube.

The X-CUBE-53L1A1 provides a complete embedded software example for the STM32 to build a simple application with the VL53L1X Time-of-Flight (ToF) sensor using the X-NUCLEO-53L1A1 expansion board. It is portable across different MCU families thanks to the STM32Cube technology.

This user manual package contains an example of the application which enables, in real time, transmission of the sensor ranging data to a PC.

Contents

1 Acronyms and abbreviations used in UM2371

Table 1. Acronyms and abbreviations

Acronym	Description
I2C	Inter-integrated circuit (serial bus)
NVM	Non volatile memory
SPAD	Single photon avalanche diode
VCSEL	Vertical cavity surface emitting laser
PAL	Photonic abstraction layer
API	Application programming interface
FMT	Final module test
XTALK	Crosstalk

2 What is STM32Cube ?

2.1 STM32Cube overview

The STM32Cube™ initiative covers the STM32 portfolio. This initiative was originated by STMicroelectronics to ease the life of developer's by reducing development effort, time, and cost.

The STM32Cube version 1.x includes:

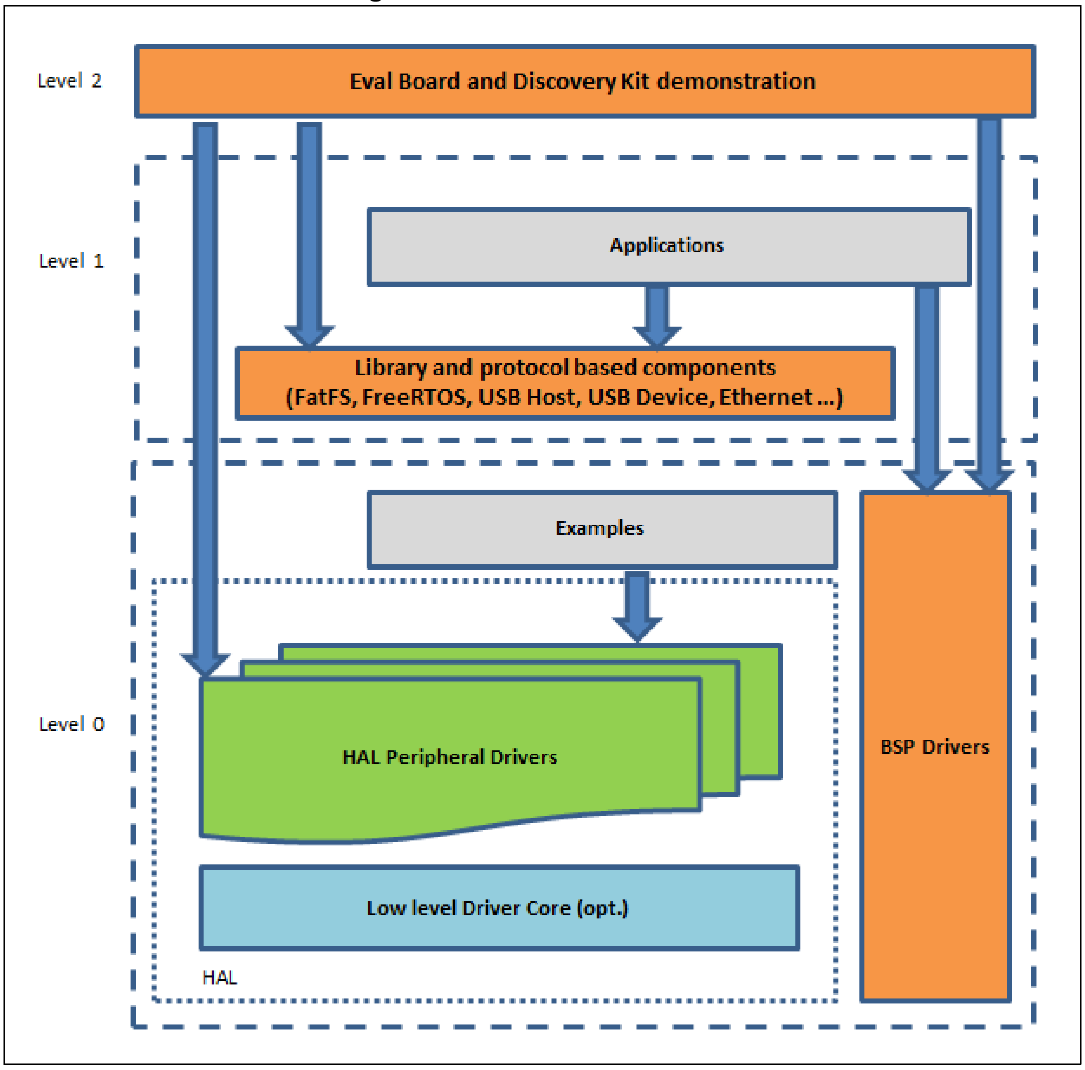
- The STM32CubeMX, a graphical software configuration tool that allows the generatation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as the STM32CubeF4 for the STM32F4 series).
  - The STM32Cube HAL, an STM32 abstraction layer embedded software, that ensures maximum portability across the STM32 portfolio.
  - A consistent set of middleware components including RTOS, USB, TCP/IP, and graphics.
  - All embedded software utilities that come with a full set of examples.

Information about the STM32Cube is available at http://www.st.com/stm32cube

2.2 STM32Cube architecture

The STM32Cube firmware solution is built around three independent levels that can easily interact with each other as shown in the figure 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
BSP Drivers
Low level Driver Core (opt.)
HAL

1. Level 0, Level 1, and Level 2 are explained in *Section 2.2.1*, *Section 2.2.2*, and *Section 2.2.3*.

2.2.1 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 composed of two parts:

- Component: this is the driver relative to the external device on the board. It is not related to the STM32. The component driver provides specific APIs to the BSP driver external components and can be ported to any other board.
- BSP driver: this permits linking of the component driver i.e. the VL53L1X API can be linked to a specific board and can provide a set of user-friendly APIs.

This layer is based on modular architecture which allows easy porting to any hardware by simply implementing the low level routines.

Hardware abstraction layer (HAL)

This layer provides low level drivers and hardware interfacing methods to interact with the upper layers (application, libraries, and stacks). It also provides a generic, multi-instance, functional oriented API which permits the user application implementation to be offloaded through a ready-to-use process. For example, regarding the communication peripherals (I2S, UART, etc.), this layer provides APIs that allow:

- Initialization and configuration of the specific peripheral
- Data management transfer based on polling
- Interruption or DMA processing
- Management of communication errors that may arise during communication

The HAL driver API's are split into two categories:

- Generic APIs which provide common and generic functions to the whole STM32 series
- Extension APIs which provide specific and customized functions for a specific family or a specific part number.

Basic peripheral use examples

This layer encloses the examples built over the STM32 peripheral using only the HAL and BSP resources.

2.2.2 Level 1

This level is divided into two sub-layers.

Middleware components

This is a set of libraries covering the USB host and device libraries, STemWin, FreeRTOS, FatFS, LwIP, and PolarSSL. The horizontal interactions between the components of this layer are made directly by calling the feature APIs. The vertical interactions with the low level drivers are made 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 the microSD drive or the USB mass storage class.

Examples based on the Middleware components

Each Middleware component comes with one or more examples (called “Applications”) which show how to use the component. Integration examples that use several Middleware components are also provided.

2.2.3 Level 2

This level is composed of a single layer which is a global real-time and graphical demonstration based on the:

- Middleware service layer
- Low level abstraction layer
- Basic peripheral use application for board based functionalitiesCube

3 X-CUBE-53L1A1 software expansion for STM32Cube

3.1 Overview

The X-CUBE-53L1A1 is a software package that expands the functionality provided by the STM32Cube.

The key features of the package are:

- Easy portability across different MCU families thanks to the STM32Cube
- Sample application to transmit real-time sensor data to a PC via a serial interface
- Free user-friendly license terms
- Implementation examples available on the X-NUCLEO-53L1A1 board, plugged on top of a NUCLEO-F401RE or a NUCLEO-L476RG.

This example software enables the distance measurement based on the ToF VL53L1X long distance ranging sensor and running on the STM32.

3.2 Architecture

This software is an expansion for the STM32Cube. It fully complies with the architecture of the STM32Cube and also expands it to enable development of applications using ST ToF sensors (see *Section 2.1: STM32Cube overview* for an overview of the STM32Cube).

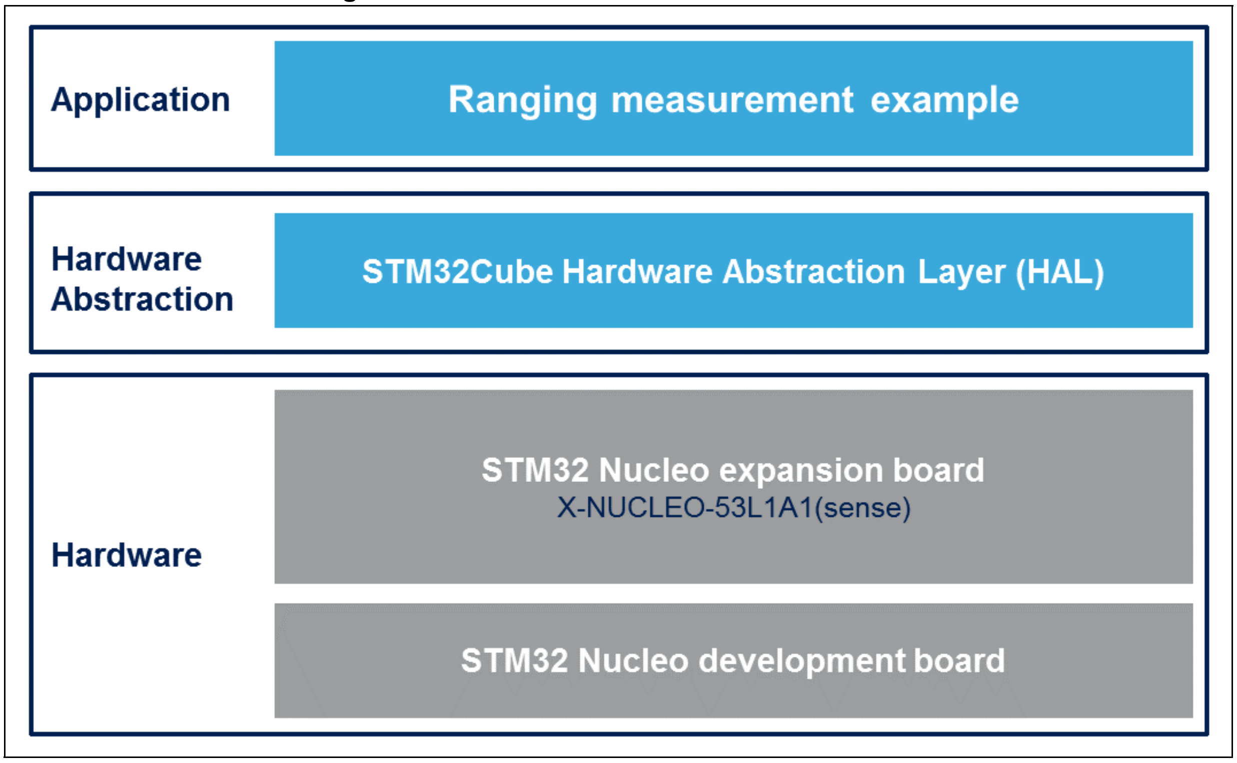
The software is based on the STM32CubeHAL which is the hardware abstraction layer for the STM32 microcontroller. The package extends the STM32Cube by providing a board support package (BSP) for the sensor expansion board.

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

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

Figure 2 shows the architecture of the software package.

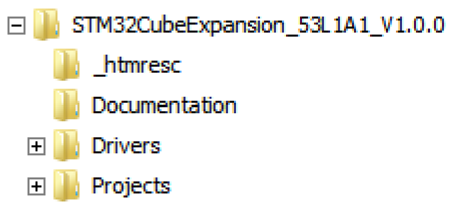
Figure 2. X-CUBE-53L1A1 software architecture


Application
Ranging measurement example
Hardware Abstraction
STM32Cube Hardware Abstraction Layer (HAL)
Hardware
STM32 Nucleo expansion board
X-NUCLEO-53L1A1(sense)
STM32 Nucleo development board

3.3 Folder structure

This section provides an overview of the package folder structure. *Figure 3* shows the high-level architecture of the software package.

Figure 3. X-CUBE-53L1A1 package folder structure



The following folders are included in the software package:

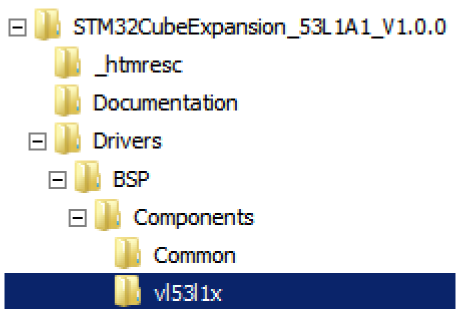
- **Documentation**: this folder contains a compiled HTML file generated from the source code which documents in detail the VL53L1X API.
- **Drivers**: this folder contains:
  - STM32 HAL drivers
  - Board specific drivers for each supported board or hardware platform, including the VL53L1X API and the CMSIS layer which is a vendor-independent hardware abstraction layer for the Cortex-M processor series.
- **Projects**: this folder contains an example of the application used to access sensor data. The example is provided for the NUCLEO-L476RG and NUCLEO-F401RE platforms with three development environments:
  - IAR embedded workbench for ARM
  - Realview microcontroller development kit (MDK-ARM)
  - System workbench for the STM32 (SW4STM32)

3.4 VL53L1X API

The VL53L1X has been designed to be fully portable on any 32-bit microcontroller platform. Detailed technical information about the VL53L1X API can be found in the VL53L1X_API.chm file located inside the "Documentation" folder of the software package (see *Section 3.3: Folder structure*).

The VL53L1X API is located in the "vl53l1x" directory as shown in *Figure 4*.

Figure 4. VL53L1X API directory



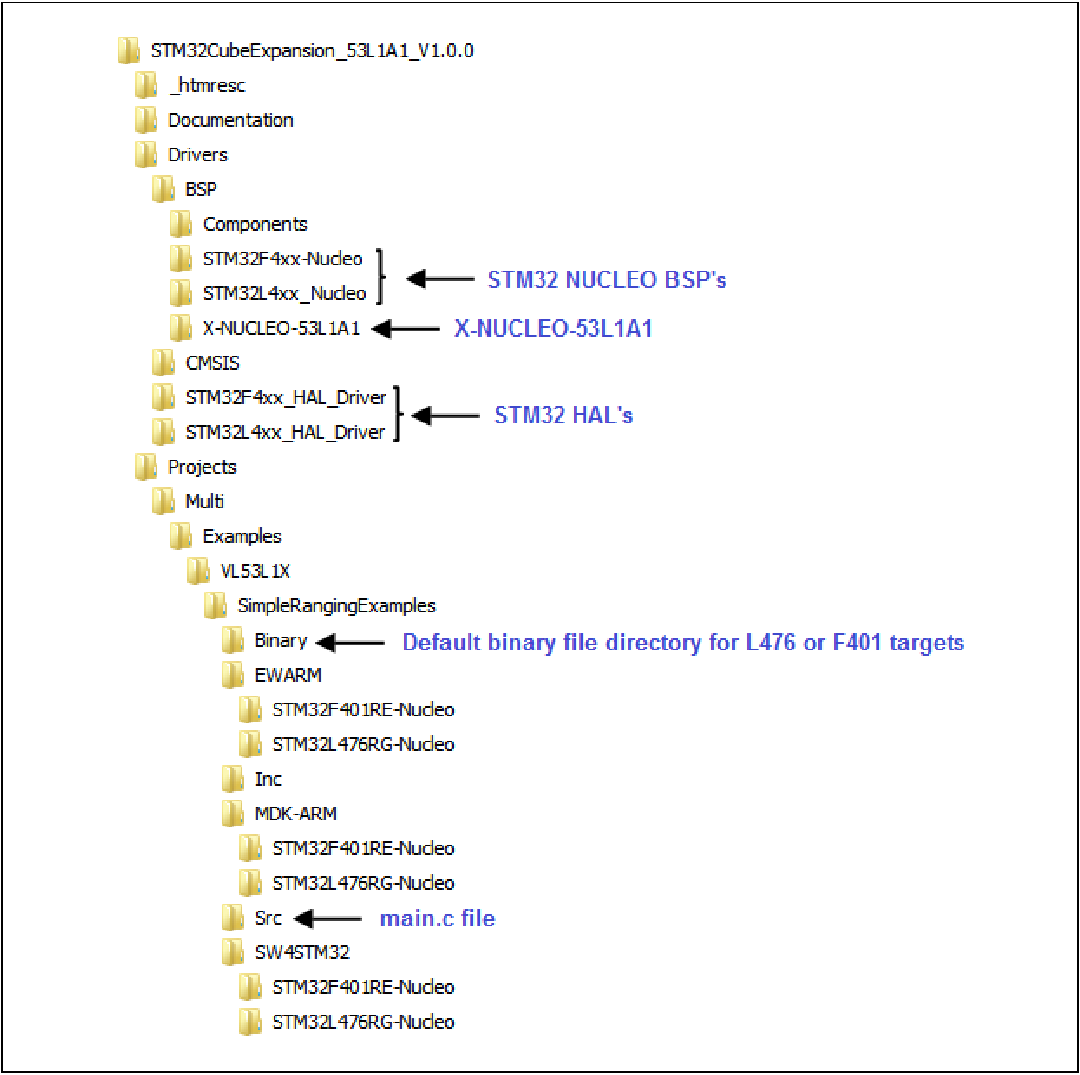
```
⊟ STM32CubeExpansion_53L1A1_V1.0.0
    _htmresc
    Documentation
  ⊟ Drivers
    ⊟ BSP
      ⊟ Components
          Common
          vl53l1x
```

3.5 Example application description

An example application using the X-NUCLEO-53L1A1 expansion board with either the NUCLEO-F401RE or NUCLEO-L476RG boards is provided in the “Projects” directory of the software package (see *Section 3.3: Folder structure*). *Figure 5* shows the directory paths.

Ready-to-build projects are available for multiple integrated drive electronics (IDEs) but also for binary files which can be dragged and dropped into the Nucleo USB mass storage device.

Figure 5. SDK file locations



```
STM32CubeExpansion_53L1A1_V1.0.0
  _htmresc
  Documentation
  Drivers
    BSP
      Components
      STM32F4xx-Nucleo   }  <---- STM32 NUCLEO BSP's
      STM32L4xx_Nucleo   }
      X-NUCLEO-53L1A1  <---- X-NUCLEO-53L1A1
    CMSIS
    STM32F4xx_HAL_Driver }  <---- STM32 HAL's
    STM32L4xx_HAL_Driver }
  Projects
    Multi
      Examples
        VL53L1X
          SimpleRangingExamples
            Binary  <---- Default binary file directory for L476 or F401 targets
            EWARM
              STM32F401RE-Nucleo
              STM32L476RG-Nucleo
            Inc
            MDK-ARM
              STM32F401RE-Nucleo
              STM32L476RG-Nucleo
            Src  <---- main.c file
            SW4STM32
              STM32F401RE-Nucleo
              STM32L476RG-Nucleo
```

In the example application, real-time sensor data are transmitted to a PC via a serial port, using the *printf()* system call. Transmitted sensor data can be viewed using software like Tera Term©, a free PC based application.

The code example describes how to run a single VL53L1X sensor in Single autonomous mode. This is achieved using either an interrupt or Polling mode to receive information that the ranging data are ready or a threshold has been triggered.

In the *main(void*) function, after the usual MCU initializations, there is a call to the *AutonomousLowPowerRangingTest()* function. This function initializes and sets up the sensor. It then starts and sends the measurement values on the serial link/ as follows:

```
status = VL53L1_WaitDeviceBooted(Dev);
status = VL53L1_DataInit(Dev);
status = VL53L1_StaticInit(Dev);
status = VL53L1_SetPresetMode(Dev, VL53L1_PRESETMODE_LOWPOWER_AUTONOMOUS);
status = VL53L1_SetDistanceMode(Dev, VL53L1_DISTANCEMODE_LONG);
status = VL53L1_SetMeasurementTimingBudgetMicroSeconds(Dev, 30000);
status = VL53L1_SetInterMeasurementPeriodMilliSeconds(Dev, 500);
status = VL53L1_StartMeasurement(Dev);
```

Once the measurement values have been sent on the serial link, the VL53L1X enters an infinite ranging loop if no stop is called. Then, it performs the sensor data reading and sends it on the serial link as detailed below:

```
status = VL53L1_GetRangingMeasurementData(Dev, &RangingData);
printf("%d,%d,%.2f,%.2f\n",
RangingData.RangeStatus,RangingData.RangeMilliMeter,
(RangingData.SignalRateRtnMegaCps/65536.0),RangingData.AmbientRateRtnMegaC
ps/65336.0);
```

Different parameters from the API can be rendered to perform a reliable ranging, as referred to in the VL53L1X API user manual (UM2356).

4 System setup guide

4.1 Hardware description

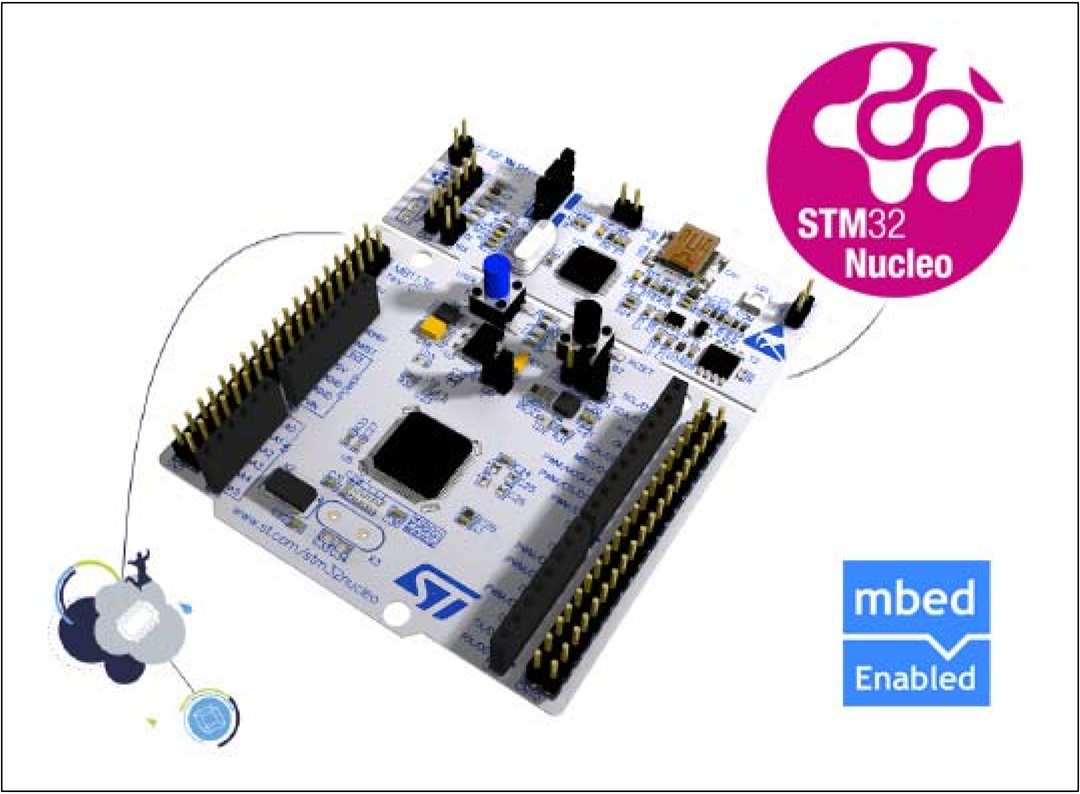
This section describes the hardware components needed for developing a sensor-based application. The sub-sections below describe the individual components.

4.1.1 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 of the 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 www.st.com at: http://www.st.com/stm32nucleo

Figure 6. STM32 nucleo board



4.1.2 X-NUCLEO-53L1A1 expansion board

The X-NUCLEO-53L1A1 expansion board is compatible with the Arduino UNO R3 connector layout. The X-NUCLEO-53L1A1 interfaces with the STM32 MCU via:

- An I2C pin
- A GPIO pin to act as a hardware interrupt flag from the VL53L1X
- Another GPIO pin to shut down the VL53L1X

The user can change the default I2C address by programming it after boot. The VL53L1X is an optical sensor and, consequently, it can be hidden and protected behind a cover window. Such a cover window generates crosstalk that can be compensated by a calibration. To carry out some evaluations of cover window performances and mechanical mountings, the X-NUCLEO-53L1A1 expansion board includes two types of cover window. They are manufactured in PMMA material and have very good transmission rates of 940 nm wavelengths.

The first type of cover window (the rectangular gray cover) can be adjusted in height thanks to different spacers as shown in *Figure 7*. The air gap, i.e. the distance between the sensor package and the cover window, can be created with a space of 0.25 mm, 0.5 mm, 1.0 mm, or a mix of all three heights.

The second type of cover window is a ~0.15 mm air gap cover window, specifically designed for the VL53L1X sensor by Hornix Inc. (http://www.hornix.com.tw). It is the black cap at the bottom of *Figure 7* and can be clipped onto the VL53L1X (see *Figure 8*).

Figure 7. X-NUCLEO-53L1A1 sensor expansion board

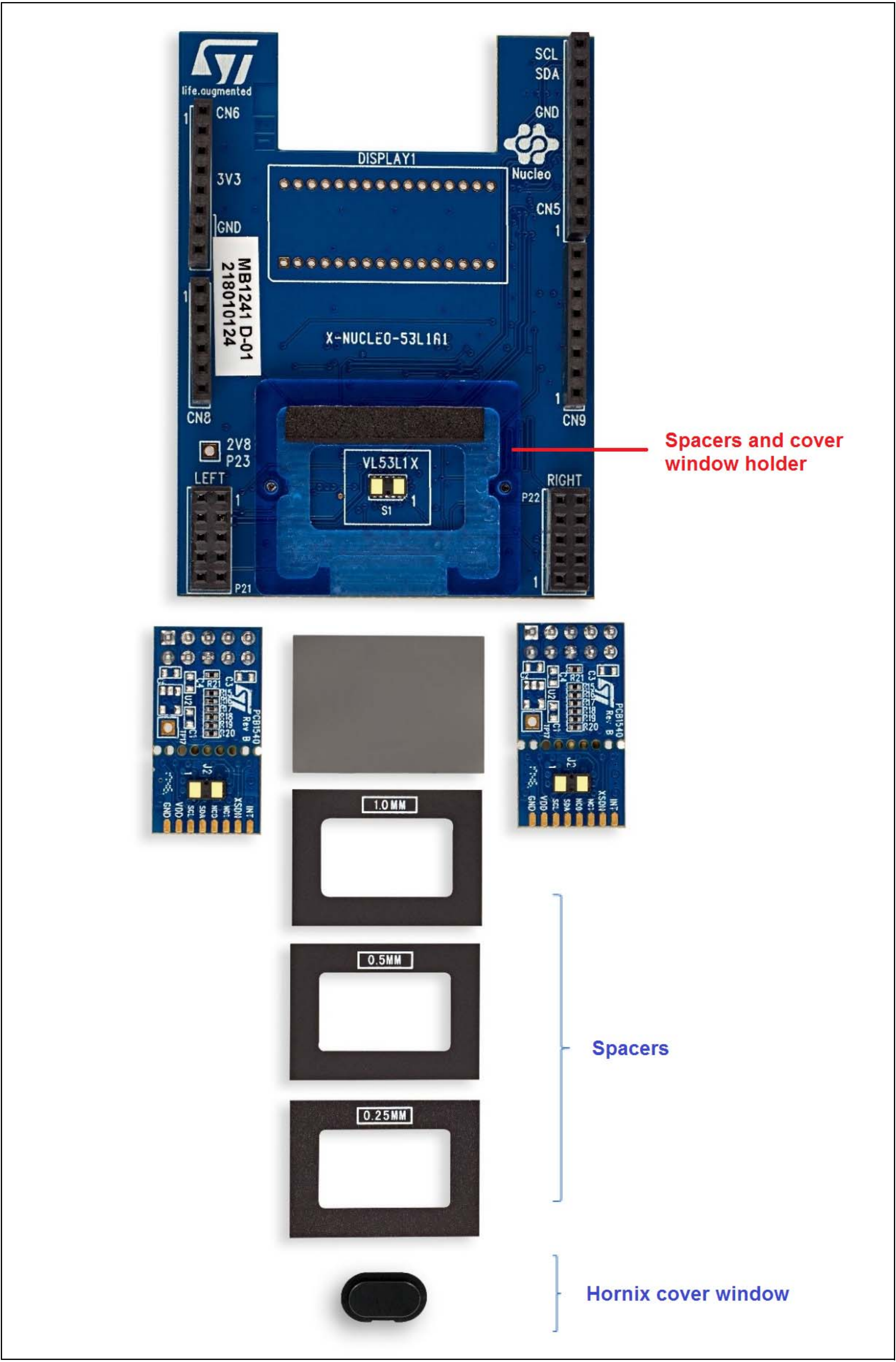
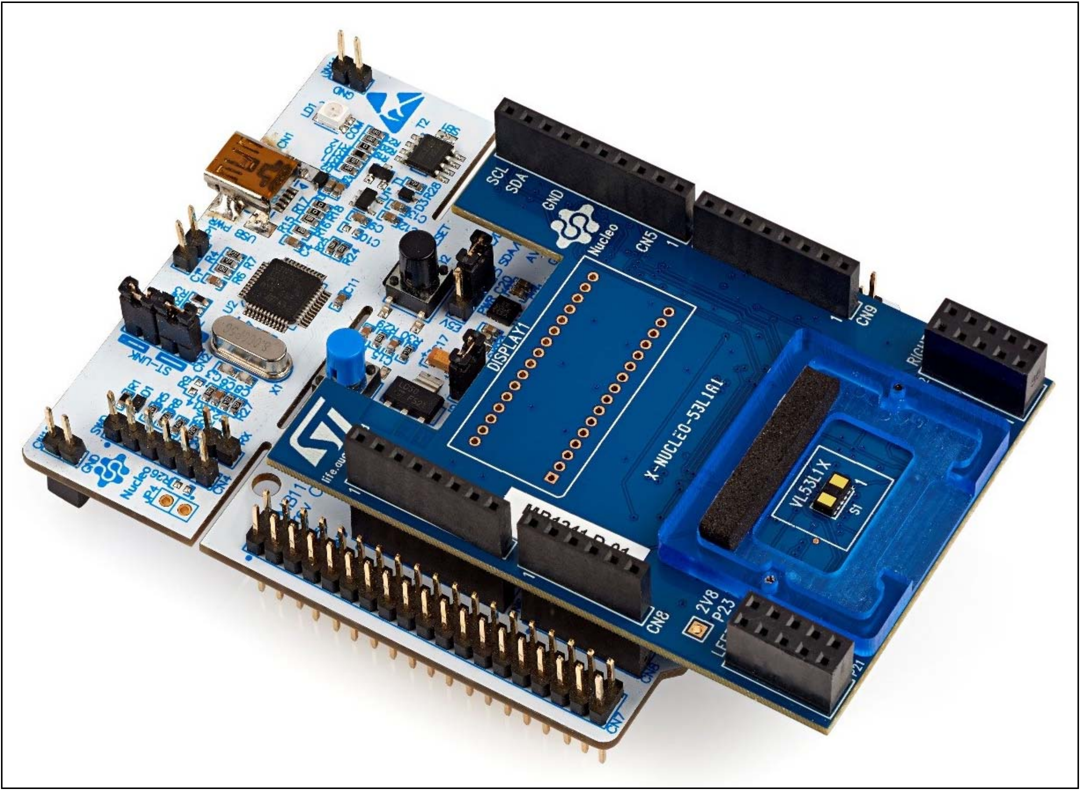


Figure 8. Sensor expansion board connected to STM32 nucleo board



A complete set of documentation (user manuals, quick start guides, etc.) about the X-NUCLEO-53L1A1 expansion board is available on www.st.com.

4.2 Software description

To set up suitable development environments for creating applications for the STM32 Nucleo, which are equipped with expansion boards, the following software components are needed:

- X-CUBE-53L1A1: an example of the embedded software for the STM32Cube which is dedicated to VL53L1X sensor application development.
- Development tool chains and compilers, including:
  - 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

4.3 Hardware setup

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

The following hardware components are needed:

1. One STM32 Nucleo development platform (suggested order code includes either the NUCLEO-F401RE or NUCLEO-L476RG).
2. One sensor expansion board (suggested order code is the X-NUCLEO-53L1A1)
3. One USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

By default, the hardware configuration is set up to enable a single sensor which is positioned at the center of the expansion board. However, the user can range with a left or right ToF sensor by changing the code in the main.c file.

4.4 Software setup

This section outlines the minimum requirements to developers for setting up the SDK and running the example application.

4.4.1 STM32 Nucleo pack software installation

ST delivers an STM32 Nucleo board software suite that allows the user to discover many ST components through standalone demonstrations and a PC graphical user interface (GUI).

The STM32 Nucleo board software suite is available from *www.st.com*. This software is compatible with all STM32 Nucleo boards.

The software suite consists of:

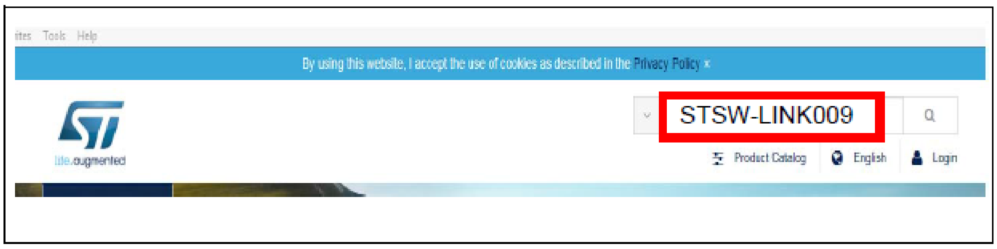
- STSW-LINK009, ST-Link, ST-Link/V2, ST-Link/V2-1 USB driver signed for XP and Windows 7 and 8. This driver must be installed first.
- STSW-LINK007 and ST-Link/V2-1 firmware update driver

When STSW-LINK009 and STSW-LINK007 firmware are installed, the STM32 Nucleo board is configured and ready to use with a PC.

STSW-LINK009: STM32 Nucleo board Windows USB driver installation

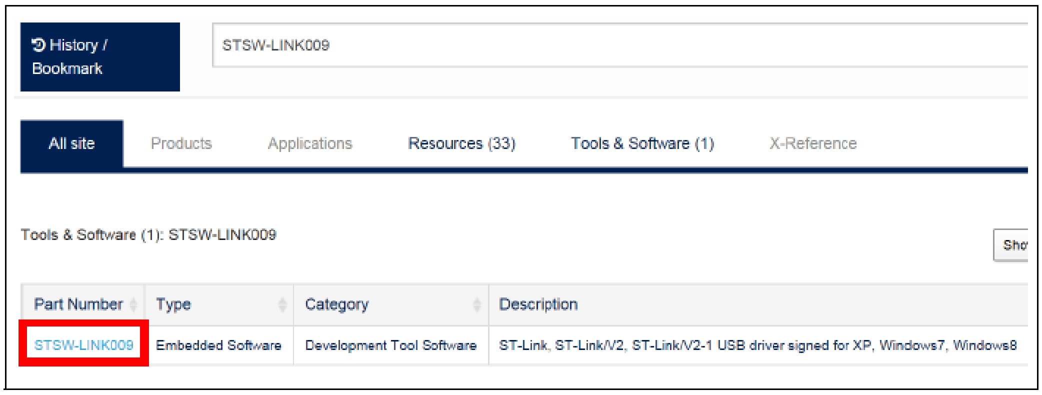
- On *www.st.com*, search for “STSW-LINK009”
- Follow the installation steps in *Figure 9*, *Figure 10*, *Figure 11*, and *Figure 12*

Figure 9. STM32 Nucleo board Windows USB driver installation step 1


ites Tools Help
By using this website, I accept the use of cookies as described in the Privacy Policy ×
life.augmented
STSW-LINK009
Product Catalog
English
Login

- On next page click on “STSW-LINK009”

Figure 10. STM32 Nucleo board Windows USB driver installation step 2

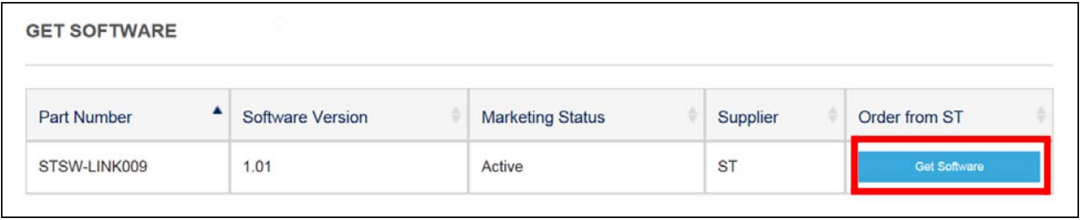


Tools & Software (1): STSW-LINK009

Part Number	Type	Category	Description
STSW-LINK009	Embedded Software	Development Tool Software	ST-Link, ST-Link/V2, ST-Link/V2-1 USB driver signed for XP, Windows7, Windows8

- On next page click on “Get software”

Figure 11. STM32 Nucleo board Windows USB driver installation step 3

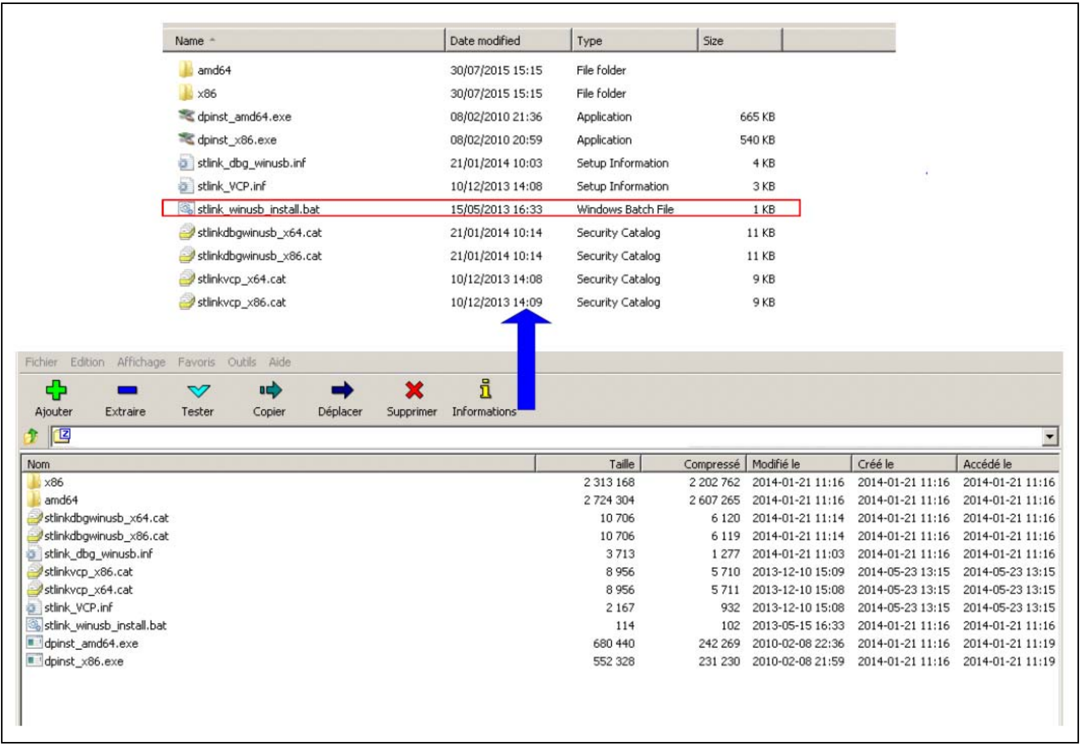


GET SOFTWARE

Part Number	Software Version	Marketing Status	Supplier	Order from ST
STSW-LINK009	1.01	Active	ST	Get Software

- Complete and sign license agreement
- From en.stsw-link009.zip, unpack the.zip file and run stlink_winusb_install.bat. This will install the necessary USB drivers to allow communication between the STM32 Nucleo board and the PC.

Figure 12. STM32 Nucleo board Windows USB driver installation step 4



Name	Date modified	Type	Size
amd64	30/07/2015 15:15	File folder	
x86	30/07/2015 15:15	File folder	
dpinst_amd64.exe	08/02/2010 21:36	Application	665 KB
dpinst_x86.exe	08/02/2010 20:59	Application	540 KB
stlink_dbg_winusb.inf	21/01/2014 10:03	Setup Information	4 KB
stlink_VCP.inf	10/12/2013 14:08	Setup Information	3 KB
stlink_winusb_install.bat	15/05/2013 16:33	Windows Batch File	1 KB
stlinkdbgwinusb_x64.cat	21/01/2014 10:14	Security Catalog	11 KB
stlinkdbgwinusb_x86.cat	21/01/2014 10:14	Security Catalog	11 KB
stlinkvcp_x64.cat	10/12/2013 14:08	Security Catalog	9 KB
stlinkvcp_x86.cat	10/12/2013 14:09	Security Catalog	9 KB

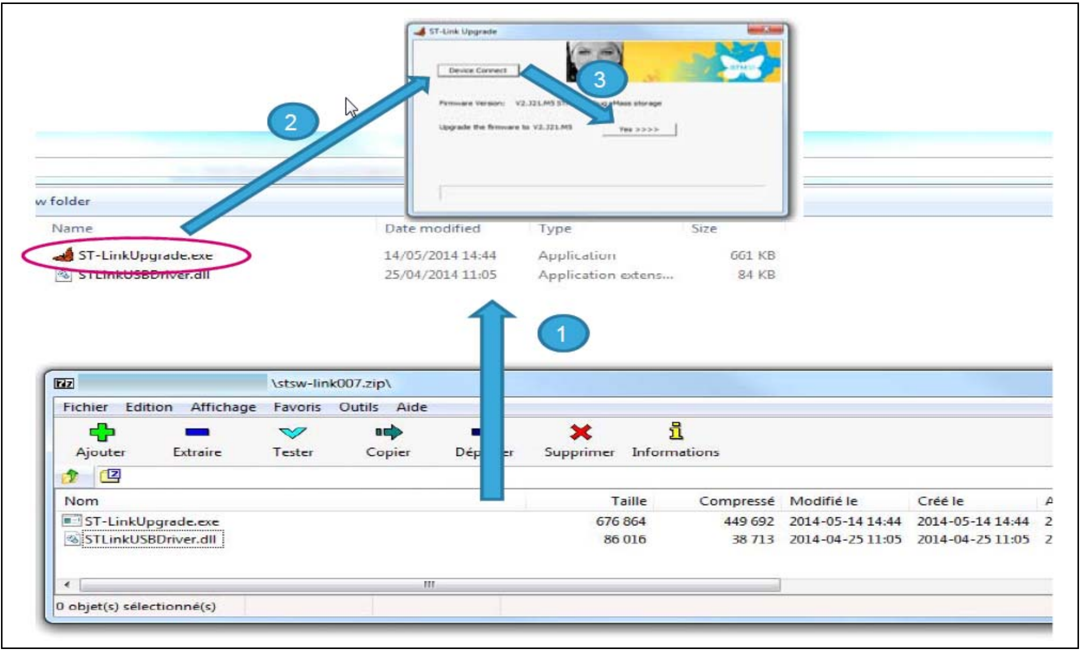
Nom	Taille	Compressé	Modifié le	Créé le	Accédé le
x86	2 313 168	2 202 762	2014-01-21 11:16	2014-01-21 11:16	2014-01-21 11:16
amd64	2 724 304	2 607 265	2014-01-21 11:16	2014-01-21 11:16	2014-01-21 11:16
stlinkdbgwinusb_x64.cat	10 706	6 120	2014-01-21 11:14	2014-01-21 11:16	2014-01-21 11:16
stlinkdbgwinusb_x86.cat	10 706	6 119	2014-01-21 11:14	2014-01-21 11:16	2014-01-21 11:16
stlink_dbg_winusb.inf	3 713	1 277	2014-01-21 11:03	2014-01-21 11:16	2014-01-21 11:16
stlinkvcp_x86.cat	8 956	5 710	2013-12-10 15:09	2014-05-23 13:15	2014-05-23 13:15
stlinkvcp_x64.cat	8 956	5 711	2013-12-10 15:08	2014-05-23 13:15	2014-05-23 13:15
stlink_VCP.inf	2 167	932	2013-12-10 15:08	2014-05-23 13:15	2014-05-23 13:15
stlink_winusb_install.bat	114	102	2013-05-15 16:33	2014-01-21 11:16	2014-01-21 11:16
dpinst_amd64.exe	680 440	242 269	2010-02-08 22:36	2014-01-21 11:16	2014-01-21 11:19
dpinst_x86.exe	552 328	231 230	2010-02-08 21:59	2014-01-21 11:16	2014-01-21 11:19

- Plug a USB cable between the PC and the STM32 Nucleo board. Allow the board driver installations to complete before proceeding.

STSW-LINK007: STM32 Nucleo board PC communication driver

- On *www.st.com*, search for "STSW-LINK007"
- To install STSW-LINK007 repeat steps 1 to 4 performed for the installation of the STSW-LINK009 STM32 Nucleo board Windows USB driver installation.
- Follow the next installation step in *Figure 13* below
- Unpack the downloaded stsw-link007.zip file and run STLinkUpgrade.exe
- Ensure the STM32 Nucleo development board is connected via the USB port
- Click "device connect" on the dialogue and confirm the board has successfully connected.
- When prompted to upgrade to the latest version, check that the suggested version is later than the current Firmware version. Then, click 'YES' to proceed.

Figure 13. STM32 Nucleo board communication driver with PC installation


ST-Link Upgrade
Device Connect
Yes >>>>
2
3
1
w folder
\stsw-link007.zip\
Fichier Edition Affichage Favoris Outils Aide
Ajouter Extraire Tester Copier Déplacer Supprimer Informations
0 objet(s) sélectionné(s)

Name	Date modified	Type	Size
ST-LinkUpgrade.exe	14/05/2014 14:44	Application	661 KB
STLinkUSBDriver.dll	25/04/2014 11:05	Application extens...	84 KB

Nom	Taille	Compressé	Modifié le	Créé le
ST-LinkUpgrade.exe	676 864	449 692	2014-05-14 14:44	2014-05-14 14:44
STLinkUSBDriver.dll	86 016	38 713	2014-04-25 11:05	2014-04-25 11:05

4.4.2 Development tool chains and compilers

- Select one of the IDE's supported by the STM32Cube expansion software
- Read the system requirements and set up the information provided by the selected IDE provider.
- Launch the dedicated start file for your IDE and the project will be installed. Note that some old tool chains may have to be upgraded.

4.4.3 Sensor data log

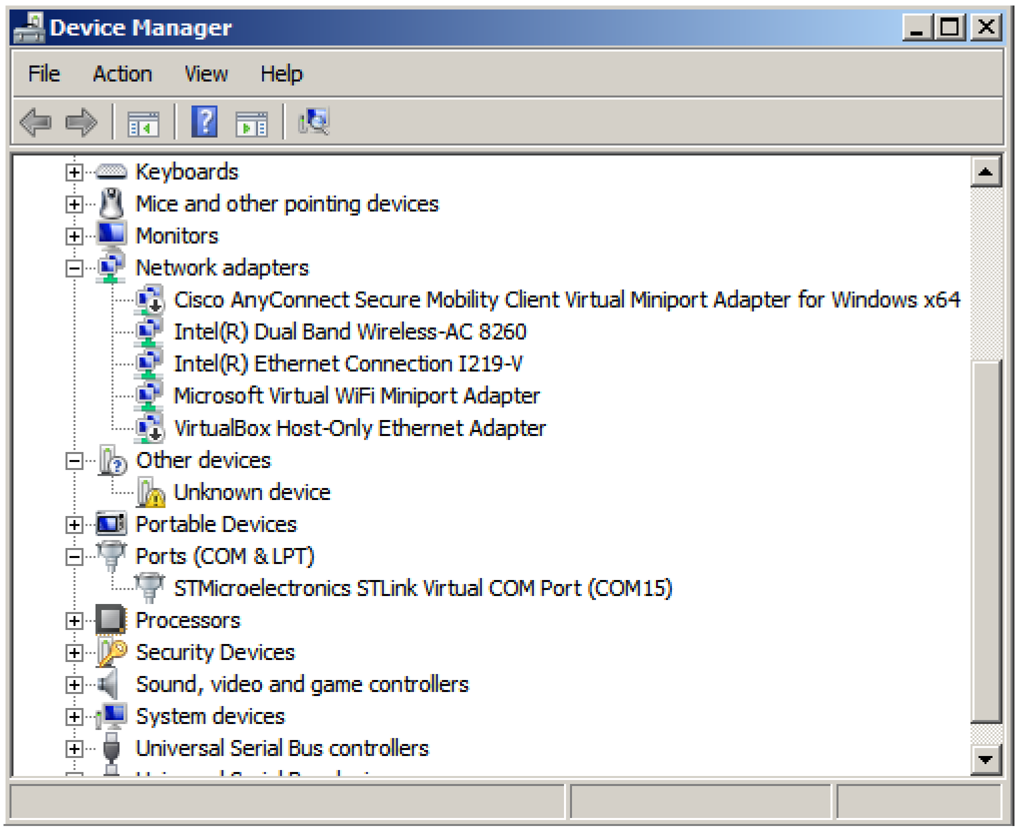
Key ranging parameters can be output from the STM32 Nucleo to the PC through a USB serial connection using any terminal emulator program supporting serial connections such as TeraTerm®, Putty, etc.

For example, the TeraTerm® utility can be downloaded @ http://logmett.com/tera-term-the-latest-version.

Sensor data log installation steps

In the device manager window (see *Figure 14*), get the COM number of STMicroelectronic's STLink virtual COM port which is used by the STM32 Nucleo board.

Figure 14. STMicroelectronic's STLink virtual COM port

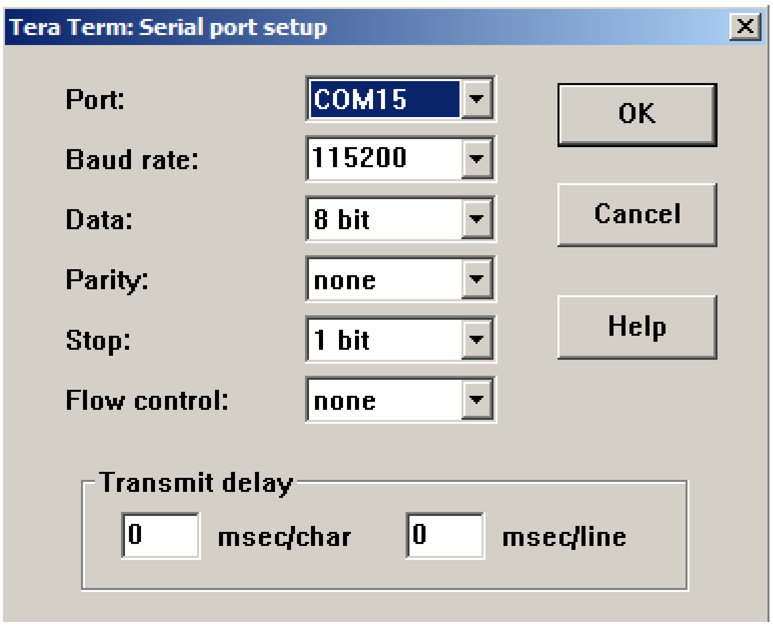


```
Keyboards
Mice and other pointing devices
Monitors
Network adapters
    Cisco AnyConnect Secure Mobility Client Virtual Miniport Adapter for Windows x64
    Intel(R) Dual Band Wireless-AC 8260
    Intel(R) Ethernet Connection I219-V
    Microsoft Virtual WiFi Miniport Adapter
    VirtualBox Host-Only Ethernet Adapter
Other devices
    Unknown device
Portable Devices
Ports (COM & LPT)
    STMicroelectronics STLink Virtual COM Port (COM15)
Processors
Security Devices
Sound, video and game controllers
System devices
Universal Serial Bus controllers
```

If using TeraTerm, open the “New connection” window or tick “Serial” to get a fresh connection. Then chose the right COM port and press OK.

The setup of the serial link is described in *Figure 15*.

Figure 15. Serial port setup



Tera Term: Serial port setup

Port: COM15

Baud rate: 115200

Data: 8 bit

Parity: none

Stop: 1 bit

Flow control: none

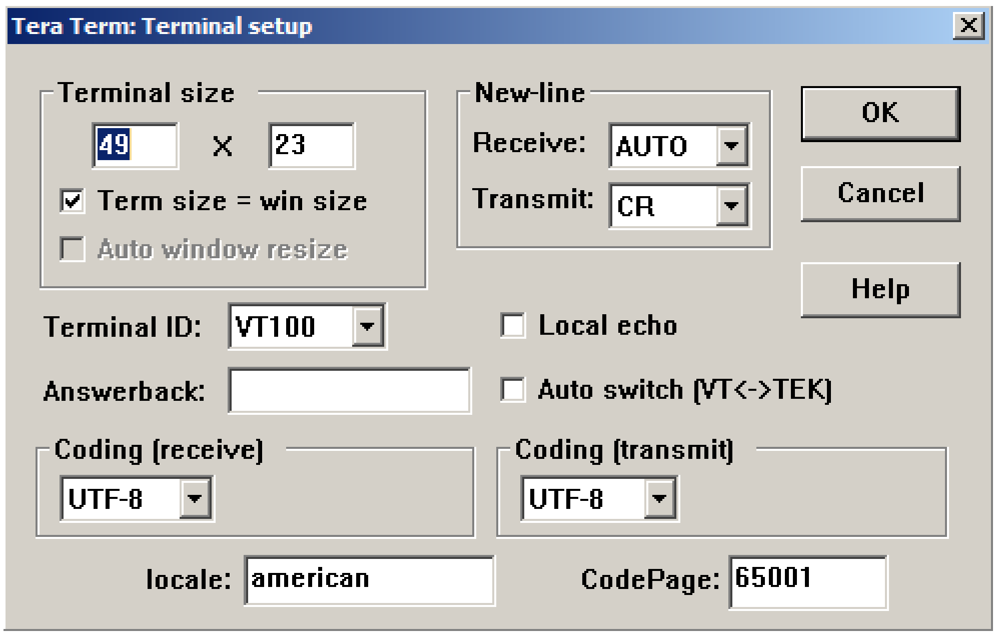
Transmit delay

0 msec/char 0 msec/line

1. Baud rate 115 200bit/s
2. 8 bit data
3. No parity
4. 1-bit stop
5. No flow control

This type of utility retrieves sensor data from the connected STM32 Nucleo board and displays it in tabular form. To display in the same view as in *Figure 17*, we recommend setting up the terminal as shown in *Figure 16* below.

Figure 16. Terminal setup



Tera Term: Terminal setup

Terminal size: 49 X 23
[✓] Term size = win size
[ ] Auto window resize

New-line
Receive: AUTO
Transmit: CR

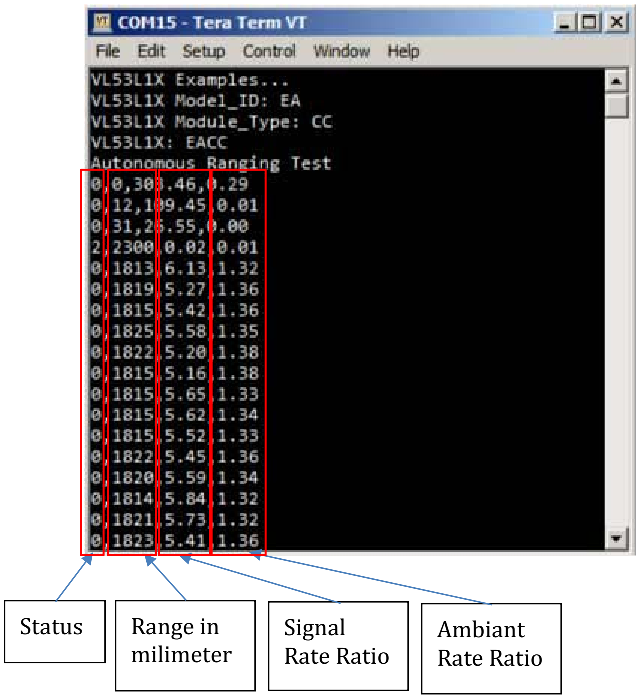
Terminal ID: VT100
Answerback:
[ ] Local echo
[ ] Auto switch (VT<->TEK)

Coding (receive): UTF-8
Coding (transmit): UTF-8

locale: american
CodePage: 65001

In the code example, the default configuration renders the “range status” on each line. This includes the “ranging data” in millimeters, the “signal rate ratio” in MegaCps/65536, and the “ambient rate” in MegaCps/65536 (see *Figure 17*).

Figure 17. Range status



```
VL53L1X Examples...
VL53L1X Model_ID: EA
VL53L1X Module_Type: CC
VL53L1X: EACC
Autonomous Ranging Test
0,0,303.46,0.29
0,12,109.45,0.01
0,31,26.55,0.00
2,2300,0.02,0.01
0,1813,6.13,1.32
0,1819,5.27,1.36
0,1815,5.42,1.36
0,1825,5.58,1.35
0,1822,5.20,1.38
0,1815,5.16,1.38
0,1815,5.65,1.33
0,1815,5.62,1.34
0,1815,5.52,1.33
0,1822,5.45,1.36
0,1820,5.59,1.34
0,1814,5.84,1.32
0,1821,5.73,1.32
0,1823,5.41,1.36
```

Nevertheless, the user can change this code to customize it and render all the data he may need.

5 Revision history

Table 2. Document revision history

Date	Revision	Changes
16-Mar-2018	1	Initial release