UM2090

User manual

Getting started with the software package for STEVAL-STLKT01V1 based on STM32Cube

Introduction

The STSW-STLKT01 firmware package for SensorTile provides sample projects for the development of custom applications.

Built on STM32Cube software technology, it includes all the low level drivers to manage the on-board devices and system-level interfaces.

The package comes with the DataLog_Audio, DataLog, AudioLoop and BLE_SampleApp applications.

The DataLog_Audio application allows the user to save the audio captured by the on-board microphone on SD card as a common .wav file.

The DataLog application features raw sensor data streaming via USB (Virtual COM Port class) and sensor data storage on an SD card exploiting RTOS features.

The AudioLoop application sends audio signals acquired by the microphone via I²S and USB interfaces, allowing the user to play the sound on loudspeakers/headphones or record it on an host PC.

The BLE_SampleApp provides an example of Bluetooth Low Energy configuration that enables SensorTile to stream environmental sensor data; it is compatible with the STBLESensor app available for Android and iOS.

1 What is STM32Cube?

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

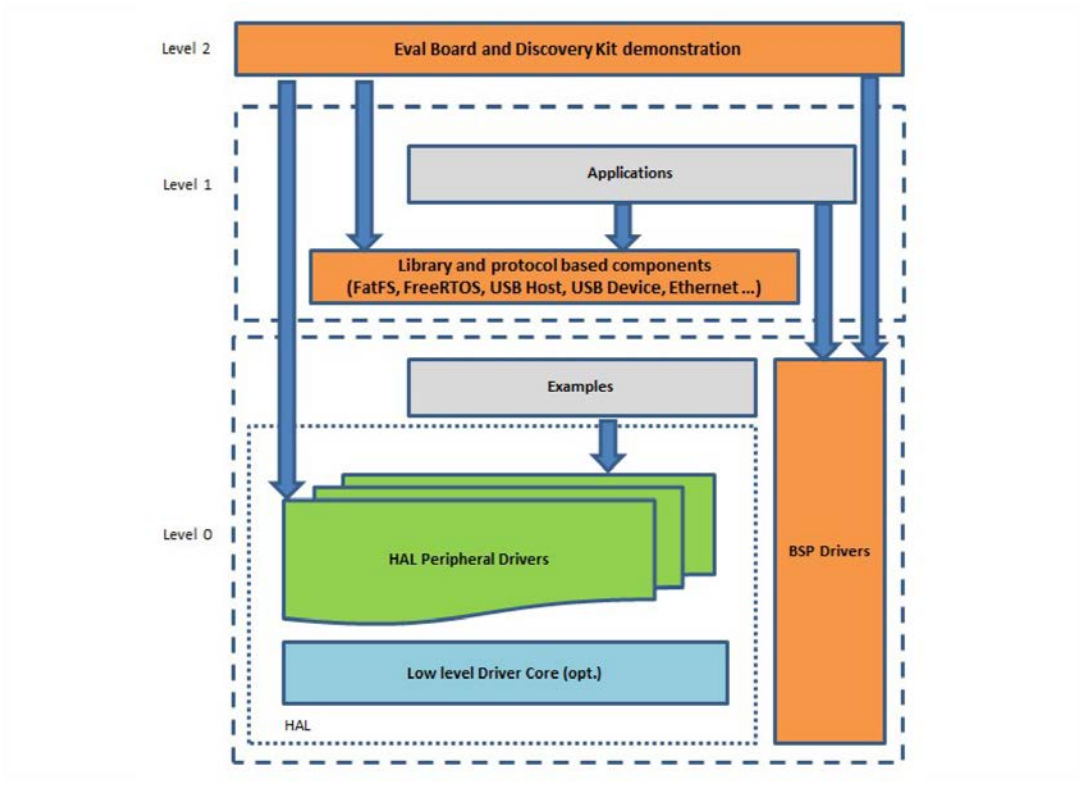
STM32Cube version 1.x includes:

- 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 STM32CubeF4 for the STM32F4 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

1.1 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
BSP Drivers
Low level Driver Core (opt.)
HAL

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 (I^2C, 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 STSW-STLKT01 software expansion for STM32Cube

2.1 Overview

This software package expands the functionality of the STM32Cube platform.

The key features of the package are:

- Embedded STM32L4 series software samples for SensorTile:
  - sensor data streaming via USB and logging on SD card
  - sensor data transfer via Bluetooth Low Energy
  - audio acquisition, playback and streaming via USB and on SD card
- Based on STM32Cube, the consistent and complete embedded software for STM32 MCU that maximizes portability across the entire STM32 series and avoids dependency issues
- DataLog_Audio application which allows the user to save the audio captured by the on-board microphone on SD card as a common .wav file
- A DataLog application which allows the real-time transmission of all sensor data to a PC via serial port or to save/log sensor data to file on an SD card
- An AudioLoop application which sends audio signals acquired by the microphone to an on-board DAC via an I²S interface and to the PC via USB
- A BLE_SampleApp which provides an example of Bluetooth Low Energy configuration
- A third party FAT file system module for small embedded systems
- Source code freely available from the ST website with developer-friendly license terms
- A third party RTOS (real-time operating system) kernel for embedded devices

This software enables data acquisition from different sensors like motion sensors, environmental sensors, and audio sensors via I²C, SPI or DFSDM.

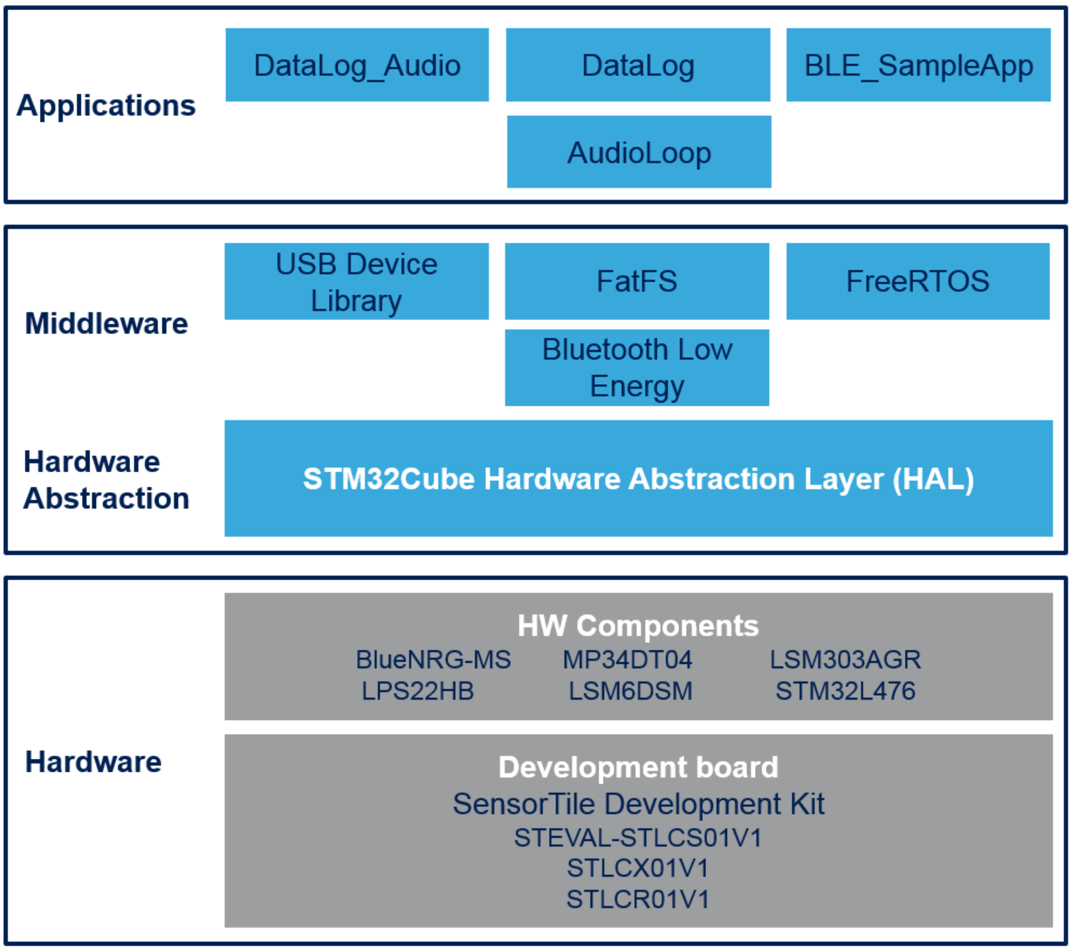
Exploiting the capabilities of an included VCP USB driver, the device is recognized as a Virtual COM Port by Microsoft Windows or any Unix-like system.

You can also save the data on an SD card if one is inserted in the relevant connector.

2.2 Architecture

This software is an expansion for STM32Cube, as such it fully complies with the architecture of STM32Cube and it expands it in order to enable development of applications using digital MEMS microphones.

Figure 2. **STSW-STLKT01 software architecture**



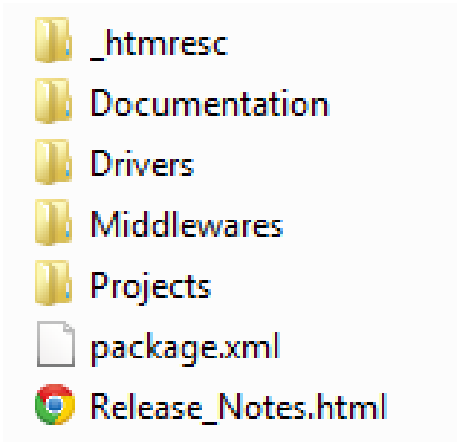
The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller and extends STM32Cube with a board support package (BSP) for the microphones expansion board and some middleware components for audio processing and USB communication with a PC.

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

- **STM32Cube HAL layer**: provides a generic, multi-instance, simple set of application programming interfaces (APIs) to interact with the upper layers (application, libraries and stacks). It is composed of generic and extension APIs. It is built directly around a generic architecture and allows layers built on it (like the middleware layer) to implement their functions without depending on specific hardware configurations for a given microcontroller unit (MCU). This structure improves library code reusability and guarantees an easy portability on other devices.
- **Board support package (BSP) layer**: contains the software to support the STM32 Nucleo board peripherals, except the MCU.

2.3 Folder structure

Figure 3. **STSW-STLKT01 package folders structure**



_htmresc
Documentation
Drivers
Middlewares
Projects
package.xml
Release_Notes.html

The following folders are included in the software package:

- Documentation: contains a compiled HTML file generated from the source code, detailing the software components and APIs.
- Drivers: contains the board-specific HAL 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: contains ST libraries for Virtual COM Port USB driver and the third party library FatFS.
- Projects: contains DataLog and AudioLoop sample applications, which can be evaluated with the IAR Embedded Workbench for ARM (IAR-EWARM), RealView Microcontroller Development Kit (MDK-ARM-STM32) and System Workbench for STM32 (SW4STM32) development environments.

2.4 APIs

Full user-API function and parameter descriptions are compiled in an HTML file located in the software Documentation folder.

2.5 The DataLog_Audio application

The DataLog_Audio application allows saving the audio captured by the on-board microphone on SD card as a common .wav file.

After reset, the STSW-STLKT01 firmware:

- configures HAL and clocks
- configures SD Card access, based on FatFS middleware
- configures sensors and microphone
- starts audio acquisition

To open the file and start saving the audio file, you have to double tap the SensorTile: the orange LED blinking means that the .wav file has been created and the recording has been initialized.

To stop the acquisition, you have to double tap the SensorTile once again: the LED stops blinking and the *SensorTile_Log_N000.wav* audio file is correctly saved on the SD card.

You can now repeat the operations and restart the data logging on another file (*SensorTile_Log_N001.wav*).

2.6 The DataLog application

The DataLog application has two operating modes that can be selected at compile time by changing the `LoggingInterface` variable in main.c

- LoggingInterface = USB_Datalog: sensors raw data streaming via USB (Virtual COM Port class)
- LoggingInterface = SDCARD_Datalog: sensors raw data storage on SD card.

After reset, the firmware:

1. configures HAL and clocks
2. configures LED1 (USB mode only)
3. initializes the USB peripheral or the SPI for SD card access
4. creates the threads and activate FreeRTOS scheduler

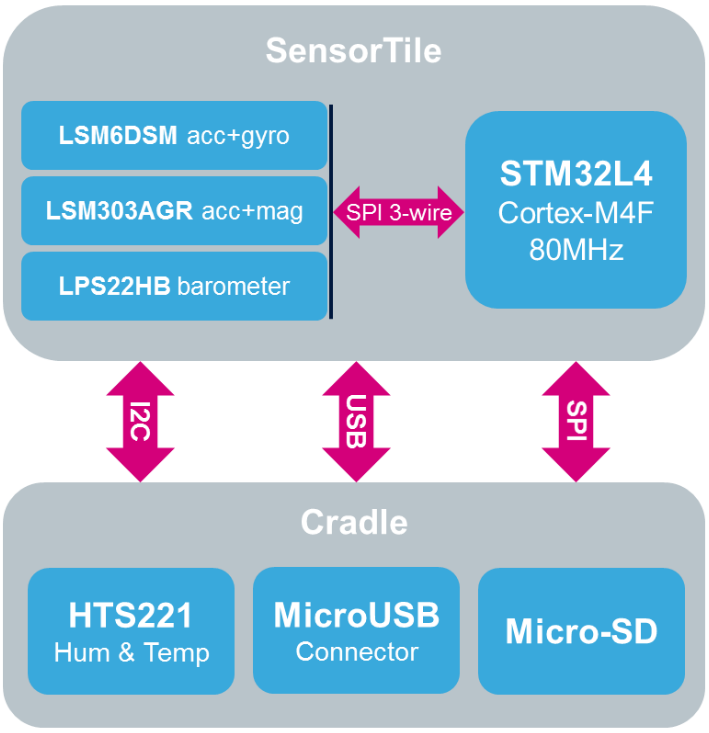
The GetData_Thread and WriteData_Thread threads are scheduled by FreeRTOS with different priorities and communicate with each other through a message queue:

- GetData_Thread: a high priority task that configures the sensors, reads data at a given frequency and pushes the new data in the queue. An OS timer triggers the execution of the thread at a given frequency.
- WriteData_Thread: a low priority task that configures the SD card and writes the sensor data as soon as they are available in the queue.

These priority differences ensure that the application does not lose sensor data even when writing operations on the SD Card take longer than the sampling period. If this happens, the WriteData_Thread is suspended by the scheduler to allow the execution of the GetData_Thread with the correct timing.

If SD card mode is selected, a double-tap on the board is needed to start logging data to SensorTile_Log_N000.tsv and another double-tap to stop. Subsequent operations restarts data logging to a new file (e.g., SensorTile_Log_N001.tsv).

Figure 4. **DataLog application block diagram**



2.7 AudioLoop application description

The AudioLoop application sends audio signals acquired by the microphone via I²S and USB interfaces, allowing the user to play these sounds on speakers or headphones, or record them on a host PC.

Following reset, the firmware:

1. configures HAL and clocks
2. configures LED1
3. configures the STM32 Serial Audio Interface peripheral (SAI) in I²S mode and the external DAC
4. configures the digital filter peripheral (DFSDM) for microphone acquisition
5. the BSP_AUDIO_IN_ClockConfig function (defined as weak in the BSP) is redefined as an empty function in main.c because clock and PLL configuration are already performed in the BSP_AUDIO_OUT_Init function

6. initializes USB audio class
7. starts audio acquisition

The main loop is empty in this application because all the operations needed to copy the audio stream acquired from the microphone to the serial audio interface are executed in the DFSDM interrupt.

For this reason, the `AudioProcess()` function is called by `BSP_AUDIO_IN_TransferComplete_CallBack()` and `BSP_AUDIO_IN_HalfTransfer_CallBack()`.

Figure 5. AudioLoop application diagram

2.7.1 Microphone acquisition process

A digital MEMS microphone can be acquired via different peripherals like SPI, I²S, GPIO or DFSDM. It requires an input clock and it outputs a PDM stream at the same frequency of the input clock. This PDM stream is further filtered and decimated for conversion into PCM standard for audio transmission.

Two different digital MEMS microphones can be connected on the same data line, configuring the first to generate valid data on the rising edge of the clock and the other on the falling edge, by setting the L/R pin of each microphone accordingly.

On the SensorTile (STEVAL-STLCS01V1), the microphone output signals are acquired via the digital filter for sigma delta modulators (DFSDM) peripheral, which generates the precise clock needed by the microphones and reads the PDM signal on the rising edge of the CLOCK line. The acquired signal is then input to the DSFDM filter for hardware filtering and decimation to generate a standard PCM stream. An additional software high pass filtering stage removes any DC offset in the output stream.

DMA is adopted in order to reduce MCU load.

2.8 BLE_SampleApp description

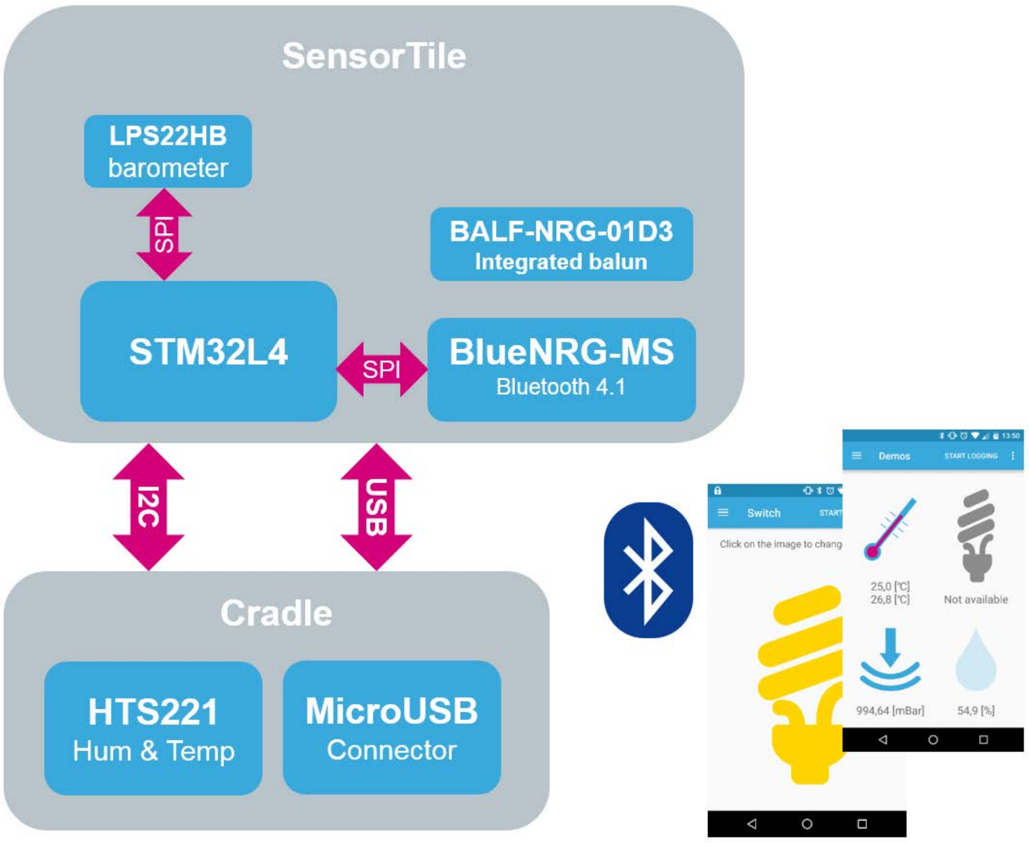
BLE_SampleApp provides an example of Bluetooth Low Energy configuration that enables SensorTile to stream environmental sensor data; it is compatible with STBLESensor app available for Android and iOS.

After resetting, the firmware:

- configures HAL and clocks
- configures and disable sensor chip select pins
- initializes the target platform:
  - USB peripheral (for debugging)
  - LED1
  - environmental sensors
- initializes the Bluetooth Low Energy stack
- initializes the Bluetooth Low Energy services
- initializes timers
- starts the main loop:
  - LED management
  - BLE event management

– environmental sensors data management

Figure 6. **BLE_SampleApp application diagram**



3 System setup guide

3.1 Hardware description and board programming

This section describes how to run the software examples on the SensorTile hardware (STEVAL-STLKT01V1).
The following figures show the hardware configurations:

Figure 7. SensorTile plus Cradle Expansion

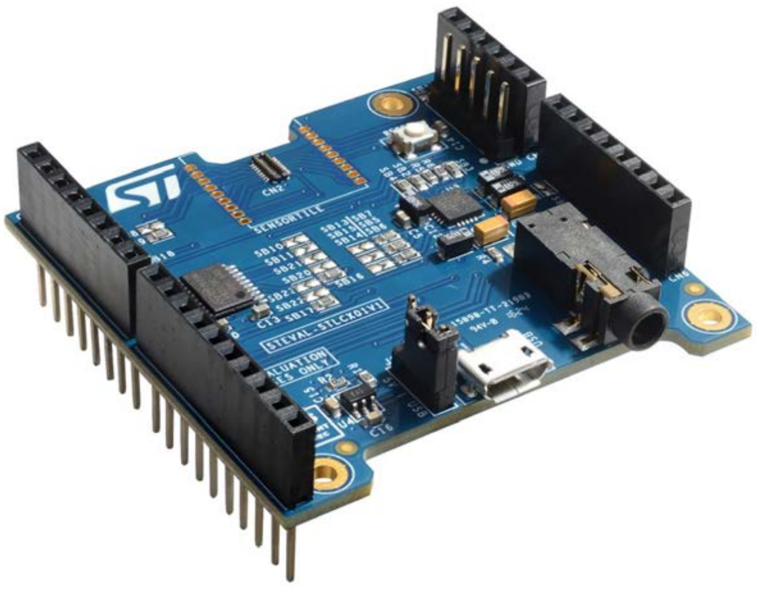
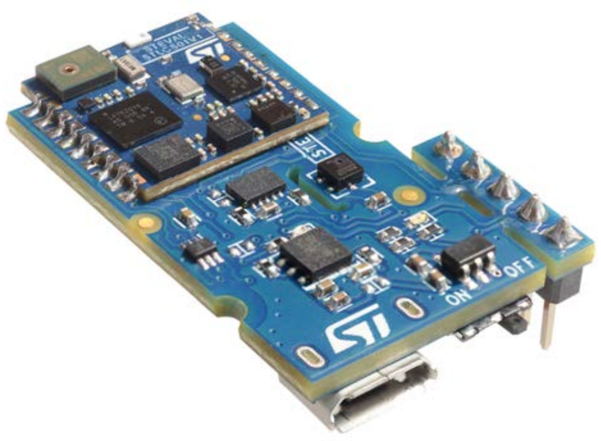


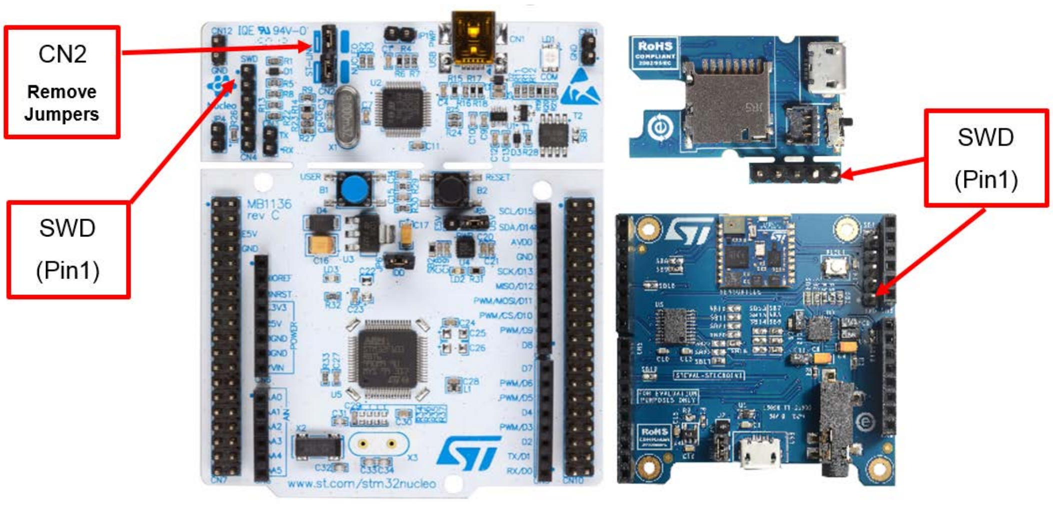
Figure 8. SensorTile plus Cradle



To program the board, connect an external ST-LINK to the SWD connector on the cradles; a five-pin flat cable is provided in the SensorTile Kit package.
The easiest way is to get an STM32 Nucleo board, which includes an ST-LINK V2.1 programmer.

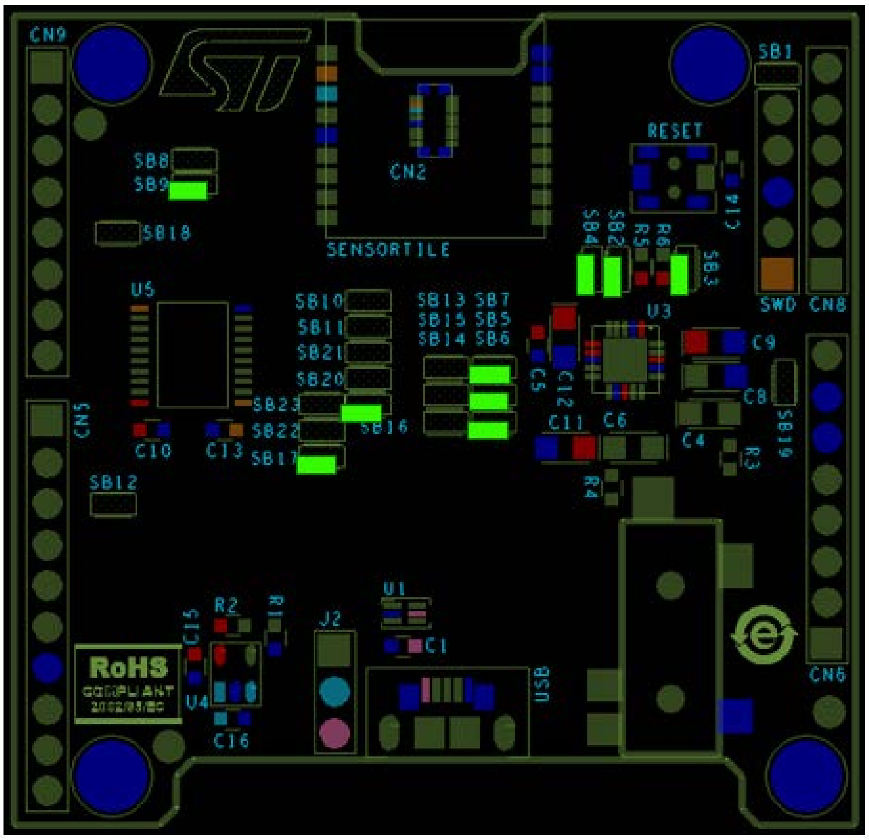
Ensure that CN2 Jumpers are OFF and connect (with correct polarity) your STM32 Nucleo board to the SensorTile Cradle with the cable provided. As shown below, Pin 1 is marked by a dot on the PCB silkscreen (STM32 Nucleo and SensorTile Cradle Expansion) or by the square shape of the soldering pad of the connector (SensorTile Cradle).

Figure 9. STM32 Nucleo board, Cradle and Cradle Expansion SWD connectors



On the SensorTile Cradle Expansion, there are a few solder bridges that can be used to modify some pin connections. The default configuration shown below will work with the provided software examples.

Figure 10. Default solder bridge configuration for Cradle Expansion (green = closed)



Next step is to select one of the applications in the package, open it with one of the supported IDEs and download the code onto the board.

3.2 Software and hardware compatibility

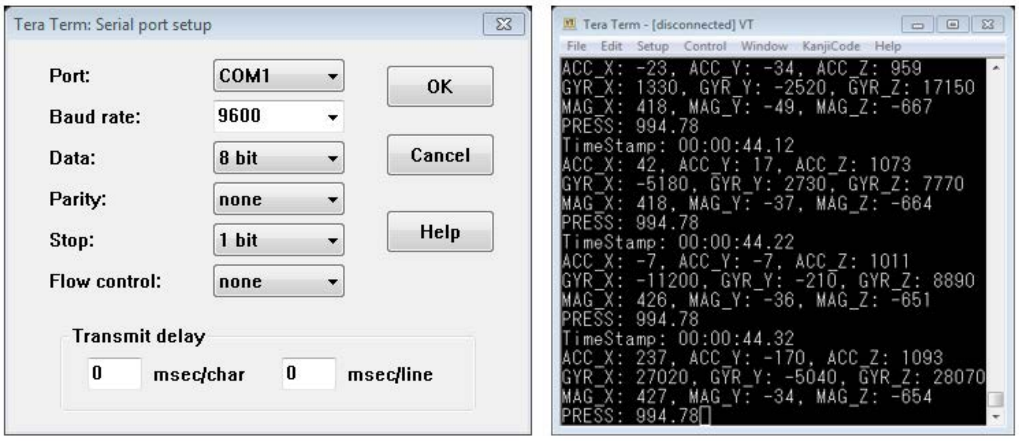
3.2.1 DataLog application - USB mode

The DataLog application in USB mode can run on both hardware configurations, as the only interface required is the USB connector present on both cradles.

Connect the board to a PC with a micro-USB cable and it will be recognized as a Virtual COM Port. If required, you can download the Windows driver from the ST website: VCP driver.

When connected to the PC, the board configures the sensors and starts streaming data to the PC. You can see the incoming data using any COM Port Terminal software like Putty or Tera Term. Check the correct COM Port number on your PC and use the following configuration.

Figure 11. Serial Port Configuration and received data example


Tera Term: Serial port setup
OK
Cancel
Help
Tera Term - [disconnected] VT
File Edit Setup Control Window KanjiCode Help

Port:	COM1
Baud rate:	9600
Data:	8 bit
Parity:	none
Stop:	1 bit
Flow control:	none

Transmit delay: 0 msec/char 0 msec/line

```
ACC_X: -23, ACC_Y: -34, ACC_Z: 959
GYR_X: 1330, GYR_Y: -2520, GYR_Z: 17150
MAG_X: 418, MAG_Y: -49, MAG_Z: -667
PRESS: 994.78
TimeStamp: 00:00:44.12
ACC_X: 42, ACC_Y: 17, ACC_Z: 1073
GYR_X: -5180, GYR_Y: 2730, GYR_Z: 7770
MAG_X: 418, MAG_Y: -37, MAG_Z: -664
PRESS: 994.78
TimeStamp: 00:00:44.22
ACC_X: -7, ACC_Y: -7, ACC_Z: 1011
GYR_X: -11200, GYR_Y: -210, GYR_Z: 8890
MAG_X: 426, MAG_Y: -36, MAG_Z: -651
PRESS: 994.78
TimeStamp: 00:00:44.32
ACC_X: 237, ACC_Y: -170, ACC_Z: 1093
GYR_X: 27020, GYR_Y: -5040, GYR_Z: 28070
MAG_X: 427, MAG_Y: -34, MAG_Z: -654
PRESS: 994.78
```

3.2.2 Datalog – SD card mode

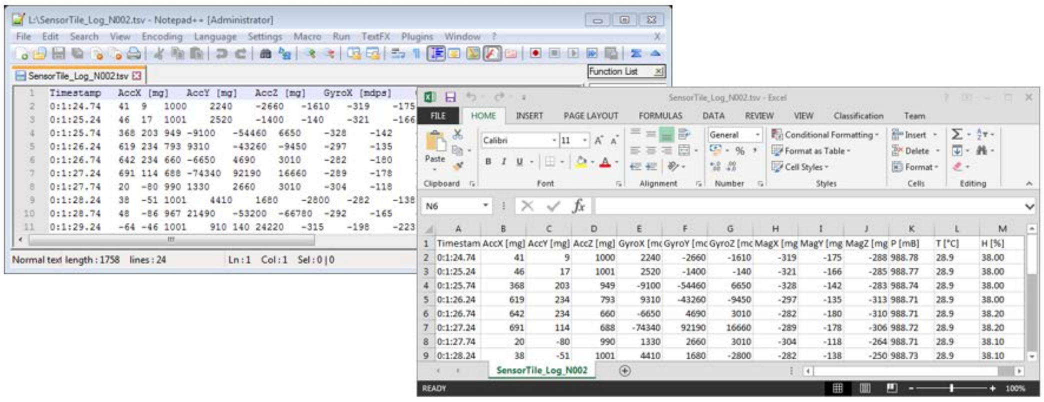
The DataLog application in SD card mode only runs on the SensorTile plus Cradle hardware configuration as the SD card socket is not present on the Cradle Expansion board.

The SD card is not provided in the package, but there is no particular limitation regarding the brand or type of card, apart from the fact that it must be an SDHC (Secure Digital High Capacity) and it must be formatted with the FAT32 file system.

Insert the card and power the board via the switch. Acquisition begins when a double-tap is detected by the on-board accelerometer; double-tap again to stop.

To verify the recorded data, just connect the SD card to a PC and open any of the .tsv (tab separated values) files you find.

Figure 12. Log file opened with a text editor (left) or Excel (right)


L:\SensorTile_Log_N002.tsv - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
SensorTile_Log_N002.tsv
Function List
SensorTile_Log_N002.tsv - Excel
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW Classification Team
Paste
Calibri
11
General
Conditional Formatting
Format as Table
Cell Styles
Insert
Delete
Format
Clipboard Font Alignment Number Styles Cells Editing
N6
SensorTile_Log_N002
READY
100%

```
Timestamp    AccX [mg]    AccY [mg]    AccZ [mg]    GyroX [mdps]
0:1:24.74    41   9    1000     2240     -2660    -1610    -319     -175
0:1:25.24    46   17   1001     2520     -1400    -140     -321     -166
0:1:25.74    368  203  949 -9100     -54460   6650     -328     -142
0:1:26.24    619  234  793 9310      -43260   -9450    -297     -135
0:1:26.74    642  234  660 -6650     4690     3010     -282     -180
0:1:27.24    691  114  688 -74340    92190    16660    -289     -178
0:1:27.74    20   -80  990 1330      2660     3010     -304     -118
0:1:28.24    38   -51  1001     4410     1680     -2800    -282     -138
0:1:28.74    48   -86  967 21490     -53200   -66780   -292     -165
0:1:29.24    -64  -46  1001     910 140 24220    -315     -198     -223
```

Normal text length : 1758   lines : 24   Ln : 1   Col : 1   Sel : 0 | 0

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Timestam	AccX [mg]	AccY [mg]	AccZ [mg]	GyroX [mc	GyroY [mc	GyroZ [mc	MagX [mg	MagY [mg	MagZ [mg	P [mB]	T [°C]	H [%]
2	0:1:24.74	41	9	1000	2240	-2660	-1610	-319	-175	-288	988.78	28.9	38.00
3	0:1:25.24	46	17	1001	2520	-1400	-140	-321	-166	-285	988.77	28.9	38.00
4	0:1:25.74	368	203	949	-9100	-54460	6650	-328	-142	-283	988.74	28.9	38.00
5	0:1:26.24	619	234	793	9310	-43260	-9450	-297	-135	-313	988.71	28.9	38.00
6	0:1:26.74	642	234	660	-6650	4690	3010	-282	-180	-310	988.71	28.9	38.20
7	0:1:27.24	691	114	688	-74340	92190	16660	-289	-178	-306	988.72	28.9	38.20
8	0:1:27.74	20	-80	990	1330	2660	3010	-304	-118	-264	988.71	28.9	38.10
9	0:1:28.24	38	-51	1001	4410	1680	-2800	-282	-138	-250	988.73	28.9	38.10

3.2.3 AudioLoop

The AudioLoop application only runs on the SensorTile plus Cradle Expansion configuration as the Audio DAC and the 3.5 mm jack are not present on the small Cradle board.

Once downloaded, the application configures the microphone and the audio DAC. The sound acquired by the microphone is then played on the connected headphones or loudspeaker.

Figure 13. AudioLoop application overview

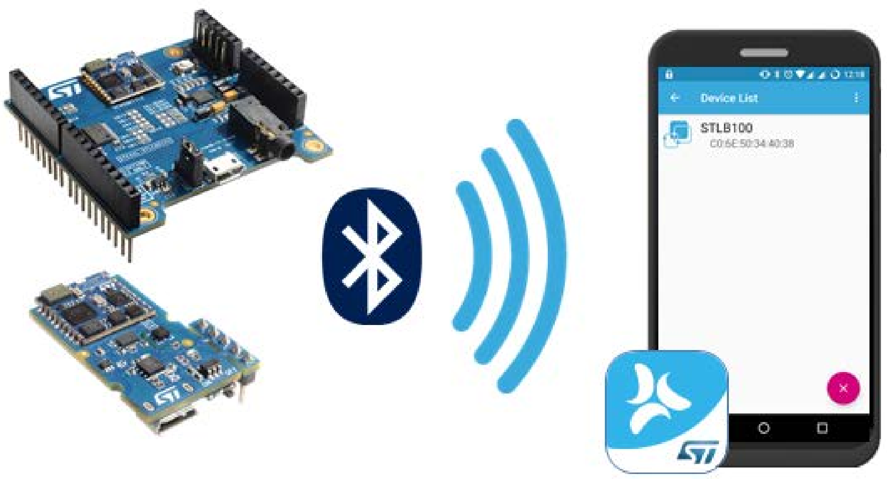
3.2.4 BLE_SampleApp

The BLE_SampleApp can run on both hardware configurations.

Once downloaded, the SensorTile LED starts blinking: the device is waiting for a connection via Bluetooth.

Open STBLESensor app on your Android or iOS smartphone and use it to connect to the SensorTile.

Figure 14. BLE_SampleApp application diagram



Revision history

Table 1. **Document revision history**

Date	Version	Changes
08-Jul-2016	1	Initial release.
09-Jan-2017	2	Updated *Introduction* and *Figure 2: "Overall system architecture"* Added *Section 2.8: "BLE_SampleApp description"* and *Section 3.2.4: "BLE_SampleApp"*
01-Jun-2017	3	Updated Introduction. Updated Figure 2: "Overall system architecture". Updated Section 2.5: "The DataLog application". Updated Section 2.6: "AudioLoop application description".
05-Mar-2019	4	Updated Introduction, Section 2.1 Overview and Figure 2. STSW-STLKT01 software architecture. Added Section 2.5 The DataLog_Audio application. Added references to STBLESensor app.

Contents

List of figures