# Getting started with STM32Cube function pack for MEMS microphones acquisition, advanced audio processing and audio output

## Introduction

FP-AUD-SMARTMIC1 firmware acquires audio signals through four digital MEMS microphones, elaborates them using embedded DSP libraries and streams the processed audio to a USB host and a loudspeaker connected to the relevant expansion board.

The package includes the following audio DSP libraries:

- AcousticBF library provides a real-time adaptive beamforming algorithm implementation: using the audio signals acquired from two digital MEMS microphones, it creates a virtual directional microphone pointing to a fixed direction;
- AcousticEC software implements a real-time echo cancellation routine based on the SPEEX implementation of the MDF algorithm;
- AcousticSL library provides a real-time sound source localization algorithm implementation: using two or four signals acquired from digital MEMS microphones, it can estimate the direction of arrival of the main audio source.

The firmware is based on the STM32Cube technology and provides an implementation example for the STM32F446RE microcontroller.

The application supports two kind of systems:

- STM32 NUCLEO-F446RE development board equipped with the X-NUCLEO-CCA01M1 expansion board (based on the STA350BW Sound Terminal® 2.1-channel high-efficiency digital audio output system), X-NUCLEO-CCA02M2 expansion board (based on the MP34DT06J digital MEMS microphones) and STEVAL-MIC001Vx, STEVAL-MIC002Vx and STEVAL-MIC003Vx digital microphone evaluation board series;
- BlueCoin starter kit (STEVAL-BCNKT01V1).

Information about STM32Cube is available at http://www.st.com/stm32cube, where a software example which performs an operation set to remotely control the device from a host PC is also provided.

# 1 Acronyms and abbreviations

**Table 1. Acronyms and abbreviations**

| Acronym | Description |
|---|---|
| BF | Beam forming |
| EC | Echo cancellation |
| SL | Source localization |
| BLE | Bluetooth low energy |
| DSP | Digital signal processing |
| MEMS | Micro electro-mechanical system |
| MCU | Micro controller unit |
| HAL | Hardware abstraction layer |
| BSP | Board support package |
| USB | Universal serial bus |
| PCM | Pulse code modulation |
| PDM | Pulse density modulation |
| PCB | Printed circuit board |

# 2 FP-AUD-SMARTMIC1 package description

## 2.1 Overview

FP-AUD-SMARTMIC1 package key features are:

- Software expansions for STM32Cube:
  - AcousticBF real-time beam forming
  - AcousticEC real-time acoustic echo cancellation
  - AcousticSL real-time sound source localization
- Complete application including all the acoustic functions in a single sample application
- Software graphic user interface to easily control parameters and algorithms from a host PC
- Free, user-friendly license terms
- Sample implementation available on a NUCLEO-F446RE development board when connected to an X-NUCLEO-CCA01M1 and an X-NUCLEO-CCA02M2 expansion board
- Sample implementation available on the BlueCoin starter kit (STEVAL-BCNKT01V1)

## 2.2 Firmware architecture

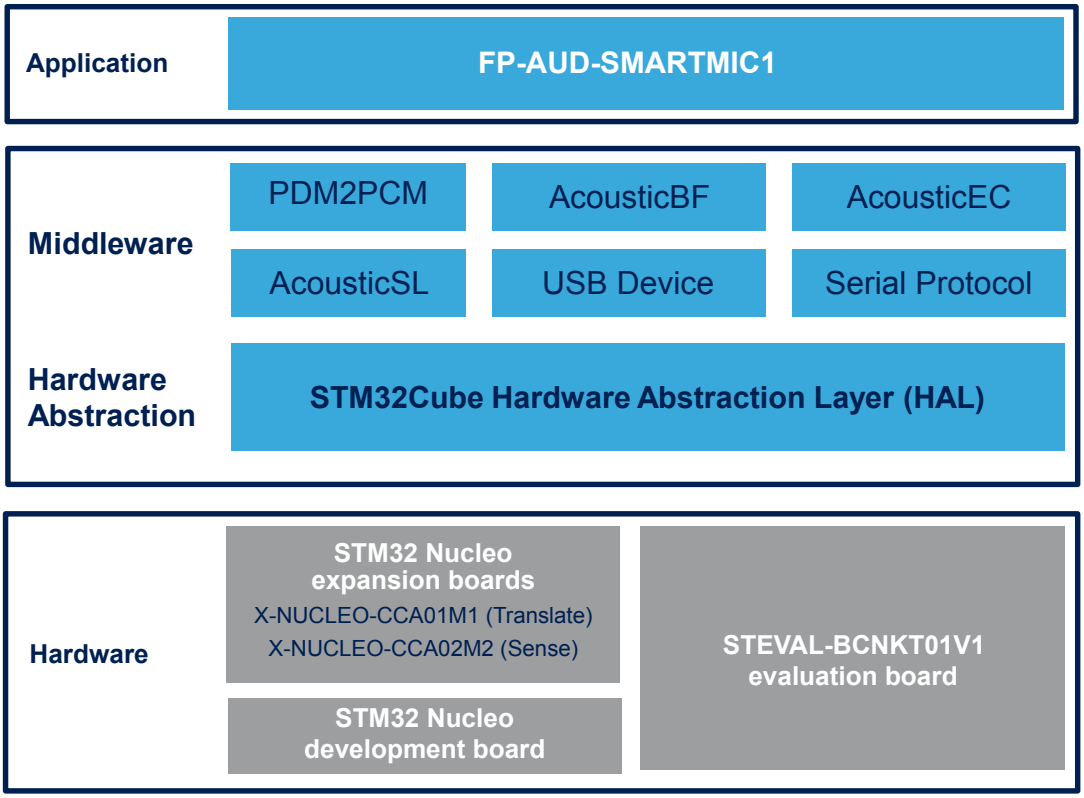
The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller.

The board support package (BSP) is intended for sensor expansion boards and middleware components for audio processing and serial communication with a PC.

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

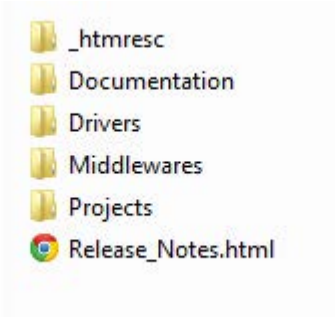
- **STM32Cube HAL layer**: provides a generic multi-instance simple set of application programming interfaces (APIs) to interact with the upper layers (application, libraries and stacks). It is composed of generic and extension APIs and directly built around a generic architecture, allowing built-upon layers (such as the middleware layer) to implement their functionalities without dependencies on the specific hardware configuration for a given microcontroller unit (MCU). This structure improves the library code reusability and guarantees an easy portability across other devices;
- **board support package (BSP) layer**: the software package supports the peripherals on the STM32 Nucleo board independently of the MCU. This software is included in the board support package (BSP). This is a limited set of APIs which provides a programming interface for certain board specific peripherals (e.g., the LED, the user button etc.). This interface also helps identifying the specific board version.

Figure 1. **FP-AUD-SMARTMIC1 software architecture**




## 2.3 Folder structure

Figure 2. **FP-AUD-SMARTMIC1 package folder structure**




The following folders are included in the software package:

- **documentation**: contains a compiled HTML file generated from the source code and documenting the software components and APIs in details;
- **drivers**: contains the HAL drivers, the board specific drivers for each supported board or hardware platform, including the on-board components and the CMSIS layer which is an independent vendor- hardware abstraction layer for the ARM® Cortex ®-M processor series;
- **middlewares**: contains different software expansions for STM32Cube: AcousticBF real-time beam forming, AcousticEC real-time acoustic echo cancellation and AcousticSL real-time sound source localization;
- **projects**: contains a sample application used to transmit the DSP middleware output to the PC host via USB, sending results through a passive speaker connected to SoundTerminal boards.

## 2.4 APIs

Detailed API function and parameter descriptions are available in a compiled HTML file in the package Documentation folder.

# 3 System setup guide

## 3.1 Hardware description

The hardware components to use the FP-AUD-SMARTMIC1 application are:

- an STM32 Nucleo-based system, consisting of a NUCLEO-F446RE board connected to an X-NUCLEO-CCA01M1 and an X-NUCLEO-CCA02M2 expansion boards.

*Note: A passive loudspeaker is required to enable some of the functions.*

- or a BlueCoin starter kit (STEVAL-BCNKT01V1)

*Note: An active loudspeaker is required to enable some of the functions.*

### 3.1.1 STEVAL-BCNKT01V1 BlueCoin kit

#### *3.1.1.1 Description*

The STEVAL-BCNKT01V1 integrated development and prototyping platform for augmented acoustic and motion sensing for IoT applications builds on the listening and balancing capabilities of the human ear.

With the expanded capabilities of its starter kit, BlueCoin lets you explore advanced sensor fusion and signal processing functions for robotics and automation applications with a 4 digital MEMS microphone array, a high-performance 9-axis inertial and environmental sensor unit and time-of-flight ranging sensors.

A high-performance STM32F446 180 MHz MCU enables real-time implementation of the very advanced sensor fusion algorithms like adaptive beamforming and sound source localization, with ready-to-use, royalty-free building blocks.

The BlueCoin can connect via the on-board BLE link to any IoT and smart industry wireless sensor network.

To upload new firmware onto the BlueCoin an external SWD debugger (not included in the starter-kit) is needed. It is recommended to use the ST-Link V2.1 found on any "STM32 Nucleo-64" development board.

#### *3.1.1.2 Features*

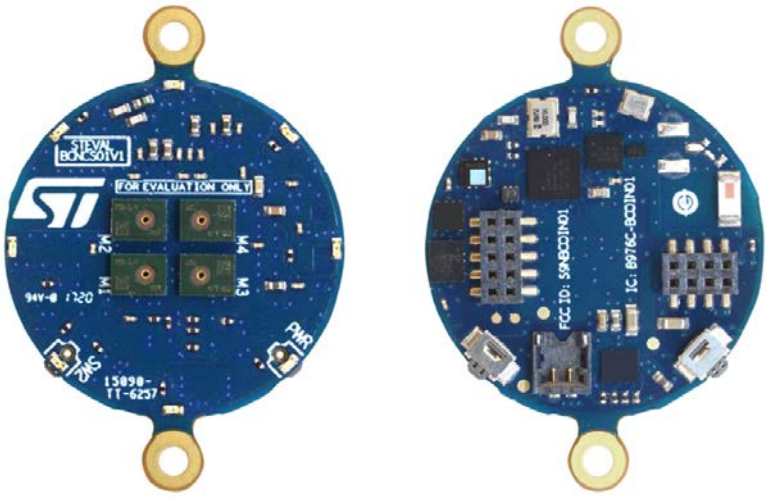
- Contains FCC ID: S9NBCOIN01
- Contains module IC 8976C-BCOIN01 certified with PMN: STEVAL-BCNKT01V1; HVIN: STEVAL-BCNCS01V1; HMN: STEVAL-BCNCR01V1; FVIN: bluenrg_7_2_c_Mode_2-32MHz-XO32K_4M.img
- The development kit package includes:
  - BlueCoin module (STEVAL-BCNCS01V1) with STM32F446, LSM6DSM, LSM303AGR, LPS22HB, 4x MP34DT06J, BlueNRG-MS, BALF-NRG-01D3, STBC03JR
  - CoinStation (STEVAL-BCNST01V1) board
  - BlueCoin Cradle (STEVAL-BCNCR01V1)
  - 130 mAh Li-Po battery
  - Plastic box for housing the BlueCoin cradle and the battery
  - SWD programming cable
- Software libraries and tools:
  - STSW-BCNKT01 firmware package with raw sensor data streaming support via USB, data logging on SD card, audio acquisition and audio streaming, time-of-flight example and BLE protocol to interface to a smartphone app
  - FP-AUD-SMARTMIC1: smart audio IN-OUT software expansion for STM32Cube
  - FP-SNS-ALLMEMS1 and FP-SNS-ALLMEMS2: STM32Cube function packs for BLE and sensors
  - FP-AUD-BVLINK1: BLE and microphones software expansion for STM32Cube
  - BlueMS: iOS™ and Android™ demo apps
  - BlueST-SDK: iOS and Android software development kit
  - Compatible with STM32 ecosystem through STM32Cube support

### 3.1.1.3 *Content of the starter kit*

**STEVAL-BCNCS01V1 - BlueCoin Core System board features**

- Very compact module for motion, audio and environmental sensing and Bluetooth low energy connectivity with a complete set of firmware examples
- Main components:
  - STM32F446 – 32-bit high-performance MCU (ARM® Cortex®-M4 with FPU)
  - 4x MP34DT06JTR – 64dB SNR Digital MEMS microphone
  - LSM6DSM – iNEMO inertial module: 3D accelerometer and 3D gyroscope
  - LSM303AGR – ultra-compact high-performance eCompass module: ultra-low power 3D accelerometer and 3D magnetometer
  - LPS22HB – MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer
  - BlueNRG-MS – Bluetooth low energy network processor
  - BALF-NRG-01D3 – 50 Ω balun with integrated harmonic filter
  - STBC03JR – linear battery charger with 150 mA LDO 3.0 V
- External interfaces: UART, SPI, SAI (Serial Audio Interface), I²C, USB OTG, ADC, GPIOs, SDIO, CAN, I2S
- SWD interface for debugging and programming capability
- The Bluetooth radio power output is set by default to 0 dBm; the FCC and IC certifications refer to this operating value. The power output can be changed up to 8 dBm by reprogramming the device firmware, but this change will require an update of the FCC and IC certifications, with additional radio emission tests to be performed.

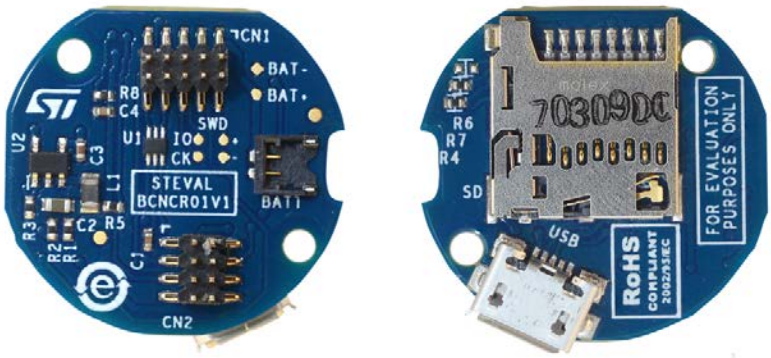
**Figure 3. STEVAL-BCNCS01V1 - BlueCoin Core System**




**STEVAL-BCNCR01V1 - BlueCoin Cradle board features**

- BlueCoin Cradle board with BlueCoin connectors
- ST1S12XX – 3.3 V step down DC-DC converter
- USBLC6-2P6 – very low capacitance ESD protection
- USB type A to Mini-B USB connector for power supply and communication
- microSD card socket

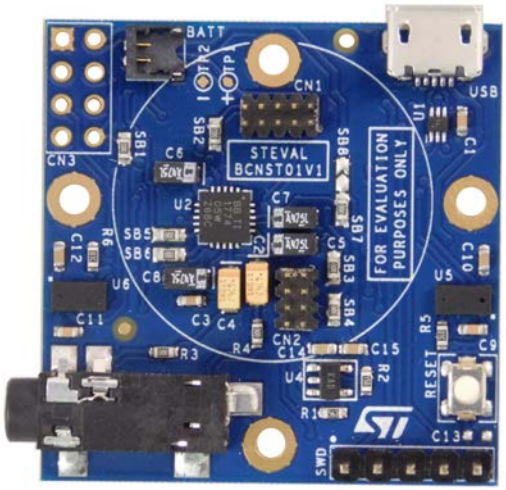
Figure 4. **STEVAL-BCNCR01V1 - BlueCoin Cradle board**




**STEVAL-BCNST01V1 - CoinStation board features**

- CoinStation expansion board with BlueCoin connectors
- LDK120M-R – 200 mA low quiescent current very low noise LDO
- USBLC6-2P6 – very low capacitance ESD protection for USB
- 2x VL53L0X Time-of-Flight (ToF) ranging sensor
- 16-Bit, low-power stereo audio DAC and 3.5 mm jack socket
- Micro-USB connector for power supply and communication
- Reset button
- SWD connector for programming and debugging

Figure 5. **STEVAL-BCNST01V1 - CoinStation board**




### 3.1.2 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 STM32 microcontroller lines.

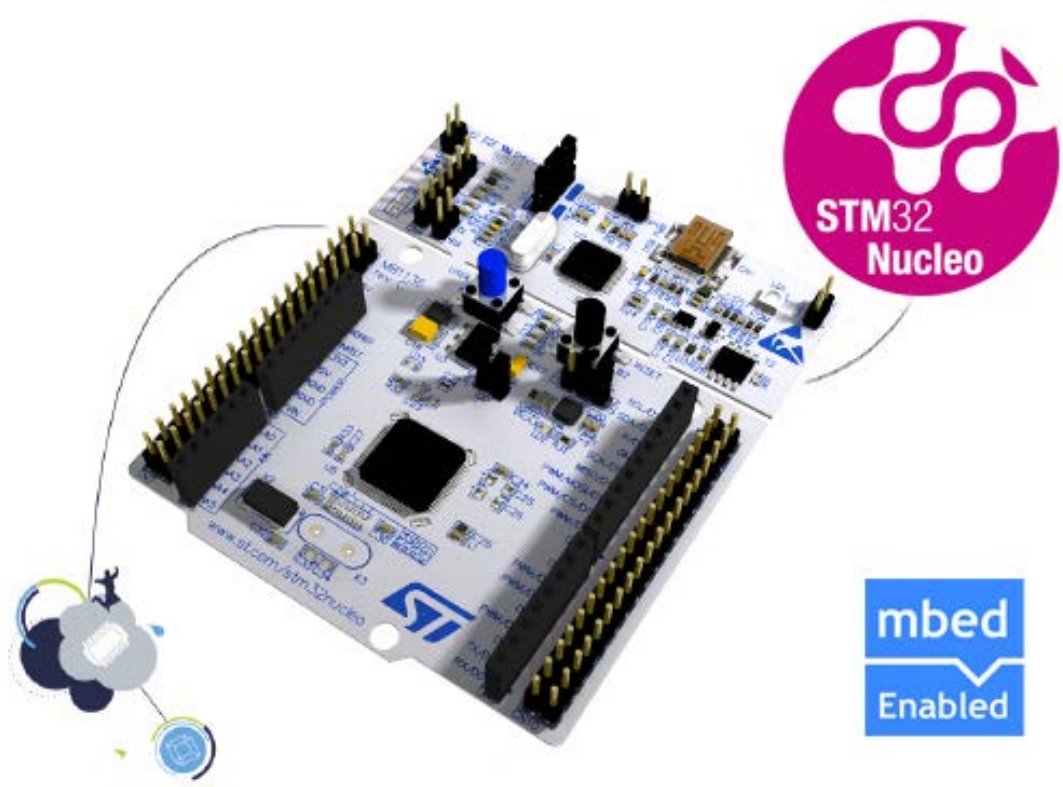
The Arduino™ connectivity support and ST morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide choice of specialized expansion boards.

The STM32 Nucleo board does not require any separate probe as it integrates the ST-LINK/V2-1 debugger/programmer. The STM32 Nucleo board includes the STM32 comprehensive software HAL library and various packaged software examples.

For SMARTMIC1 demonstration, a NUCLEO-F446RE STM32 Nucleo development board is specifically required.

Information about STM32 Nucleo boards is available at https://www.st.com/stm32nucleo

**Figure 6. STM32 Nucleo board**



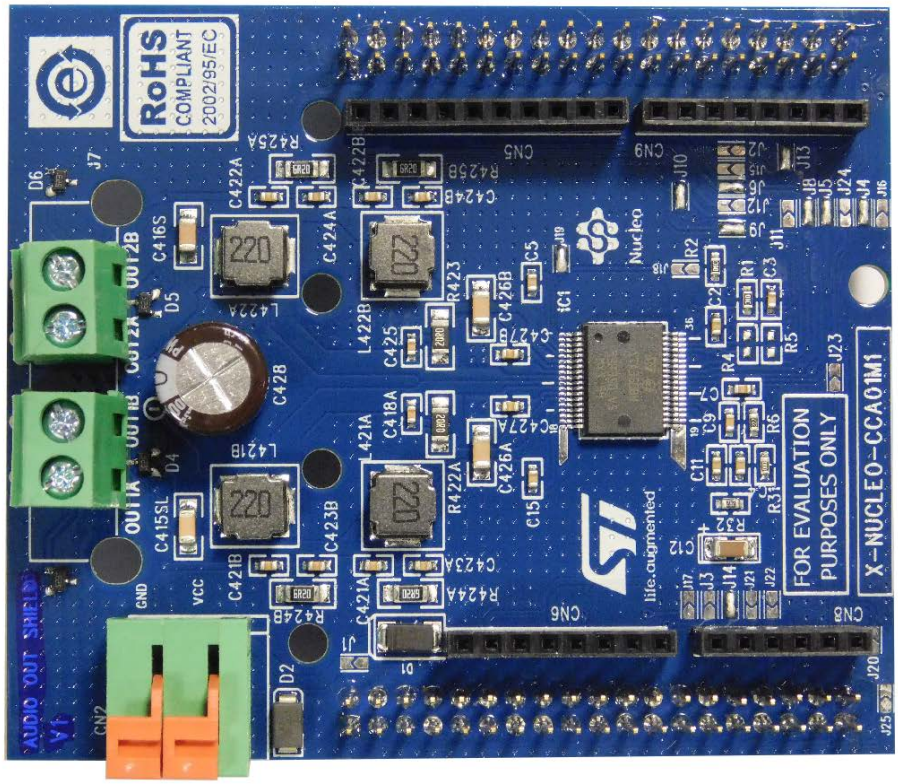

### 3.1.3 X-NUCLEO-CCA01M1 expansion board

The X-NUCLEO-CCA01M1 is an expansion board based on the STA350BW Sound Terminal® 2.1-channel high-efficiency digital audio output system.

It can be plugged on top of an STM32 Nucleo board and is compatible with the ST morpho connector layout. It enables the digital audio streams output to a speaker pair connected directly to the board and allows the evaluation of the STA350BW digital audio output component. Up to two X-NUCLEO-CCA01M1 expansion boards can be plugged on top of the same STM32 Nucleo host to build a four-channel digital audio output system.

The communication between the STM32 MCU and the STA350BW device is performed through the I²C bus interface for setup and control purposes and the I²S bus for digital audio transmission. A dedicated connector is available on the board to supply the power source for the output stage.

Figure 7. **X-NUCLEO-CCA01M1 expansion board**




### 3.1.4 X-NUCLEO-CCA02M2 expansion board

The X-NUCLEO-CCA02M2 expansion board has been designed around MP34DT06J digital MEMS microphone.

It is compatible with the ST morpho connector layout and with digital microphone coupon boards such as STEVAL-MIC001V1, STEVAL-MIC002V1 and STEVAL-MIC003V1.

The X-NUCLEO-CCA02M2 embeds two MP34DT06J microphones and allows synchronized acquisition and streaming of up to 4 microphones through I²S, SPI, DFSDM or SAI peripherals.

It represents a quick and easy solution for the development of microphone-based applications as well as a starting point for audio algorithm implementation.

Figure 8. **X-NUCLEO-CCA02M2 expansion board**

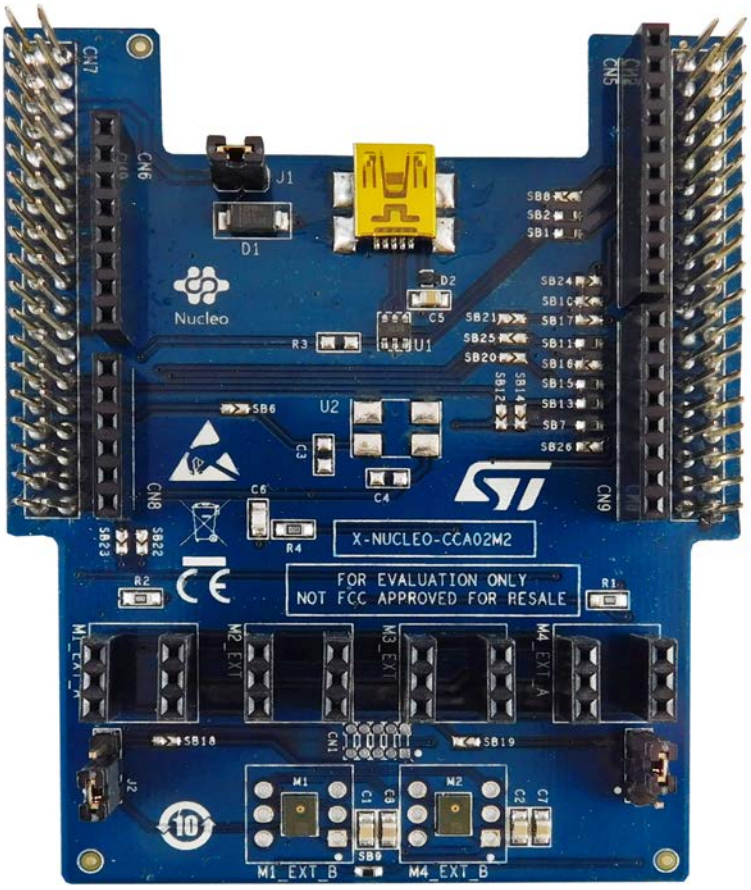
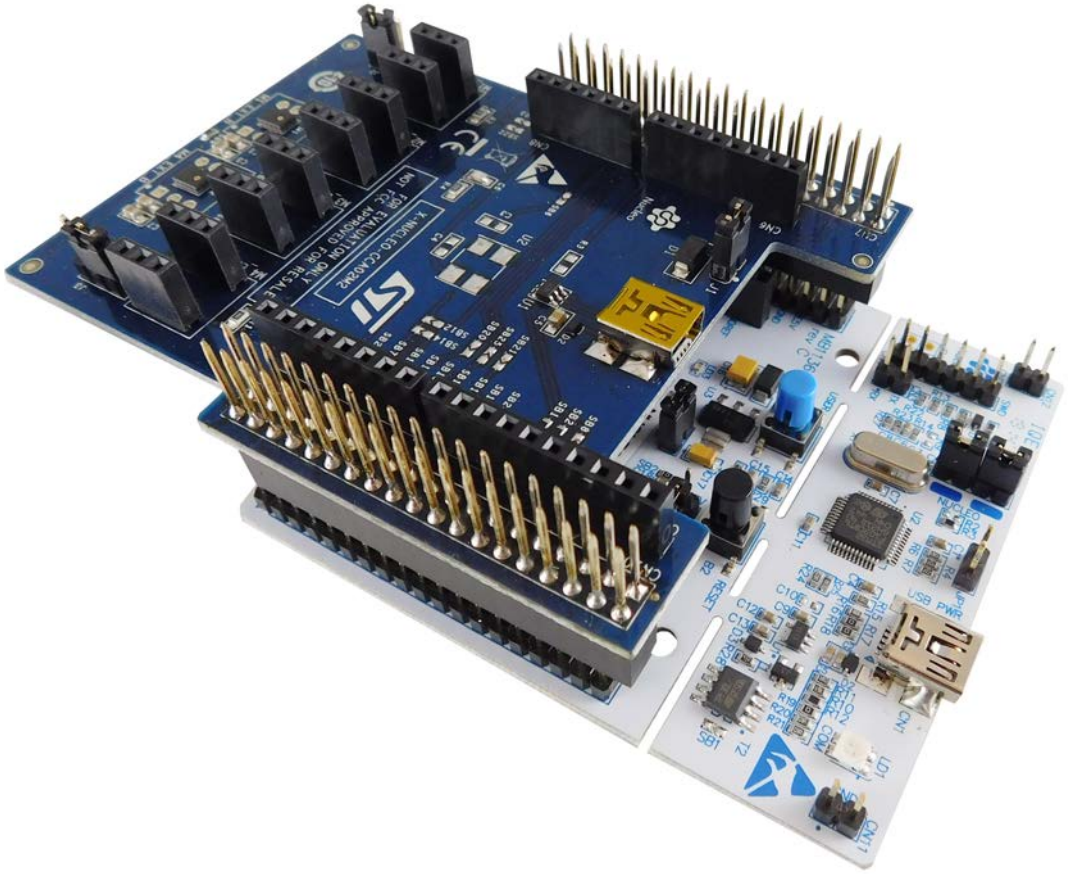


Figure 9. **X-NUCLEO-CCA02M2 on STM32 Nucleo board**



### 3.1.5 STEVAL-MIC001V1 evaluation board

The STEVAL-MIC001V1 daughterboard is designed to be used with the X-NUCLEO-CCA02M2 expansion board.

It contains four MP34DT05-A microphones and can export the four additional PDMs for any user application requirement (NBW algorithm detection).

The single PCB hosting each microphone can be detached.

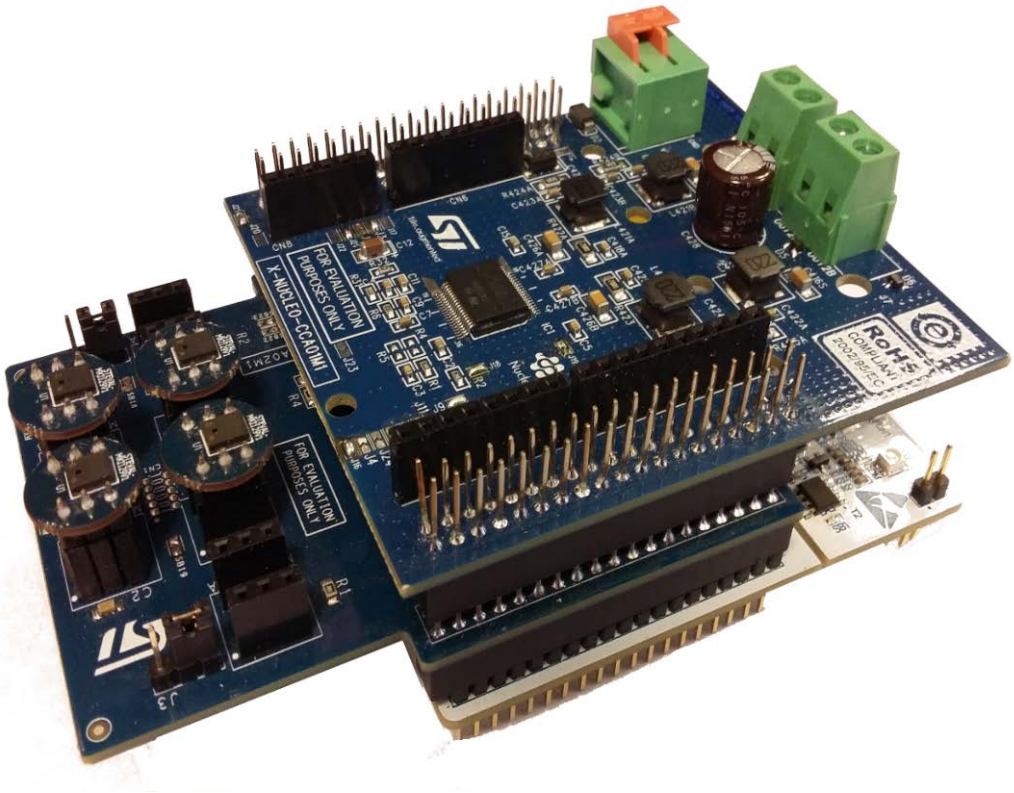
Figure 10. **STEVAL-MIC001V1 evaluation board**


### 3.1.6 Loudspeaker

When adopting an STM32 Nucleo-based system, a passive loudspeaker must be connected to the X-NUCLEO-CCA01M1 expansion board OUT1 connector.

Figure 11. **Loudspeaker connected to the board stack**

Figure 12. **X-NUCLEO-CCA01M1 expansion board connected to a STM32 Nucleo development board over the X-NUCLEO-CCA02M1 expansion board with STEVAL-MKI129V1**




When using a BlueCoin starter kit, a standard active loudspeaker can be plugged to the headphone output connector on the BlueCoin station.

## 3.2 Hardware and software setup

### 3.2.1 Hardware setup

When using an STM32 Nucleo-based system, the following hardware components are needed:

- an STM32 Nucleo development platform (NUCLEO-F446RE)
- an STA350BW Sound Terminal® expansion board (X-NUCLEO-CCA01M1)
- a digital MEMS microphone expansion board (X-NUCLEO-CCA02M2)
- a digital MEMS microphone daughter board (STEVAL-MIC001V1)
- two USB type A to Mini-B USB cables to connect the NUCLEO-F446RE and the X-NUCLEO-CCA02M2 to the PC (respectively for serial communication and for microphone input)
- a passive loudspeaker

When using BlueCoin systems, the following hardware components, both available in a single BlueCoin kit, are needed:

- an STEVAL-BCNCS01V1: BlueCoin core system
- an STEVAL-BCNST01V1: BlueCoin station
- an active loudspeaker

### 3.2.2 STM32 Nucleo and expansion board setup

#### *3.2.2.1 Hardware configuration*

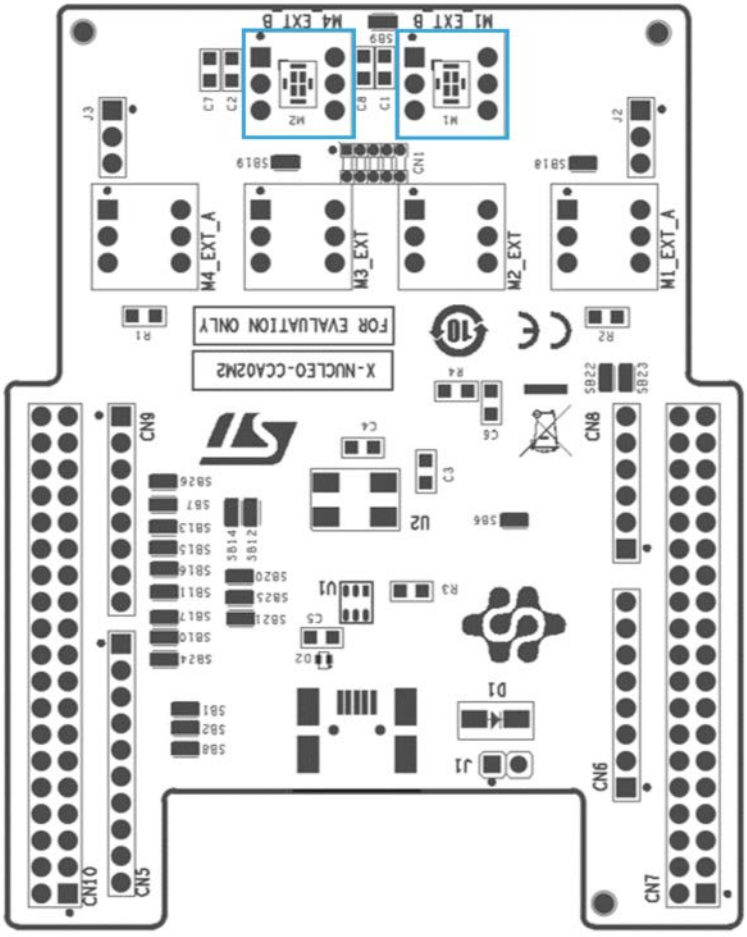
When adopting a Nucleo-based system, some preliminary operations are needed to configure the expansion boards for specific use cases(For further information about audio expansion configuration, refer to www.st.com.).

In order to build the whole system, the following steps are needed on the X-NUCLEO-CCA02M2:

**Step 1.** solder the external headers (M1_EXT_B, M4_EXT_B) onto the X-NUCLEO-CCA02M2 board to create a square array with microphones coupon.

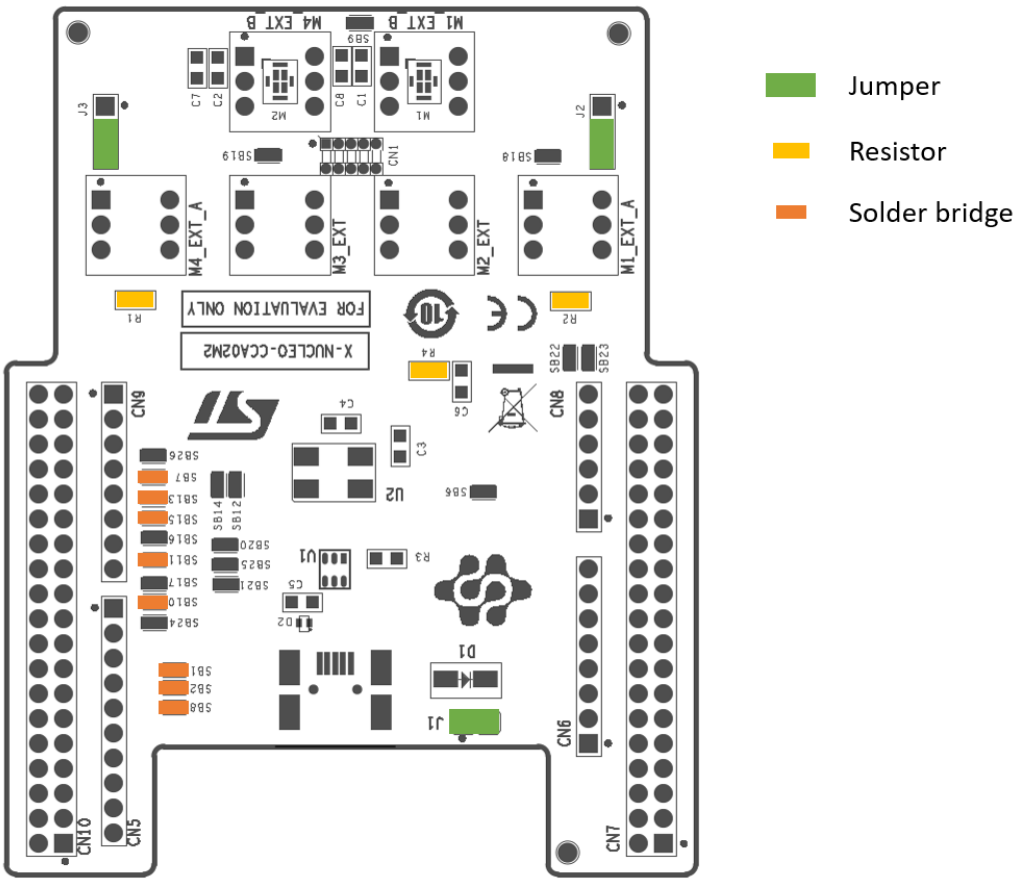
The connectors to be soldered are highlighted in the picture below.

**Figure 13. X-NUCLEO-CCA02M2 soldered connectors**



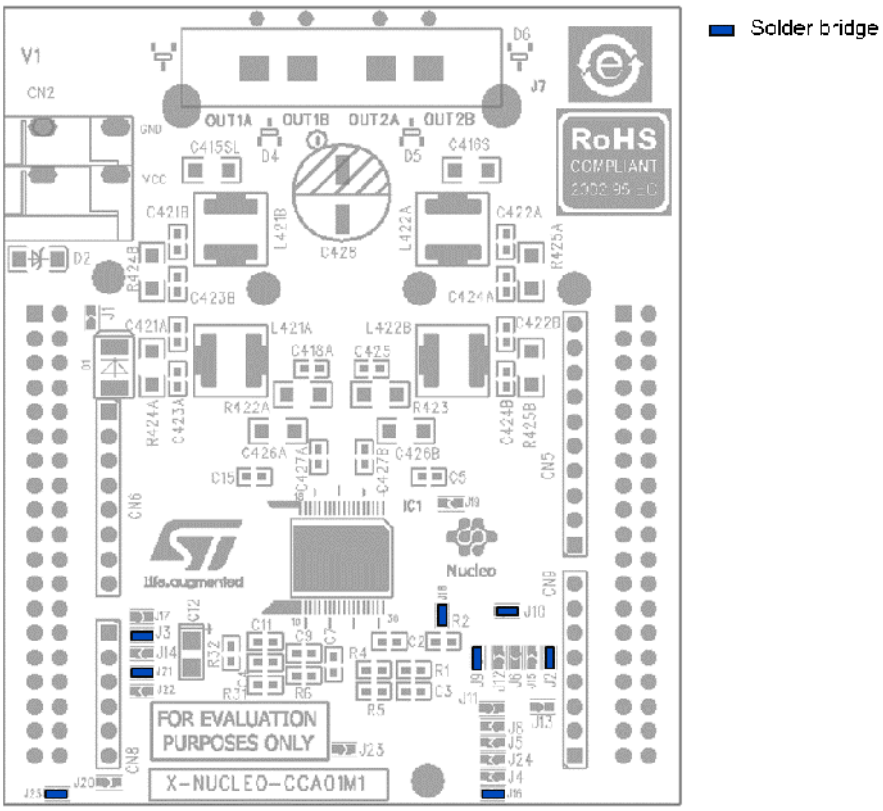
**Step 2.** plug 4x microphone coupons on the four headers M1_EXT_B, M2_EXT, M3_EXT M4_EXT_B and configure the board in order to acquire 4 microphones, as shown below.

**Figure 14. X-NUCLEO-CCA02M2 configuration**



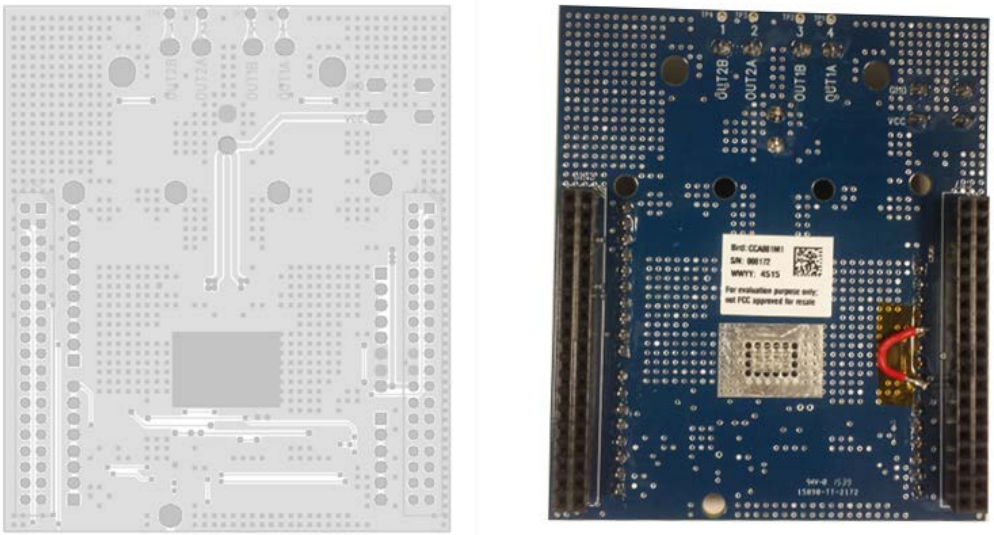
**Step 3.** Configure the X-NUCLEO-CCA01M1 to act as the second device, as shown below.

**Figure 15. X-NUCLEO-CCA01M1 configuration**




**Step 4.** Power the X-NUCLEO-CCA01M1 power stage using a wire connected between CN6.5 on the Arduino connector and CN2.VCC on the expansion board (or soldering a patch on the bottom side)

**Figure 16. Wire connection on X-NUCLEO-CCA01M1**



**Step 5.** Connect the passive loudspeaker to the X-NUCLEO-CCA01M1 OUT1 connector (refer to Section 3.1.6 Loudspeaker).

When the hardware setup is completed, the firmware can be loaded into the STM32 Nucleo board.

#### 3.2.2.2 *Host PC configuration*

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. This device exports a Virtual Com Port used by the application to exchange messages between the FP-AUD-SMARTMIC1 system and the host PC.

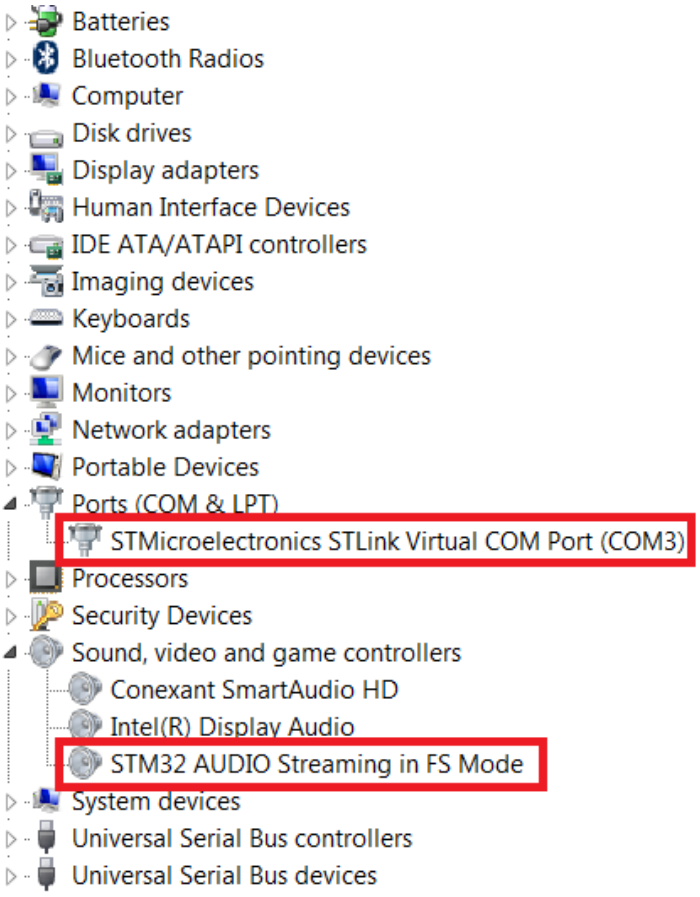
The developer can download the ST-LINK/V2-1 USB driver by searching the STSW-LINK009 package available at www.st.com..

Use the following procedure to ensure the correct configuration of the hardware/firmware system.

**Step 1.** Connect the system to the host PC using two USB cables connected to the STM32 Nucleo USB and the X-NUCLEO-CCA02M1 USB receptacles.

Thus, the system can be recognized by the PC as a Virtual Com Port (thanks to the connection to the STM32 Nucleo board) and a standard USB microphone (thanks to the connection to the X-NUCLEO-CCA02M1 expansion board)

**Figure 17. FP-AUD-SMARTMIC1 device recognition in Windows 7 when using STM32 Nucleo-based systems**

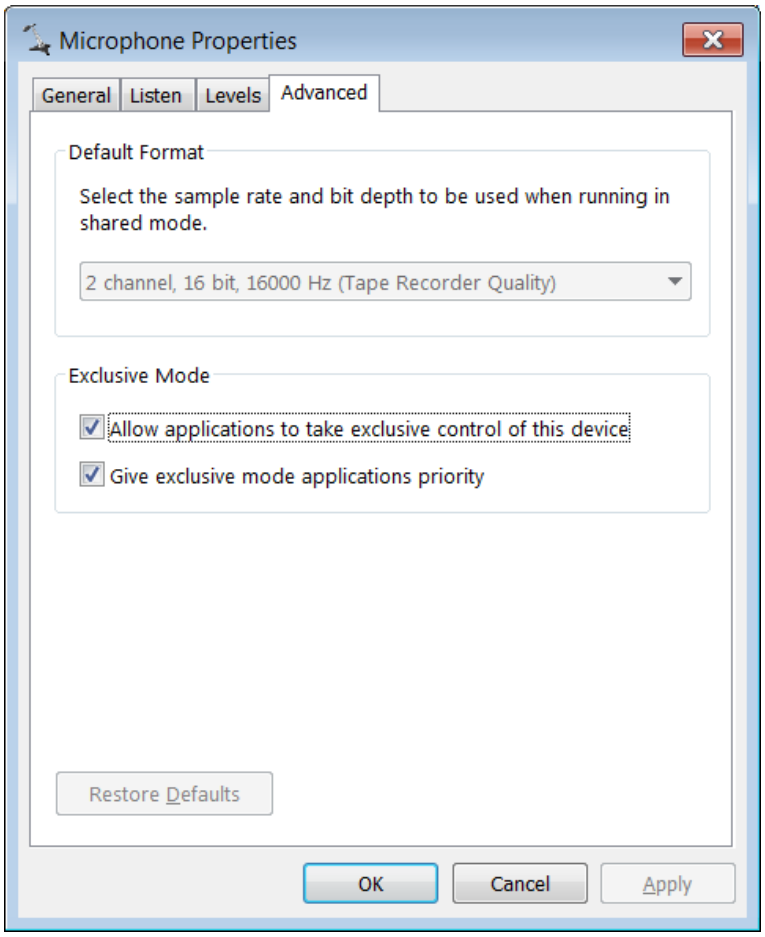



**Step 2.** Right-click on the volume icon in the Windows task bar (on the bottom right corner of the screen) and choose **Recording device**.

**Step 3.** Select STM32 microphone and click on **Properties**.

The **Advanced tab** includes a summary of the current device setup in terms of sampling frequency and number of channels. You should see the module recognized as a "1 channel, 16000 Hz" microphone.

**Figure 18. Microphone properties – Advanced tab**



### 3.2.3 STEVAL-BCNKT01V1 BlueCoin kit

#### *3.2.3.1 Hardware configuration*

**Step 1.** Plug the BlueCoin core system (STEVAL-BCNCS01V1) to the Coin Station (STEVAL-BCNST01V1) through the dedicated connectors.

**Step 2.** Connect an external ST-LINK to the SWD connector on the Coin Station.

A 5-pin flat cable is provided in the BlueCoin Kit package.

STM32 Nucleo boards bundle ST-LINK V2.1 debuggers and programmers.

**Step 3.** Ensure that CN2 jumpers are OFF and connect your STM32 Nucleo board to the Coin Station via the cable provided, paying attention to the polarity of the connectors.

Pin 1 is identified by a small circle on the PCB silkscreen on the STM32 Nucleo board and the Coin Station.

It is possible to reproduce the audio with a headset or a speaker connected to the 3.5 mm phone jack.

**Figure 19. SWD connections with 5-pin flat cable**

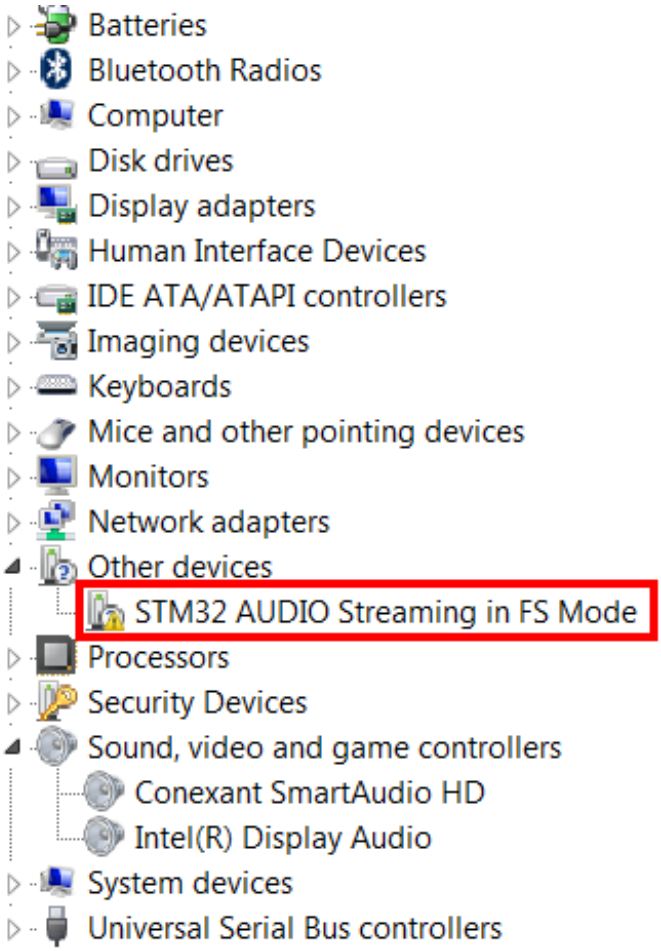
#### *3.2.3.2 Host PC configuration*

When using a BlueCoin kit, the firmware adopts a composite driver that exports a Virtual Com Port and a standard USB microphone. This kind of device is automatically recognized by all Windows versions, except Windows 7.

For Windows 7, follow the steps below to use the device.

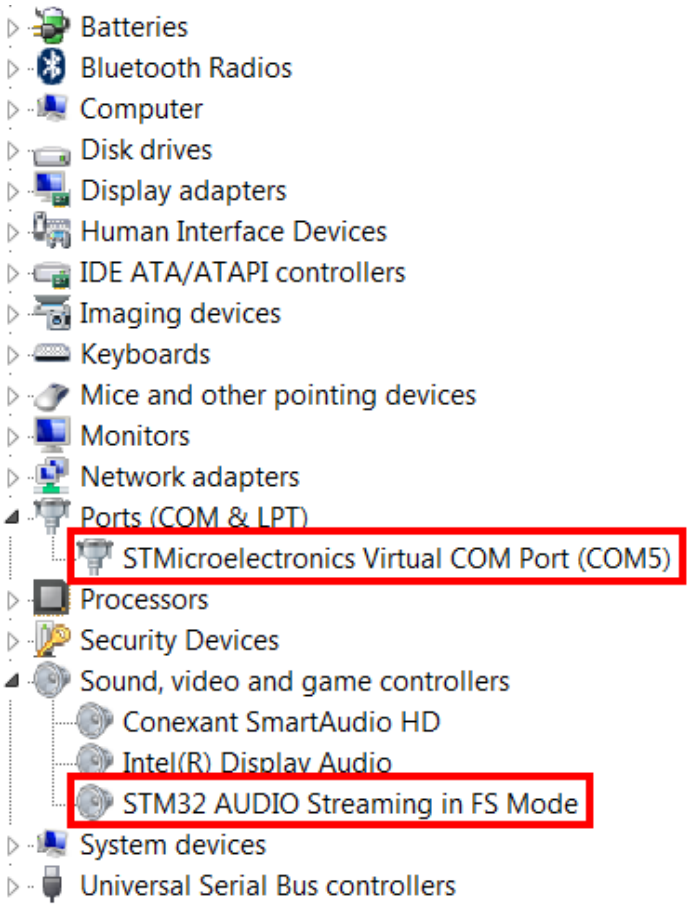
**Step 1.** Connect the system to the host PC using a micro-USB cable connected to the BlueCoin station.

**Figure 20. Windows 7: the composite device is not automatically recognized**



**Step 2.** Right click on the unrecognized STM32 Audio streaming device, choose update driver and locate the .inf driver included in the FP-AUD-SMARTMIC1 package downloaded from ST website.

**Figure 21. FP-AUD-SMARTMIC1 devices recognition in Windows 7 when using BlueCoin-based systems**

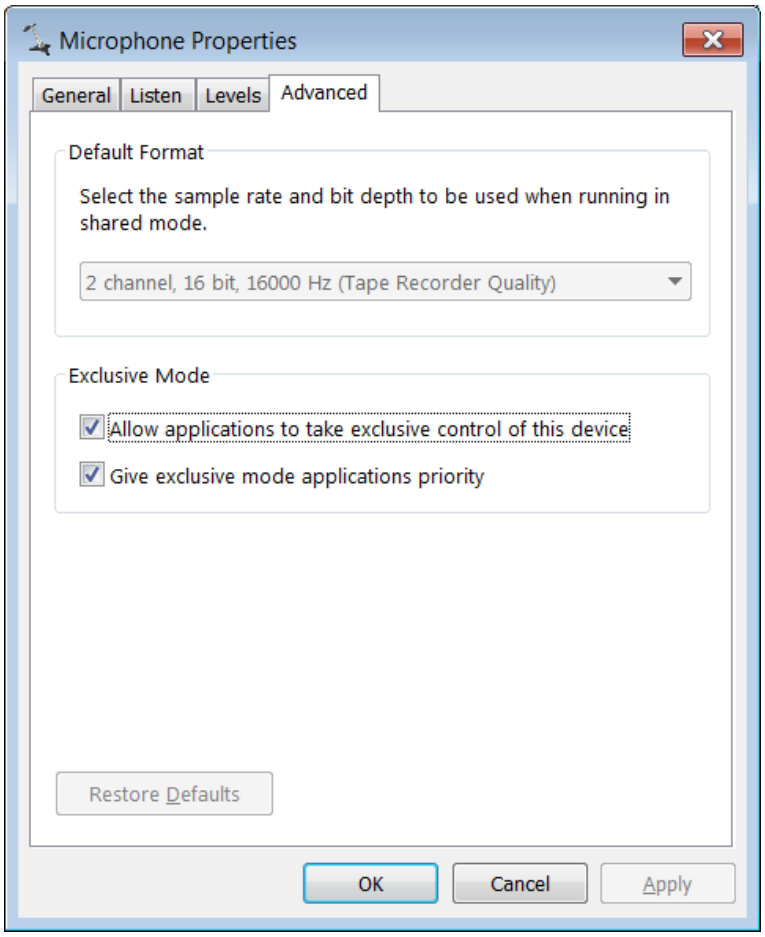



**Step 3.** Right-click on the volume icon in the Windows task bar (on the bottom right corner of the screen) and choose **Recording device**.

**Step 4.** Select STM32 microphone and click on **Properties**.

The **Advanced tab** includes a summary of the current device setup in terms of sampling frequency and number of channels. You should see the module recognized as a "2 channel, 16000 Hz" microphone.

**Figure 22. Microphone properties – Advanced tab**




### 3.2.4 Software setup

The following software components are needed to set up a suitable development environment for testing and using the FP-AUD-SMARTMIC1 application for the STM32 Nucleo:

- FP-AUD-SMARTMIC1: a smart in/out audio software package; firmware and documentation are available at www.st.com ;
- one of the integrated development environments supported by the STM32Cube expansion software; refer to the system requirements and setup information provided by the selected IDE provider.

#### *3.2.4.1 Development tool chains and compilers*

The STM32Cube expansion software supports the following environments:

- IAR embedded workbench for ARM® (EWARM) tool chain + ST-LINK
- RealView microcontroller development kit (MDK-ARM-STR) tool chain + ST-LINK
- STM32CubeIDE + ST-LINK

#### *3.2.4.2 Parameter optimization*

The STEVAL-BCNKT01V1 (BlueCoin) production lot “1720” mounts MP34DT04-C1 MEMS microphone instead of MP34DT06J.

You can easily recognize the lot from the BlueCoin Core System PCB silkscreen as highlighted in the figure below.

Figure 23. **BlueCoin production lot "1720"**


By default the FP-AUD-SMARTMIC1 firmware comes optimized parameters for the newer BlueCoin.
To get the best audio performance, when using a "1720" BlueCoin, it is necessary to:

- uncomment `#define BLUECOIN_1720` in audio_application.h

Figure 24. **STEVAL-BCNCS01V1: custom code for the 1720 lot**

```
//#define BLUECOIN_1720   /*Uncomment to configure mic distance for '1720' signed BlueCoin*/

#if defined (STM32_BLUECOIN)
#define TOP_RIGHT_MIC 2
#define TOP_LEFT_MIC 3
#define BOTTOM_RIGHT_MIC 0
#define BOTTOM_LEFT_MIC 1

#ifdef BLUECOIN_1720
#define SIDE_X 40
#define SIDE_Y 40
#define DIAGONAL 57
#else
#define SIDE_X 40
#define SIDE_Y 34
#define DIAGONAL 53
#endif
```

- rebuild all files and load the binary on the BlueCoin

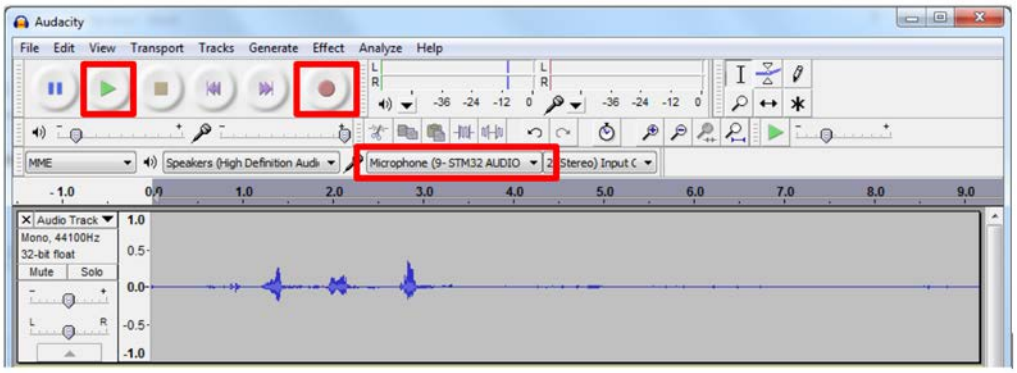
### 3.2.4.3 *PC audio recording utility example: Audacity*

In order to test the application and full capabilities of the FP-AUD-SMARTMIC1 application, a third party recording software is necessary to record the processed audio and evaluate the running algorithms.

Audacity® is an open source, cross-platform program for recording and audio editing environment, freely available on the web.

To start audio recording, check that the audio input device is STM32 AUDIO streaming in FS mode and then start recording and performing other functions using the interface.

**Figure 25. Audacity for Windows**



# 4 Application description

## 4.1 Overview

FP-AUD-SMARTMIC1 is an advanced application based on STMicroelectronics digital MEMS microphones and STM32 microcontroller.

Designed for the evaluation and testing of STMicroelectronics hardware and firmware solutions, it embeds advanced audio processing routines performing Beamforming, Sound Source Localization, Acoustic Echo Cancellation, packed in easy-to-use and free acoustic libraries.

The whole system is able to communicate with a host PC using a specific serial protocol to control the system at run time to:

- tune algorithm parameters
- activate and deactivate functions
- display algorithm output in real-time.

The processed audio stream is sent to the host via a standard USB audio-IN class and can be saved using a standard audio recording tool (like Audacity) to evaluate the acoustic performance.

Audio output capabilities are enabled as well, allowing the evaluation of algorithms such as Beamforming and Acoustic Echo cancellation.

FP-AUD-SMARTMIC1 source code can be used as an example showing the adoption of the STMicroelectronics acoustic libraries and the communication engine. It can also be a starting point for the development of complex audio applications.

## 4.2 Firmware description

When an STM32 Nucleo board is used, the USB-to-serial-bridge embedded in the ST-LINK is adopted. For this reason, the firmware configures a UART to communicate with the bridge and the USB, exploiting an audio-IN USB device class that only exposes standard stereo microphones.

Two USB peripherals are used to communicate with the PC:

- the ST-LINK USB for message exchange via the USB to serial bridge;
- the main STM32 Nucleo USB for audio streaming.

When a BlueCoin board is used, a composite audio-IN/VCP USB class is used, allowing audio dataflow and serial communication on a single USB peripheral.

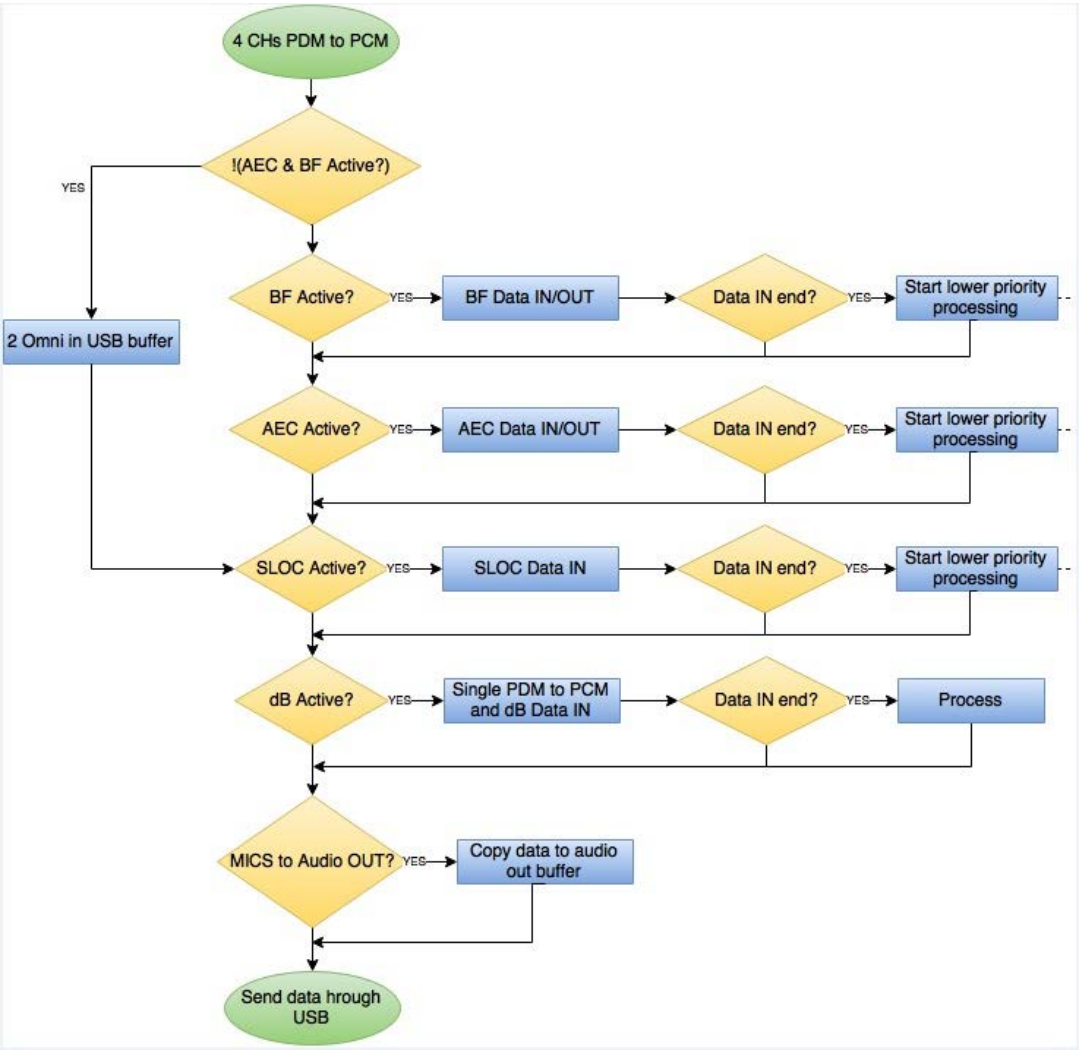
The whole audio application is driven by microphone acquisition and all functions are managed inside the relevant interrupt, thrown every 1 ms. This audio-IN interrupt handler routine calls the function `void audio-process(void)`, representing the entry point of all the audio-related operations, where 1 ms of PDM audio is available to the user.

The audio-process routine performs the high priority operations of the active algorithm, depending on the data structure status.

All high-priority/fast-executing routines (like library data input steps) are performed inside this routine, while the processor-intensive parts are managed in lower-priority tasks generated by software interrupts.

The figure below outlines the routine execution.

Figure 26. **Firmware flowchart**




## 4.3 Application status and serial communication protocol

During the application execution, the current status is stored in a data structure containing information such as the active algorithms, specific configurations and parameters (e.g., the beamforming direction or the source localization resolution), last computed results (e.g., the last angle retrieved by the source localization), etc.

This data structure, or a part of it, can be exchanged with the host through the communication infrastructure to change the application behavior at runtime or to retrieve relevant data.

The communication paradigm is based on the following data flow:

1. the host performs requests via specific commands;
2. the device sends a response.

*Note: The embedded application never sends information without a specific request coming from the host.*

To avoid race conditions and unexpected behavior, two instances of the data structure are used and kept aligned in the embedded application:

- one internal, used by the communication routine;
- a specific replica, modified by the user for relevant data fields (containing data calculated on the device itself, as the dB_noise output or sound source localization algorithms)

A periodic routine in the application layer is in charge of aligning the two data structures to ensure a consistent communication.

An easy-to-use implementation of the serial protocol is provided within the project for both the device (C source code) and the host (C++ source code).

### 4.3.1 The data structure

The following data structure is used to store the application status:

```
/**
* @brief Overall Status Structure definition
*/
typedef struct
{
GeneralParam_t GeneralStatus; /*!<General Status*/
SLocParam_t SLocStatus; /*!<SLoc Status*/
BeamParam_t BeamStatus; /*!<Beam Status*/
AECParam_t AECStatus; /*!<AEC Status*/
dBParam_t dBStatus; /*!<dbSPL Status*/
ASRParam_t ASRStatus; /*!<ASR Status*/
OutputParam_t OutputStatus; /*!<Out Status*/
}
AudioStatus_t;
```

The overall structure is split into a set of substructures, each one representing the status of a specific acoustic routine.

The following example shows the substructure for the AcousticSL (source localization):

```
/**
* @brief  Source Localization Status Structure definition
*/
typedef struct
{
  int16_t Angle;                    /*!<Computed angle*/
  uint8_t Algorithm;                /*!<Running algorithm*/
  uint8_t Resolution;               /*!<Current resolution*/
  int16_t Threshold;                 /*!<Current threshold*/
  uint32_t Reserved[4];             /*!< For future use*/
}
SLocParam_t;
```

Both the host and the device keep track of the application status, using a local copy of the overall structure, which can be sent (entirely or for a specific substructure) to the communication link to perform host-device communication.

### 4.3.2 Serial communication protocol: layers 1 and 2

The serial communication implemented in the demo is composed of multiple layers (from the top to the bottom):

- application layer
- serial protocol layer 1
- serial protocol layer 2

**Table 2. Serial protocol: layer 1 packet**

| Type | Encoded payload | EOF |
|---|---|---|
| Bytes | N | 1 |

The **Encoded Payload** is the message exchanged by the protocol between the module and a host. Its length can change but it is limited by the physical MTU (maximum transmission unit).

The actual payload is encoded by a bytestuffing function (see Section 4.3.5 Bytestuffing) to avoid any EOF character in its content.

**EOF (0xF0)** is a byte representing the end of the packet, whose value is 0xF0.

The layer 1 encoded payload must be processed by a function which performs the **reverseByteStuffing**, to obtain the payload. The resulting payload is a layer 2 packet.

**Table 3. Serial protocol: layer 2 packet**

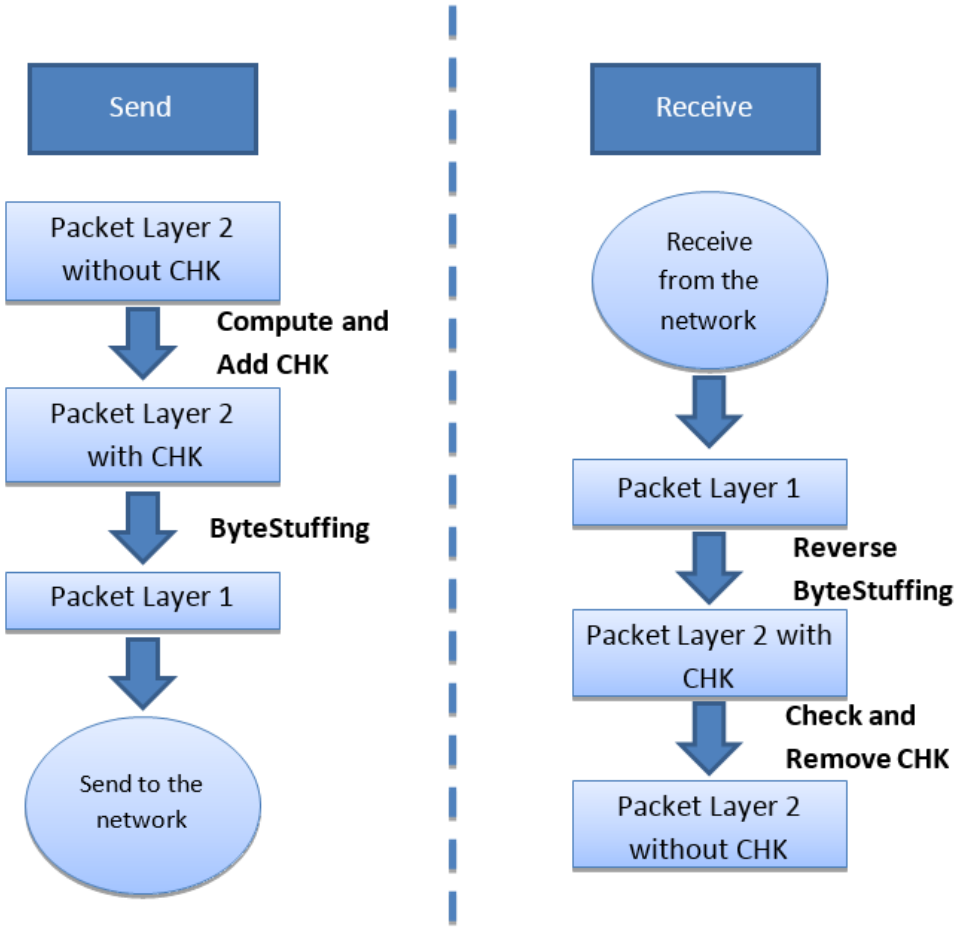
| Type | Destination address | Source address | Command (CMD) | Payload | CHK |
|---|---|---|---|---|---|
| Bytes | 1 | 1 | 1 | N | 1 |

The **Destination Address** is the destination symbolic address and enables unicast messages on a shared bus.
The **Source Address** is the source symbolic address.
The **Command** code represents the issued command and specifies how the payload should be interpreted.
The **Payload** data are interpreted with respect to the CMD field.
**CHK** represents the computed sum of all the bytes in the packet which must be equal to zero.

### 4.3.3 Send/receive packet process

The application for checksum and bytestuffing functions, and their reverses, ensures the correct interpretation of the CMD and its payload.

**Figure 27. Send/receive packet process**




### 4.3.4 Checksum algorithm

A checksum algorithm ensures that the packet you handle contains the correct information.

The algorithms needed to compute and verify the checksum integrity are explained by the following pseudo-code snippets.

### 4.3.5 Bytestuffing

Bytestuffing is a process that transforms a sequence of data bytes, which may contain 'illegal' or 'reserved' values, into a potentially longer sequence free of those values.

The EOF character identifies the end of the packet, therefore the packet must not contain any other occurrence of the character itself.

For this reason, the following special characters are defined:

- `TMsg_EOF (0xF0)` is the EOF of layer 1 packet
- `TMsg_BS (0xF1)` is the bytestuffing escape character
- `TMsg_BS_EOF (0xF2)` is the substitution for `TMsg_EOF`

In the bytestuffing algorithm (used in sending actions), given layer 2 message, for each character:

- `TMsg_BS` is replaced by `TMsg_BS` followed by `TMsg_BS` (which doubles the character) and `0xF1` becomes `0xF1 0xF1`;
- `TMsg_EOF` is replaced by `TMsg_BS` followed by `TMsg_BS_EOF` and `0xF0` becomes `0xF1 0xF2`.

As it can be seen, the layer 2 packet length may change in the process.

In the reverse bytestuffing algorithm, given a layer 1 payload, for each character:

- `TMsg_BS` followed by `TMsg_BS` is replaced by a single `TMsg_BS` and `0xF1 0xF1` becomes `0xF1`;
- `TMsg_BS` followed by `TMsg_BS_EOF` is replaced by `TMsg_EOF` and `0xF1 0xF2` becomes `0xF0`.

### 4.3.6 Serial commands

To ease the application development, the system implements a reduced number of commands.

The protocol can be extended to include more commands which are specifically related to the target application and can be divided in: generic, basic and application-specific commands.

**Table 4. Basic commands**

| Command | CMD value | Meaning |
|---|---|---|
| CMD_PING | 0x01 | This is the standard ping command; the device will reply accordingly. |
| CMD_Read_PresString | 0x02 | It requests the presentation string which contains basic device information (name and version) |
| CMD_Reset | 0x0F | It requests the device reboot |
| CMD_Reply_Add | 0x80 | This value is added to the CMD field value to create the command for the reply (e.g., 0x81 = reply to the PING command). |

**Table 5. Application specific commands**

| Command | CMD value | Meaning |
|---|---|---|
| CMD_AudioModule_SetStatus | 0x40 | It sets the new device status |
| CMD_AudioModule_GetStatus | 0x41 | It requests the device current status |

The commands shown in the above table specify the CMD field value which can be inserted in the request packet (from the host to the module). The module will reply adding **CMD_Reply_Add**to the CMD field value.

Note that the packet length is determined by a low level function using "terminator" and "escape" special characters. For this reason, the packet real length could not be equivalent to the message length.

Moreover, the data contained in the layer 2 payload are serialized before being copied in a packet so that, using a function to deserialize them, the data are independent from the architecture (big/little-endian).

The commands, in layer 2 format, are detailed below.

`CMD_PING`

**Table 6. `CMD_PING` packet**

| Type | Destination address | Source address | Command |
|---|---|---|---|
| Length (Bytes) | 1 | 1 | 1 |
| Value | XX | YY | CMD_PING |

The module sends the same packet format but source and destination addresses are obviously swapped and the command field contains `CMD_PING +CMD_Reply_Add`.

`CMD_Read_PresString`

The request packet is equal to `CMD_PING` where CMD is replaced by `CMD_Read_PresString`

**Table 7. `CMD_Read_PresString` packet**

| Type | Destination address | Source address | Command | Payload |
|---|---|---|---|---|
| Length (bytes) | 1 | 1 | 1 | K |
| Value | YY | XX | CMD_Read_PresString+ CMD_Reply_Add | String of K characters |

`CMD_Reset`

The request packet is equal to `CMD_PING` where CMD is replaced by `CMD_Reset`. In this application, the reset command is handled, but no device reboot is performed.

`CMD_Set_Status`

This packet is sent when the host wants to change the device status, for example, to change algorithms activation, parameters and general device behavior.

The request packet has the conventional layer 2 structure shown in the table below.

**Table 8. `CMD_Set_Status` packet**

| Type | Destination address | Source address | Command (CMD) | Payload | CHK |
|---|---|---|---|---|---|
| Value | XX | YY | CMD_Set_Status | N | 1 |

The N-length payload field can be further divided into 2 sections.

**Table 9. N-length payload field**

| Type | Domain mask | Payload |
|---|---|---|
| Bytes | 1 | N - 1 |

In the table above:

- **DomainMask** describes the data structure subsections to be included in the payload field; this allows sending only specific parts of the overall data structure to save communication bandwidth and address only specific functions.
- **Payload** represents the actual data to be sent

The module answers with a simple ack, shown in the table below.

**Table 10. `CMD_Set_Status` answer packet**

| Type | Destination address | Source address | Command |
|---|---|---|---|
| Length (bytes) | 1 | 1 | 1 |
| Value | YY | XX | CMD_Set_Status + CMD_Reply_Add |

`CMD_Get_Status`

This packet is sent when the host wants to retrieve data from the device (for example, to make the source localization data or noise estimation available to the host).

The request packet follows the same principle of the `CMD_Set_Status` command.

The layer 2 payload byte contains the **DomainMask** which, in this case, contains the subsections of the data structure the host wants to retrieve from the device.

**Table 11. `CMD_Get_Status` packet**

| Type | Destination address | Source address | Command (CMD) | Payload | CHK |
|---|---|---|---|---|---|
| Value | XX | YY | CMD_Get_Status | DomainMask | 1 |

The module answers with a message containing an ack and the relevant data, as shown in the table below.

**Table 12. `CMD_Get_Status` answer packet**

| Type | Destination address | Source address | Command | Payload |
|---|---|---|---|---|
| Length (bytes) | 1 | 1 | 1 | K |
| Value | YY | XX | CMD_Set_Status + CMD_Reply_Add | Application status |

## 4.4 Host PC source code example description

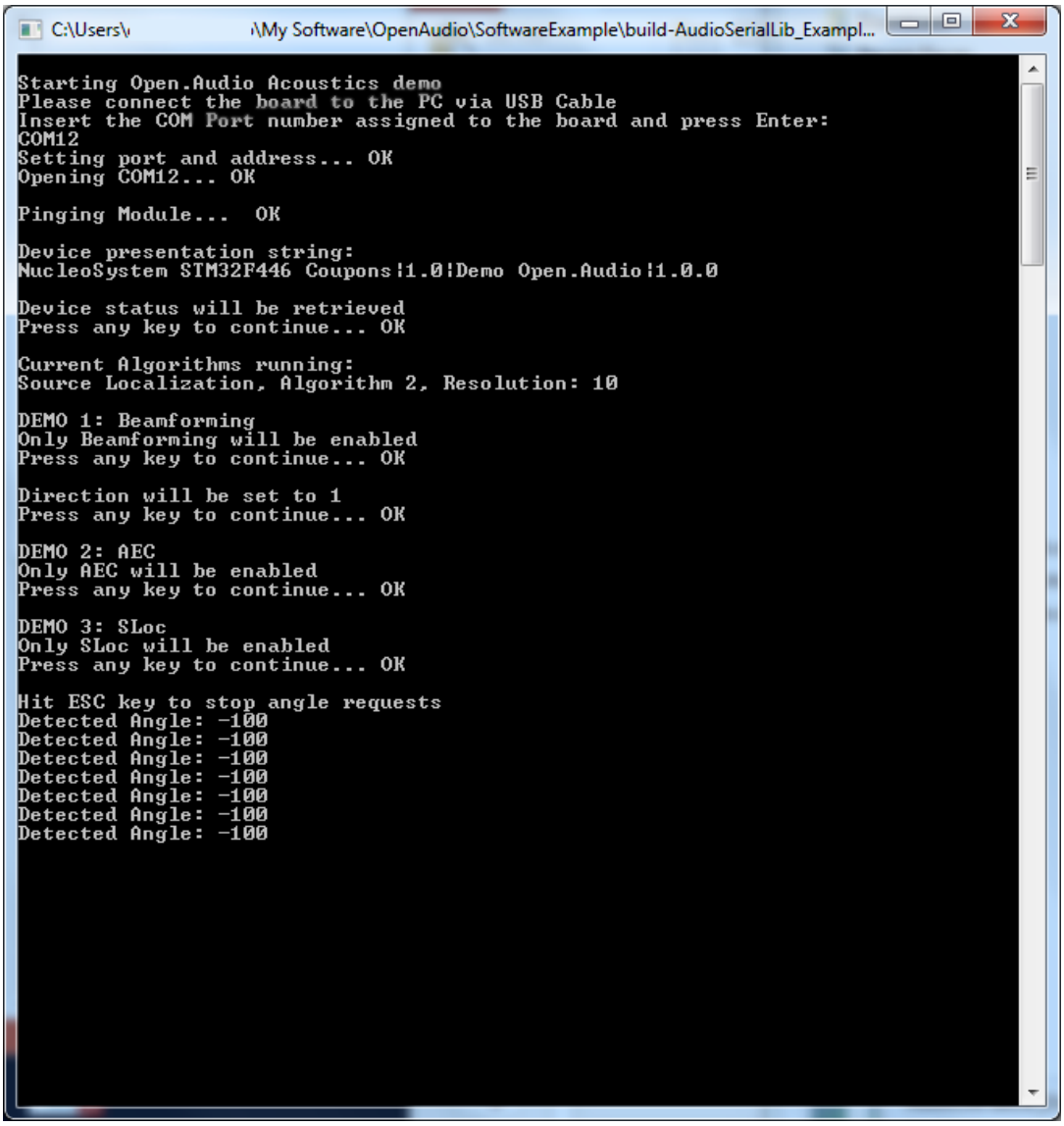
A C++ software example in source code is provided, performing a set of operations to remotely control the device. The device audio status is shown in the host to allow control of all the application features.

The software:

1. connects to the serial port specified by the user
2. retrieves a string describing the firmware version
3. retrieves the currently running algorithms
4. activates beamforming only
5. switches beamforming direction
6. activates AEC only
7. activates source localization and queries for the estimated direction

The software can be compiled via GCC or via QT Creator.

Figure 28. **Terminal window showing the running software**



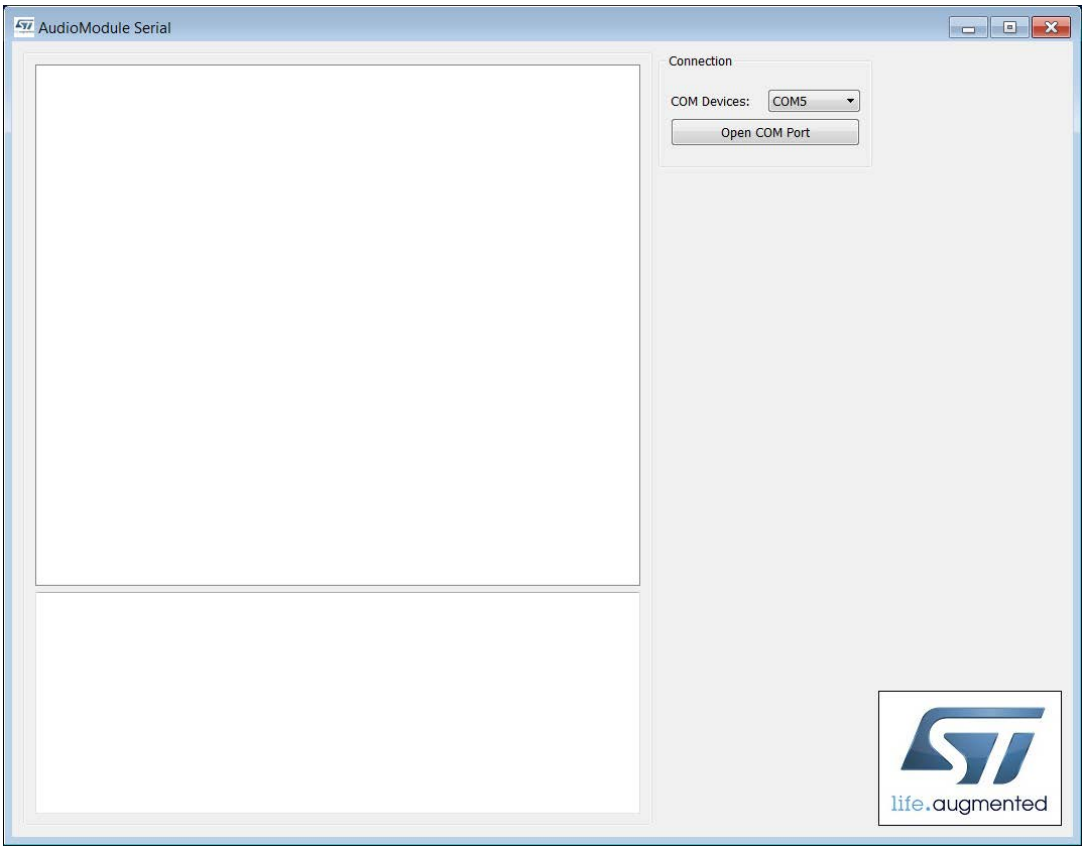

## 4.5 Host PC GUI application description

The current guide is based on a BlueCoin device, but the same information is valid for an STM32 Nucleo-based system.

A compiled software GUI is provided to allow the user to easily control the device and test all the functions in the standard demo.

**Step 1.** Double click on the executable to start the application.
The following window appears:

**Figure 29. GUI - initial page**

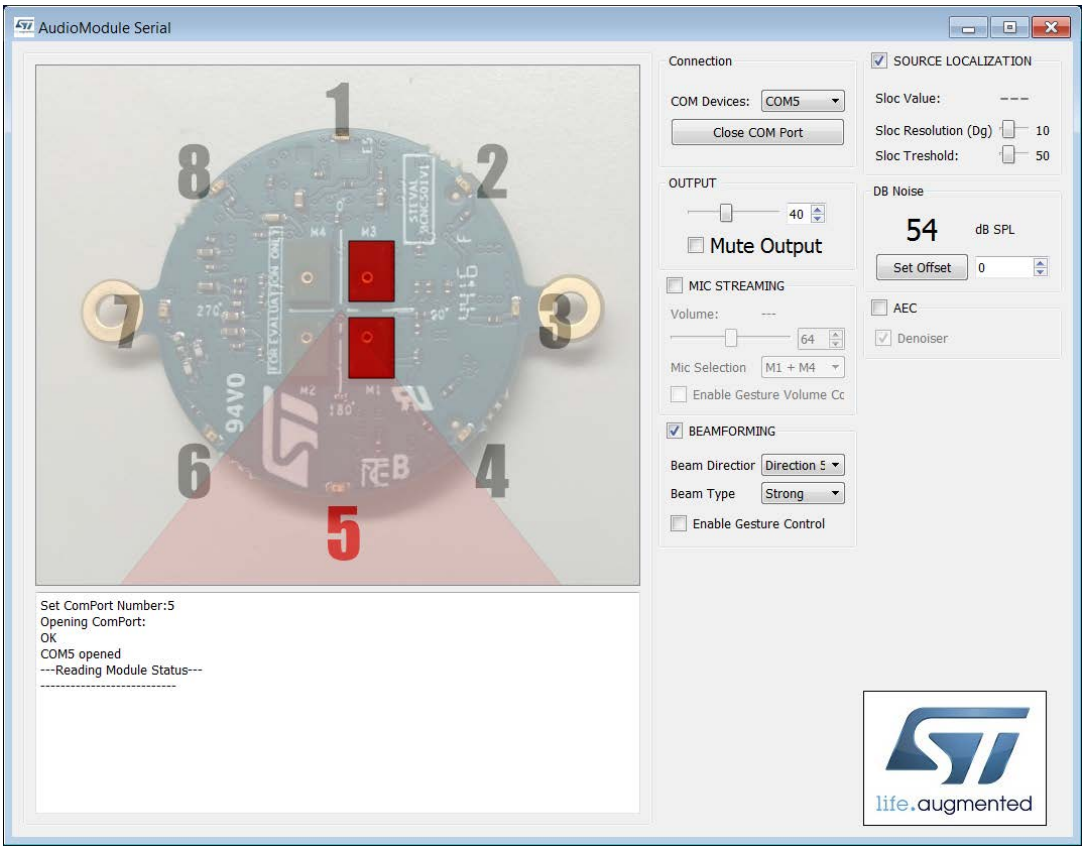



The correct STMicroelectronics COM port (recognized by the host PC as described in Section 3.2.2.2 Host PC configuration), should be automatically highlighted in the drop down menu.

**Step 2.** Click on the **Open COM Port** button to start the communication session and show the connected system current status.

A different image appears according to the connected hardware (STM32 Nucleo-based system or a BlueCoin kit).

**Figure 30. GUI - after COM opening**




When testing the application, you should use audio recording software to record and play back the processed signals (refer to Section 3.2.4.3 PC audio recording utility example: Audacity).

The board always sends two channels to the PC: the left one contains the processed audio, whereas the right one contains a single omnidirectional microphone to evaluate the difference with respect to the processed signals.

You can activate all possible functions and relevant parameters by clicking the specific checkbox.

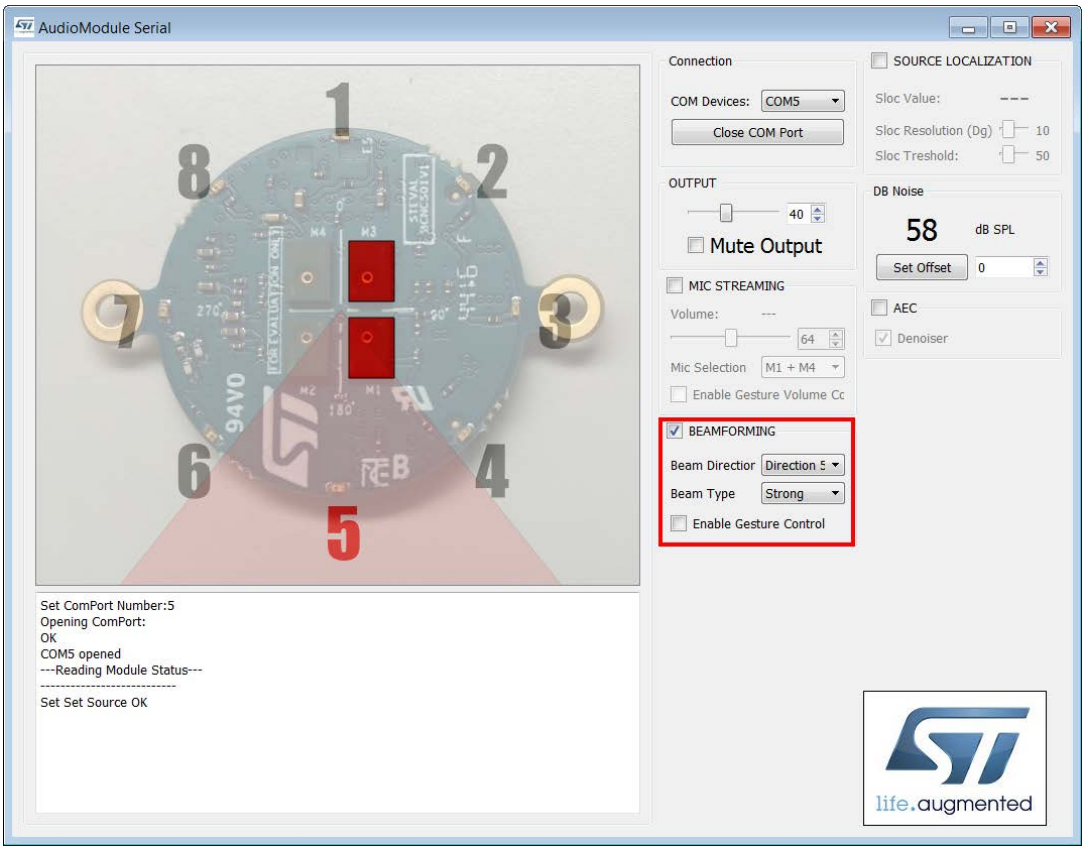
*Note: Some functions can be active at the same time.*

### 4.5.1 Beamforming

**Step 1.** Click on the **Beamforming** check box.

The beamforming algorithm starts and uses the audio acquired by two microphones to create a virtual microphone pointing in a specific direction.

**Figure 31. Beamforming**




The figure above shows the current direction and the microphones used by the algorithm: in this particular case, the direction is number 5, obtained using microphones 1 and 3.

The user can change direction through the **Beam Direction** dropdown menu: the resulting image reflects the user choice, highlighting the new direction and adopted microphones.

The user can also choose the beamforming type through the **Beam Type** dropdown menu, including all the levels provided by ST Acoustic Beam Forming library.

**Step 2.** Select the **Enable Gesture Control** checkbox.

Gesture recognition for beam direction selection starts.

Using the two Time-Of-Flight sensors mounted on the BlueCoin station, the system is able to recognize hand movement over the board: wiping the hand over the system from the right to the left (from direction 3 to 7) changes beam direction to 7; moving the hand in the opposite direction (from 7 to 3) results in the direction number 3

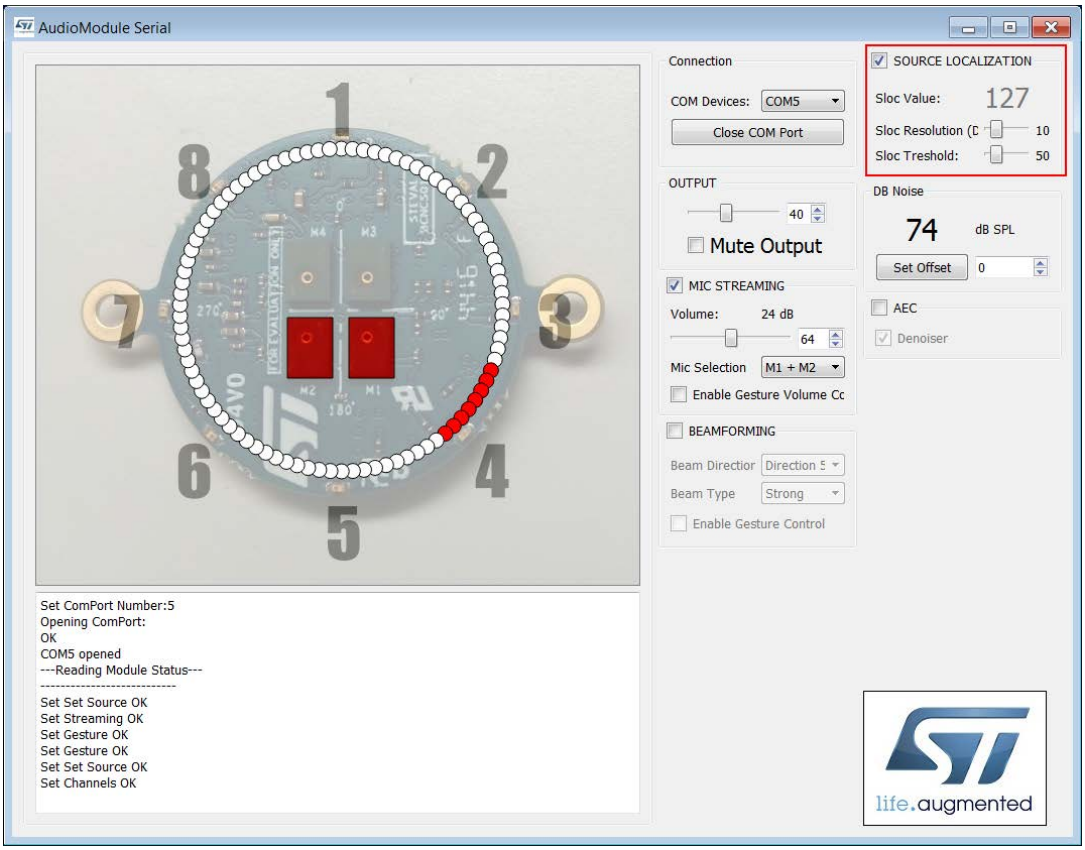
*Note: Gesture recognition is only available for BlueCoin based systems*

### 4.5.2 Sound source localization

**Step 1.** Click on the **Source localization** check box.

The sound source localization algorithm starts and estimates the direction of arrival of the main audio source, by using audio acquired by the four microphones.

**Figure 32. Sound source localization**




In the figure above, the red circles around the board show the estimated direction.

The current detected angle value and the microphones sent to the PC in that specific moment are also shown.

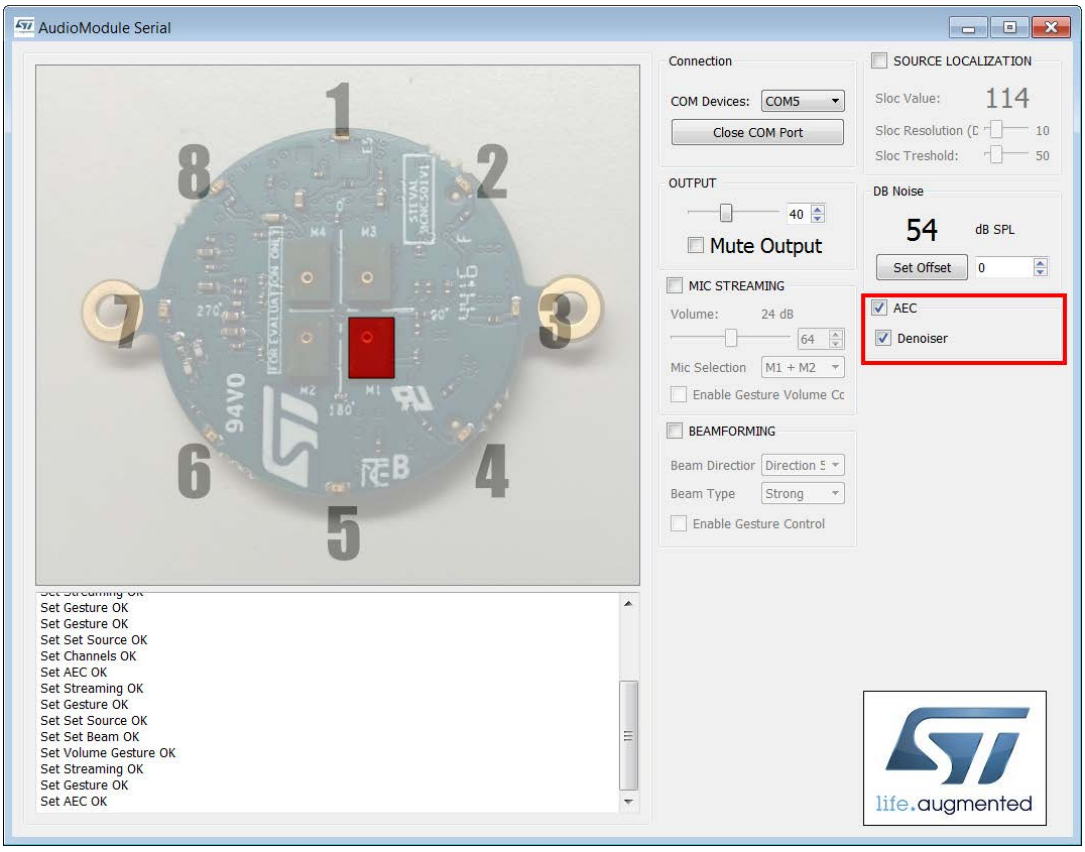
If Beamforming is active at the same time, beamforming information (direction and used microphones) is shown as well.

The user can change ST Acoustic Sound Source Localization parameters using the two sliders to modify the algorithm resolution and sensitivity settings.

### 4.5.3 Acoustic echo cancellation (AEC)

**Step 1.** Click on the **AEC** check box.
The acoustic echo cancellation algorithm starts.

**Figure 33. Acoustic Echo Cancellation**



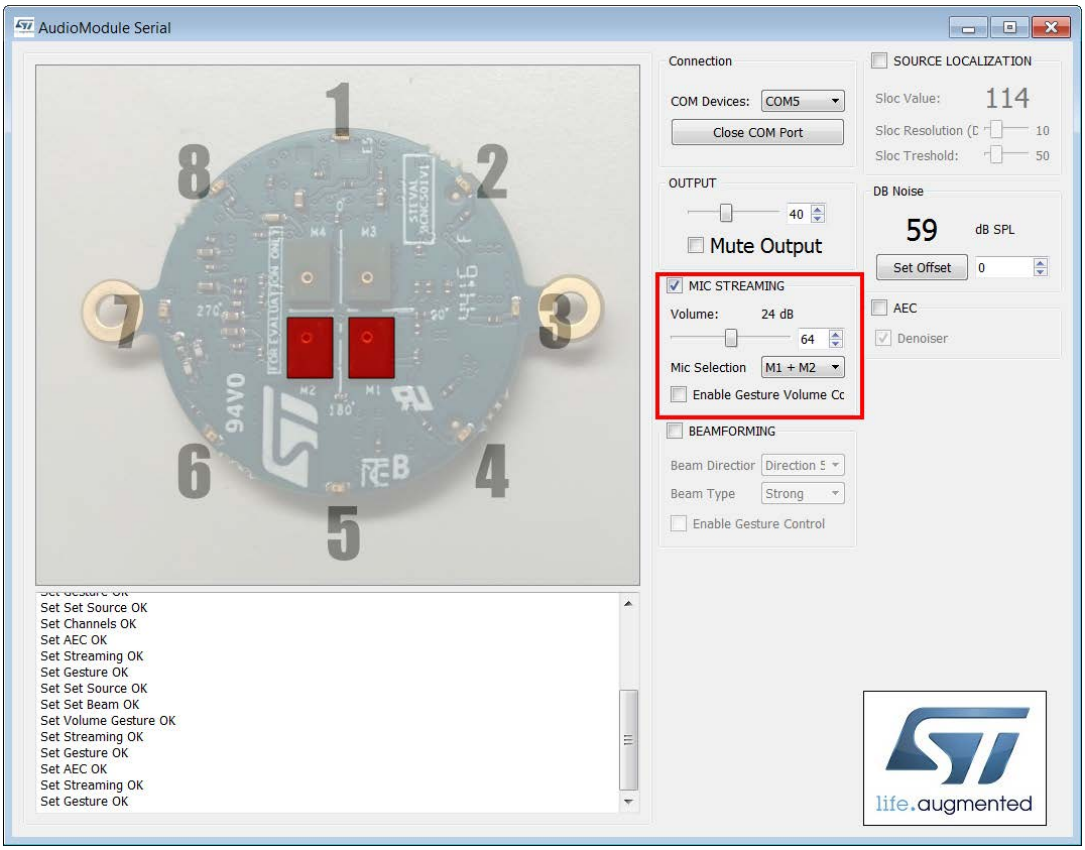

A song fragment located in the MCU flash is emitted by the loudspeaker, simulating the far end signal.
The system acquires microphone number 1 and applies echo cancellation to remove the emitted song from the microphone signal. A white noise removal function can be activated/deactivated selecting the specific checkbox.

### 4.5.4 Microphone streaming

**Step 1.** Click on the **MIC STREAMING** check box.

The omnidirectional microphone streaming becomes active. Beamforming or AEC algorithms, if active, will be automatically switched off.

**Figure 34. Microphone streaming**




The user can select the microphone to be sent via USB (two microphones out of four can be chosen) and the gain to be applied.

**Step 2.** Select the **Enable Gesture Volume Control** checkbox.

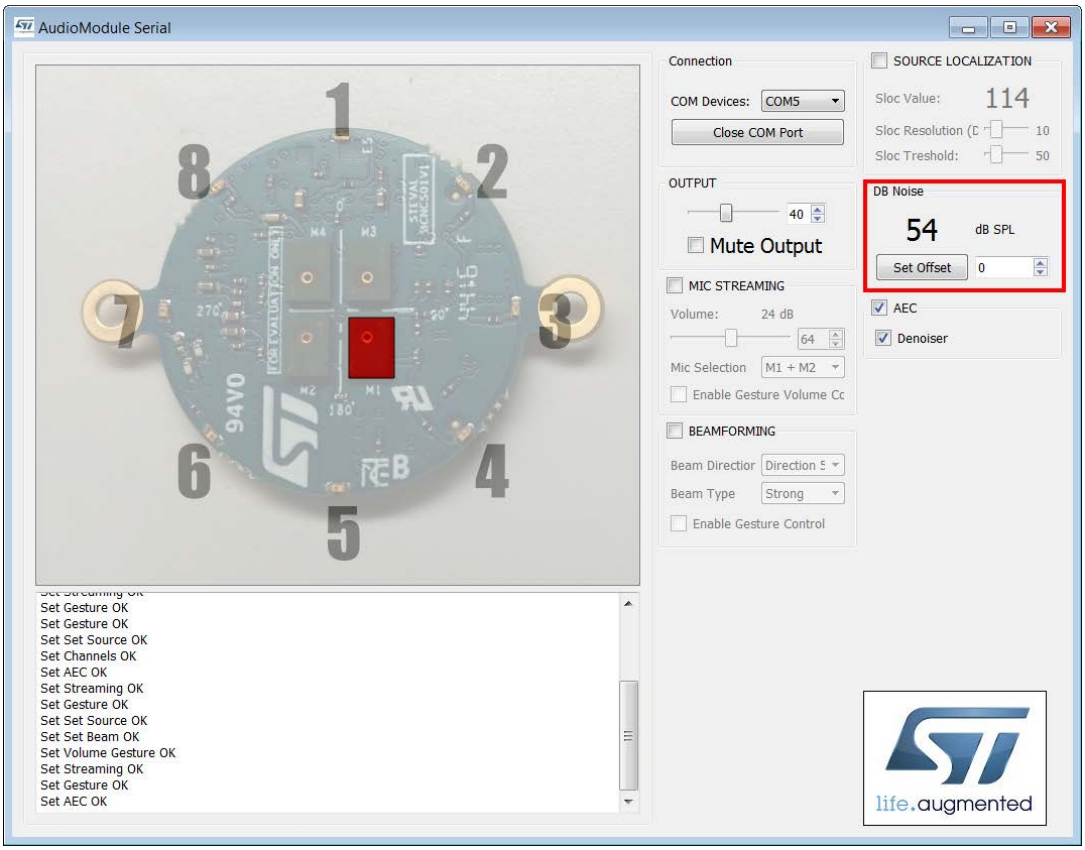
Gesture recognition for the gain becomes active: using the two Time-Of-Flight sensors mounted on the BlueCoin station, the system is able to recognize the distance between the speaker and the board (up to 50 cm) and control the gain accordingly.

*Note: Gesture recognition is only available for BlueCoin-based systems.*

### 4.5.5 dB noise

This feature is always active and performs an environmental noise level estimation using one of the four microphones (unweighted dB SPL). An offset can be added to calibrate the measure.

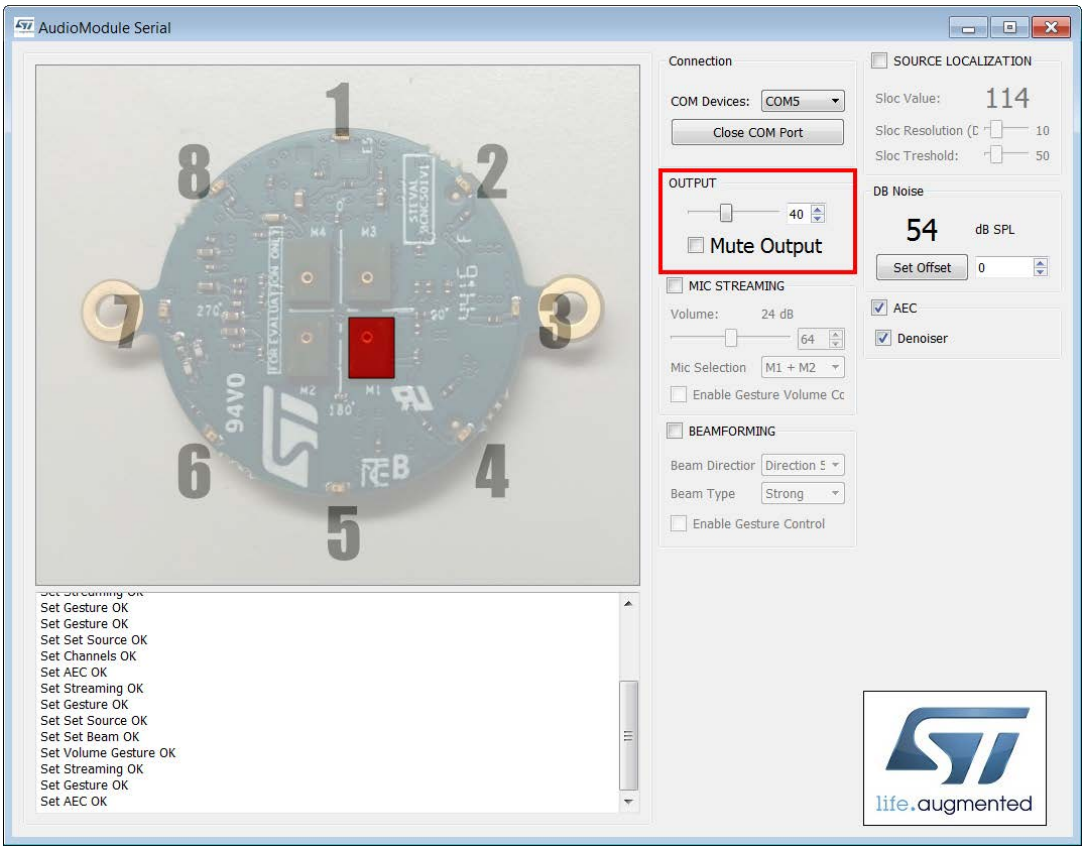
Figure 35. **dB SPL Estimation**




### 4.5.6 Output control

This section allows control of the loudspeaker volume.

Figure 36. **Output control**



# Revision history

Table 13. **Document revision history**

| Date | Revision | Changes |
|---|---|---|
| 12-Jun-2017 | 1 | Initial release. |
| 20-Apr-2018 | 2 | Updated title and Figure 1. FP-AUD-SMARTMIC1 software architecture. |
| 11-Oct-2018 | 3 | Added references to MP34DT06J microphone.<br>Added Section 3.2.4.2 Parameter optimization. |
| 24-Jan-2020 | 4 | Updated Introduction, Section 2.1 Overview, Figure 1. FP-AUD-SMARTMIC1 software architecture, Section 3.2.1 Hardware setup, Section 3.2.2.1 Hardware configuration and Section 3.2.4.1 Development tool chains and compilers.<br>Added Section 3.1.4 X-NUCLEO-CCA02M2 expansion board and Section 3.1.5 STEVAL-MIC001V1 evaluation board.<br>Added X-NUCLEO-CCA02M2 expansion board and STEVAL-MIC001V1 evaluation board compatibility information. |

# Contents

# List of tables

# List of figures