Getting started with the STM32Cube function pack for full-duplex voice streaming over Bluetooth low energy using Opus compression

Introduction

FP-AUD-BVLINK2 is an STM32Cube function pack that performs voice streaming over BLE in a full-duplex configuration using the advanced Opus compression algorithm. The application runs on the STM32 Nucleo and includes drivers and middleware for BLE (BlueNRG-MS) and digital MEMS microphones.

The software with the suggested combination of STM32 and ST devices can be used, for example, to develop wireless audio communication systems in smart home or wearable applications.

Thanks to the low bitrates you can achieve while maintaining high voice quality with Opus, combined with the low power features of BlueNRG, you can develop applications featuring very low consumption.

The same audio codec can be used to implement high quality stereo music streaming.

The software runs on the STM32 microcontroller and includes all the necessary drivers to use the devices on the STM32 Nucleo development board and expansion boards, as well as on the STEVAL-BCNKT01V1 and STEVAL-STLKT01V1 evaluation boards.

It also includes the Opus audio codec as the STM32Cube middleware.

UM2382 - Rev 1 - March 2018
For further information contact your local STMicroelectronics sales office.

www.st.com

1 Acronyms and abbreviations

Table 1. Acronyms and abbreviations

Term	Description
ATT	Attribute protocol
BLE	Bluetooth low energy
BSP	Board support package
GAP	Generic access profile
GATT	Generic attribute profile
HAL	Hardware abstraction layer
MEMS	Micro electro-mechanical systems
PCM	Pulse code modulation
PDM	Pulse density modulation
UUID	Universally unique identifier

2 FP-AUD-BVLINK2 software description

2.1 Overview

The key features of the FP-AUD-BVLINK2 package are:

- Complete firmware to implement full-duplex speech communication over Bluetooth low energy (BLE) using Opus compression
- A BlueVoiceOPUS customized profile for audio over BLE, including an easy-to-use set of APIs to exploit advanced Opus functionality (source code available)
- Third-party Opus v1.2.1 (downloadable from https://opus-codec.org/) middleware: an open, royalty-free and highly versatile audio codec that is standardized by the Internet Engineering Task Force (IETF) as RFC 6716
- Digital audio signal acquisition and processing
- Audio out playback through USB or jack connector
- Sample implementation available for X-NUCLEO-IDB05A1 plus X-NUCLEO-CCA02M1 connected to a NUCLEO-F446RE or NUCLEO-L476RG, for SensorTile (STEVAL-STLKT01V1) and BlueCoin (STEVAL-BCNKT01V1)
- Easy portability across different MCU families thanks to STM32Cube
- Free, user-friendly license terms

This software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for BlueNRG-MS, MEMS microphone expansion boards, SensorTile and BlueCoin; middleware components for audio acquisition, communication with other BLE devices, USB streaming of recorded signals and a dedicated profile for full-duplex speech transmission over BLE (BlueVoiceOPUS).

The third party Opus (v1.2.1) middleware is included in the function pack.

The BlueVoiceOPUS profile defines a BLE service which includes one characteristic for audio transmission and one for optional control message. In a full-duplex system, both sides of the communication (central and peripheral) can act as a server of information. Periodic notifications containing compressed audio data are sent from the central node acting as a server to the peripheral node acting as a client, and vice versa.

The BlueVoiceOPUS middleware is responsible for audio encoding and periodic data transmission on the server side and for decoding of received data on the client side.

The drivers abstract low-level hardware details and allow the middleware components and applications to access the devices in a hardware-independent fashion.

The package includes a sample application that developers can use to start experimenting with the code. It enables audio acquisition, compression and transmission over BLE from the module acting as a transmitter to the module acting as a receiver. The receiver is responsible for audio decompression and USB streaming of audio data to a PC. The system is recognized by the PC as a standard microphone, and any freeware or commercial audio recording software can be used to interface with it. Depending on the hardware used, the audio can be played through a jack connector. Both the central and the peripheral modules can act as a transmitter and a receiver at the same time, enabling full-duplex voice streaming.

2.2 Architecture

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

The FP-AUD-BVLINK2 provides a board support package (BSP) for the Bluetooth low energy (X-NUCLEO-IDB05A1) and the digital MEMS microphone expansion boards (X-NUCLEO-CCA02M1), as well as for the NucleoF4xx and NucleoL4xx.

It also includes BSP for SensorTile (STEVAL-STLKT01V1) and BlueCoin (STEVAL-BCNKT01V1) for a complete, platform-independent compatibility.

The BSP is provided with a set of middleware components for audio acquisition, compression and decompression, data transmission over BLE and USB audio in class implementation.

The application works with the following software layers:

- **STM32Cube HAL layer**: provides a simple, generic and multi-instance set of generic and extension APIs (application programming interfaces) to interact with the upper application, library and stack layers. These generic and extension APIs are built on a common architecture and allow layers built on top of them (like middleware) to implement functions without requiring specific microcontroller unit (MCU) hardware information. This structure improves library code reusability and guarantees easy portability across devices.

- **Board support package (BSP) layer**: supports the peripherals on the STM32 board, apart from the MCU. This limited set of APIs provides a programming interface for board-specific peripherals like LEDs and user buttons; it also helps in identifying the specific board version:
  – for the BLE expansion board (X-NUCLEO-IDB05A1), it provides a set of APIs for initialization and communication with the BlueNRG-MS component
  – for the microphone acquisition board (X-NUCLEO-CCA02M1), it provides APIs for audio acquisition and processing
  – for SensorTile (STEVAL-STLKT01V1) and BlueCoin (STEVAL-BCNKT01V1), it provides a set of APIs for BlueNRG-MS management, sensors and microphone acquisition. It also enables audio streaming by exploiting an audio DAC, mounted on the SensorTile motherboard (STEVAL-STLCX01V1) or on the Coin Station (STEVAL-BCNST01V1)

Figure 1. FP-AUD-BVLINK2 software architecture
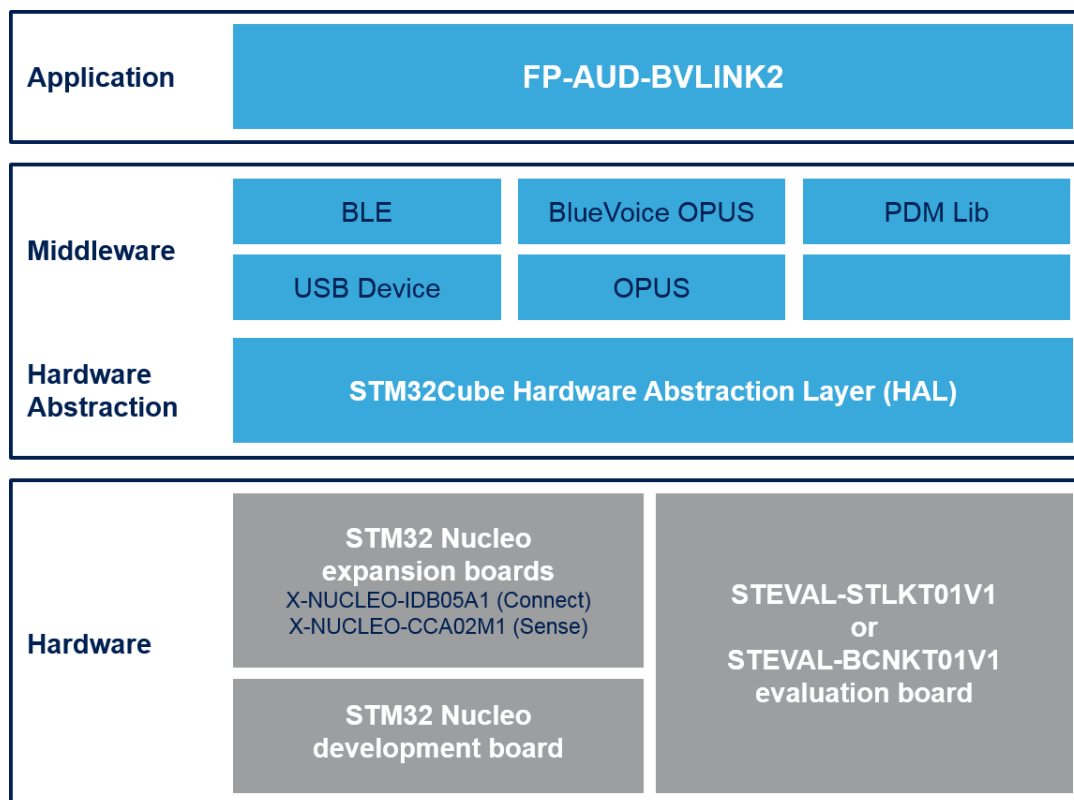
2.3 Folder structure
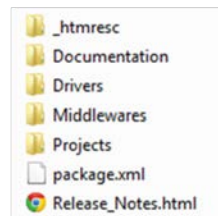
The following folders are included in the software package:

- **Documentation**: contains a compiled HTML file generated from the source code detailing the software components and APIs.
- **Drivers**: contains the HAL drivers, the board specific drivers for each supported board or hardware platform and the CMSIS vendor-independent hardware abstraction layer for the ARM® Cortex®-M processor series.
- **Middlewares**: contains libraries and protocols for the PDM-to-PCM conversion process, audio-input USB driver, BlueNRG Bluetooth low energy network module, the BlueVoiceOPUS profile and the Opus codec source code that can be downloaded from the opus official website (https://opus-codec.org/).
- **Projects**: contains central and peripheral applications to demonstrate voice transmission over BLE. The projects are supplied for the NUCLEO-F446RE, NUCLEO-L476RG, STEVAL-STLKT01V1 and STEVAL-BCNKT01V1 development board with the IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM) and Ac6 System Workbench for STM32 development environments.

2.4 APIs

Detailed technical information about the APIs available to the user can be found in a compiled HTML file located inside the "Documentation" folder of the software package.

2.5 Overall architecture

The FP-AUD-BVLINK2 processing components have been designed to create a wireless audio link between a trasmitter and a receiver module.

The whole speech processing chain, from the MEMS digital microphone acquisition to the playthrough via USB or I²S, is implemented and shown in Figure 3. FP-AUD-BVLINK2 complete speech processing chain.

On the Tx side, audio is acquired by a digital MEMS microphone as a 1-bit PDM signal and converted by a PDM to PCM conversion filter into a 16-bit PCM. Every time an audio frame is ready (according to the Opus encoder parameters), it is transferred to the compression algorithm; the encoded buffer size returned by the Opus encoder can significantly change accordingly to the compression parameters.

The Bluetooth low energy allows sending packets with a maximum size of 20 B. Since encoded bytes can be above this threshold, the compressed buffer must be split in multiple BLE packets. Moreover, the encoded buffer size can change every audio frame and the receiver has to know how long it is to rebuild it (for further information on the transfer protocol implemented for this scope, refer to Section 2.7.2.3 BlueVoiceOPUS transfer protocol).

As soon as the BLE packets are received by the Rx module, they are parsed to rebuild the encoded buffer; then, the Opus decoder returns the decompressed PCM audio data that can be sent to an audio out interface via USB or I²S.

The FP-AUD-BVLINK2 implements a full-duplex communication: both the central and the peripheral nodes act simultaneously as a transmitter and a receiver.

The sample application included in the functional pack is configured to acquire audio with a sampling frequency of 16 kHz, encode data each 20 ms (audio frame size) and stream them obtaining a bitrate of 16 kbps, an optimum compromise in terms of audio quality, radio transmission time and power consumption.

An easy-to-use set of APIs is available in the BlueVoiceOPUS middleware to customize every encoder and decoder parameter as explained in the following sections.

Figure 3. **FP-AUD-BVLINK2 complete speech processing chain**
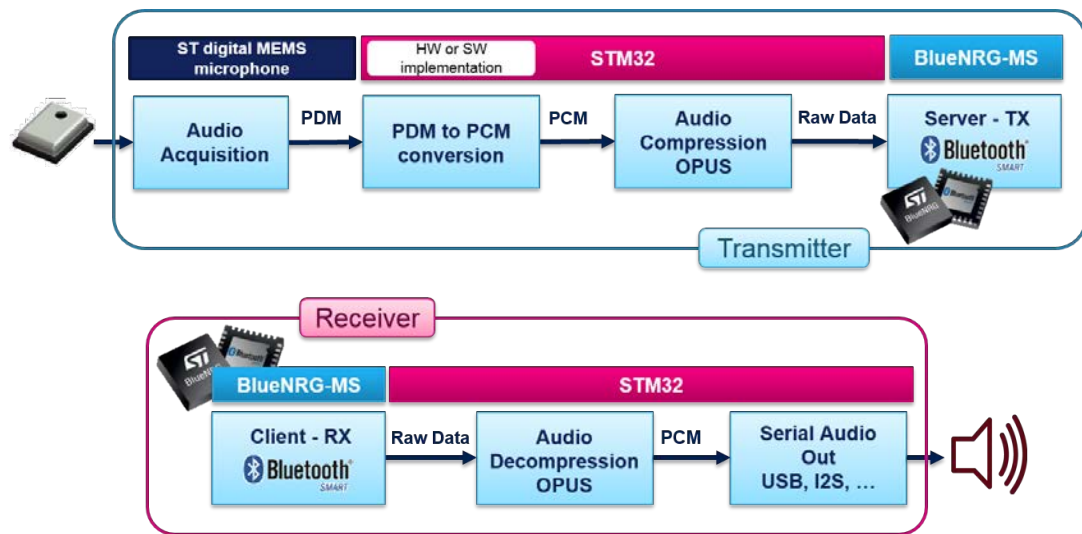
2.5.1 PDM filter

The digital MEMS microphone transmits digital signals in pulse density modulation (PDM) format. The analog signal from the MEMS microphone sensing element is amplified, sampled at a high rate and quantized in the PDM modulator, which combines the operations of quantization and noise shaping, giving as output a single bit at the high sampling rate. The noise shaping mechanism ensures a quantization of noise density inconstant over frequency. The resulting high-frequency one-bit signal has low quantization noise in the audio band and high noise at higher frequencies.

In a PDM signal, the amplitude of the original analog signal is represented by the density of pulses: full positive waveforms are all 1 and full negative waveforms are all 0.

Pulse code modulation (PCM) is chosen as an intermediate step between PDM and compressed audio data actually sent over BLE.

To convert the PDM stream to PCM data, decimation filters are typically used. Depending on the STM32 used, the following options are available:

1.   a software library (for NUCLEO-F446RE and STEVAL-BCNKT01V1);
2.   a hardware peripheral DFSDM (for NUCLEO-L476RG and STEVAL-STLKT01V1).

The conversion library (PDMLib) consists of a decimation filter converting 1-bit PDM data to PCM data, followed by two individually-configurable IIR filters (low pass and high pass).

The first stage of decimation is used to reduce the sampling frequency, followed by a high pass filter to remove the signal DC offset. The reconstructed audio is in 16-bit PCM format.

The DFSDM peripheral generates the clock needed by the microphones and reads data on the rising and falling edges of each PDM line.

Acquired signals are inputs for DSFDM filters for hardware filtering and decimation to generate standard PCM streams. An additional software high pass filtering stage removes any DC offset in the output stream. DMA is used to reduce the MCU load.

2.5.2 Opus

Opus is an open, royalty-free, highly versatile audio codec. It can be used for different kind of application as speech and music streaming or compressed audio storage. It is standardized by the Internet Engineering Task Force (IETF) as RFC 6716.

The scalability, from low bitrates narrowband speech at 6 kbit/s to a very high-quality stereo music at 510 kbit/s with low complexity, makes it suitable for a wide range of interactive applications.

It consists of two layers: one based on Linear Prediction (LP), the other based on the Modified Discrete Cosine Transform (MDCT).

Opus efficiently combines lossless and lossy results. For example, in speech applications, linear prediction techniques (such as Code-Excited Linear Prediction, or CELP) codify low frequencies more efficiently than in transform (for example, MDCT) domain techniques.

Opus codec consists of the SILK and CELT coding technologies. The former (initially developed by Skype) uses a prediction based model (LPC) whereas the latter (from Xiph.Org) is completely modelled on the MDCT transform. This versatility allows Opus to operate in three modes (SILK, CELT or hybrid mode) and ensures multiple configurations for different applications.

Opus can handle a wide range of audio applications, including Voice over IP, videoconferencing, in-game chat, and even remote live music performances. It can scale from low bitrate narrowband speech to very high quality stereo music. Supported features are:

- Bitrates from 6 kbps to 510 kbps
- Sampling rates from 8 kHz (narrowband) to 48 kHz (fullband)
- Frame sizes from 2.5 ms to 60 ms
- Support for constant bitrate (CBR) and variable bitrate (VBR)
- Audio bandwidth from narrowband to fullband
- Support for speech and music
- Support for mono and stereo
- Support for up to 255 channels (multistream frames)
- Dynamically adjustable bitrate, audio bandwidth and frame size
- Good loss robustness and packet loss concealment (PLC)
- Floating point and fixed-point implementation

In the FP-AUD-BVLINK2 an Opus encoder and a decoder are created for the central and the peripheral module. The package implements a full duplex communication and the audio parameters selected are the same for both nodes.

The default parameters for the BlueVoiceOPUS middleware are:

- for the encoder:
    – Opus application type: OPUS_APPLICATION_VOIP
    – Bitrate: 16000 bps
    – Channels: 1 (mono)
    – Opus complexity: 0
    – Audio frame size: 20 ms
    – Audio sampling frequency: 16000 kHz
- for the decoder:
    – Bitrate: 16000 bps
    – Channels: 1 (mono)
    – Audio frame size: 20 ms
    – Audio sampling frequency: 16000 kHz

2.6 BlueVoiceOPUS profile description

2.6.1 Generic access profile (GAP)

Bluetooth low energy communication can be either broadcast or connection-based.

The FP-AUD-BVLINK2 application deploys a connection-based communication paradigm providing a permanent point-to-point link between two devices. Data sent through a BLE connection are organized through an additional protocol layer, the Generic Attribute Profile (GATT).

The Bluetooth Specification v 4.1 sets the following device roles:

- **Central** role supporting multiple connections and starting connections with peripheral devices. These devices require a controller that supports the master role with more complex functions.
- **Peripheral** role for devices supporting a single connection and less complexity. These devices only require a controller that supports the slave role and use the central controller frequency to exchange data.

Figure 4. BlueVoiceOPUS Profile master-slave GAP role assignment

Central and peripheral role assignments are related to the asymmetric design concept of BLE: a slave cannot start complex procedures, whereas a master manages communication timing, adaptive frequency hopping, encryption setting, and so on.

Note: *According to the specification, data can be sent independently by either device at each connection event and the roles do not impose restrictions in data throughput or priority. In a full-duplex communication scheme, BlueVoiceOPUS role assignment is therefore decoupled via transmitter or receiver functionality.*

2.6.2 Generic attribute profile (GATT)

The Bluetooth SIG GATT specification provides standard profiles to ensure interoperability among different vendor devices whose features are, for example, Proximity Profile, Glucose Profile and Health Thermometer Profile.

The Bluetooth specification also lets you add custom profiles for new features.

GATT defines client and server roles for interacting devices independent of the GAP master/central and slave/ peripheral roles:

- **Client** performs service discovery about the presence and nature of server attributes; it sends requests to a server and accepts responses and server-initiated updates.
- **Server** accepts requests, commands and confirmations from a client and sends responses and server-initiated updates; it arranges and stores data according to the attribute (ATT) protocol.

In a mono-directional audio streaming asymmetric system, the device with voice data is the one with a microphone and is therefore considered the server. The client device sends requests to the server and accepts server-initiated updates containing audio data.

In a bidirectional system, where voice signals travel in either direction, the architecture is symmetric. The central and peripheral modules (with a microphone) may act as servers as well as clients sending requests and accepting updates.

Figure 5. BlueVoice Profile GATT role assignment in a bidirectional system
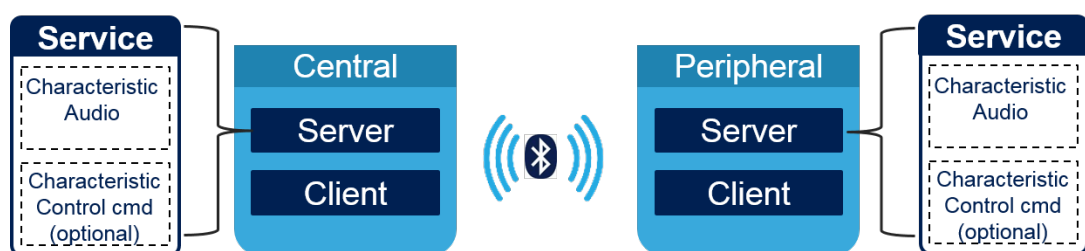

In both directions, streaming audio data transmission is based on periodic server-to-client notifications which do not require a request or response from the receiving device. Server-initiated updates are sent as asynchronous notification packets which include the handle of a characteristic value attribute along with its current value.

2.6.3 BLE communication

According to the BLE specification, the peripheral enters advertising mode at start-up and sends advertisement packets at relatively long intervals. The central unit enters discovery mode and sends a connection request on reception of an advertisement packet from the slave device. After the connection is created, notifications carrying audio data are periodically sent from the server to the client, according to the selected direction: peripheral-to-central, central-to-peripheral or simultaneously in both ways.

Figure 6. BLE connection setup
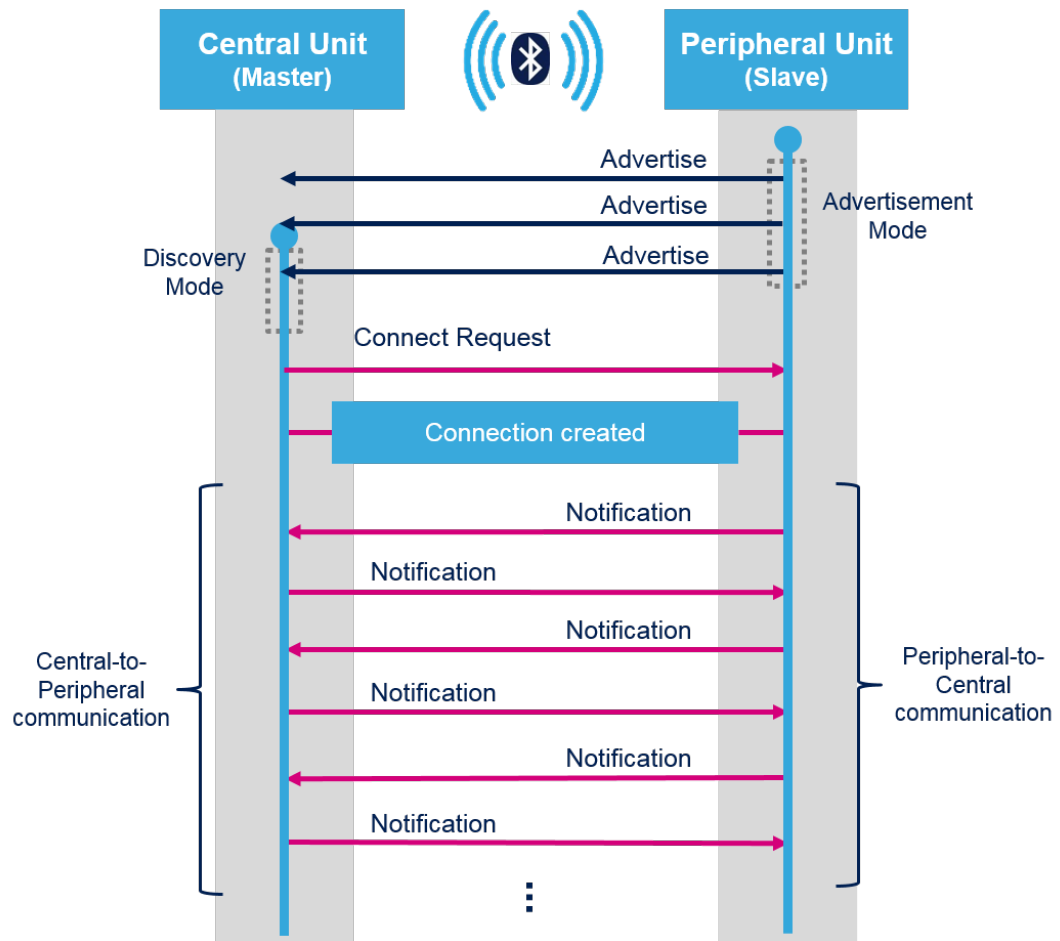
2.6.4 BlueVoiceOPUS service

The Attribute Protocol (ATT) is used by GATT as the transport protocol for exchanging data among devices. The smallest entities defined by ATT, named attributes, are addressable pieces of information that may contain user data or meta-information regarding the architecture of the attributes themselves, as stored in the server and as exchanged between client and server.

Attributes are described in the following fields:

- **Handle**: a unique 16-bit identifier for each attribute on a particular GATT server; it makes each attribute addressable, and it is guaranteed not to change.
- **Type**: a 16-, 32-, or 128-bit UUID (universally unique identifier) that determines the kind of data present in the value of the attribute. Apart from standard and profile UUIDs, proprietary and vendor-specific UUIDs can also be used in custom implementations like BlueVoiceOPUS.
- **Permissions**: metadata describing ATT data access permissions, encryption, and authorization.
- **Value**: the actual data content of the attribute; this is the part of an attribute that a client can access (if permitted) to both read and write.

GATT server attributes are organized as a sequence of services, each one starting with a service declaration attribute marking its beginning. Each service groups one or more characteristics and each characteristic can include zero or more descriptors.

Since audio streaming is not part of the predefined set of profiles, the FP-AUD-BVLINK2 defines a vendor-specific service named BlueVoiceOPUS service which exposes a user voice to a client device.

Table 2. BlueVoiceOPUS service definition

Type	Handle	UUID	Permissions	Value
Service	Att1	0x2800	READ	bv_opus_service_uuid
Audio characteristic	Att1	0x2803	READ	NOT|0x0013|Audio UUID
	Att2	bv_opus_audio_char_uuid	NONE	Audio data
	Att3	0x2902	READ/WRITE	Client characteristic configuration
Sync characteristic	Att1	0x2803	READ	NOT|0x0016|Ctrl UUID
	Att2	bv_opus_ctrl_char_uuid	NONE	Ctrl data
	Att3	0x2902	READ/WRITE	Client characteristic configuration

As shown in th table above, the BlueVoiceOPUS service is described by the following attributes:

- Att1 contains the service declaration for the BlueVoiceOPUS service with:
  - UUID: standard 16-bit UUID for a primary service declaration, UUID primary service (0x2800).
  - Permissions: Read.
  - Value: the proprietary 128-bit UUID for the BlueVoiceOPUS Service (UUID: 00000000-0001-11e1-9ab4-0002a5d5c51b).

The BlueVoiceOPUS service consists of an **Audio characteristic** to expose actual compressed audio data and a **Ctrl characteristic** to expose control information used to implement optional commands:

- **Audio Characteristic**
  - Att1 contains the Audio characteristic declaration. Its attribute fields are:
    - UUID: standard 16-bit UUID for a characteristic declaration, UUID characteristic (0x2803).
    - Permissions: Read.
    - Value: the properties for this characteristic are "notify only" and the UUID is for Audio Data (UUID: 00001000-0001-11e1-ac36-0002a5d5c51b).
  - Att2 contains the characteristic value (in this case, audio data). Its attribute fields are:
    - UUID: the same UUID in the last 16 bytes of the characteristic definition attribute value.
    - Permissions: None.
    - Value: the actual audio data.
  - Att3 contains the client characteristic configuration, defining how the characteristic may be configured by a specific client. Its attribute fields are:
    - UUID: the UUID is the standard 16-bit UUID for a client characteristic configuration (0x2902).
    - Permissions: Read/Write.
    - Value: Bit 0 Notifications disabled/enabled; Bit 1 Indications disabled/enabled
- **Ctrl Characteristic**
  - Att1 contains the control characteristic declaration. Its attribute fields are:
    - UUID: the UUID is the standard 16-bit UUID for a characteristic declaration, UUID characteristic (0x2803).
    - Permissions: Read.
    - Value: the properties for this characteristic are notify only and the UUID is for Ctrl Data (UUID: 00002000-0001-11e1-ac36-0002a5d5c51b).
  - Att2 contains the characteristic value, in this case control data. Its attribute fields are:
    - UUID: the same UUID present in the last 16 bytes of the characteristic definition's attribute value.
    - Permissions: None.
    - Value: the actual control data.
  - Att3 contains the client characteristic configuration, defining how the characteristic may be configured by a specific client. Its attribute fields are:
    - UUID: standard 16-bit UUID for a client characteristic configuration (0x2902).

- ◦ Permissions: Read/Write.
- ◦ Value: Bit 0 Notifications disabled/enabled; Bit 1 Indications disabled/enabled.

Table 3. BlueVoiceOPUS UUID summary table

UUID name	UUID
bv_opus_service_uuid	00000000-0001-11e1-9ab4-0002a5d5c51b
bv_opus_audio_char_uuid	00001000-0001-11e1-ac36-0002a5d5c51b
bv_opus_ctrl_char_uuid	00002000-0001-11e1-ac36-0002a5d5c51b

2.7 BlueVoiceOPUS middleware description

2.7.1 Overview

BlueVoiceOPUS is an STM32 middleware which implements a vendor-specific profile for audio streaming over BLE, not included in the standard Bluetooth low energy profiles. Bluetooth 4 is characterized by a lower bitrate with respect to its former version; so, an audio codec is needed to fit the acquired data in the available BLE bandwidth.

BlueVoiceOPUS is implemented as ST middleware ready to be integrated in projects for STM32 connected to a BlueNRG-MS Bluetooth low energy network module.

The software is provided as source code and requires two more modules: the STM32 Bluetooth low energy middleware included in the X-CUBE-BLE1 software expansion package, and the official Opus audio codec package (available at https://opus-codec.org/) included in the FP-AUD-BVLINK2 as a third-party middleware.

BlueVoiceOPUS includes:

- bluevoice_opus_service: the core of BlueVoiceOPUS that manages all the functions related to BLE, from the creation of the service and characteristics to the data sending and receiving.
- bluevoice_opus_interface: implements an interface between Opus codec and the BlueVoiceOPUS service, providing an easy-to-use set of APIs for Opus initialization, configuration, data compression and decompression.
- bluevoice_opus_transfer_protocol: manages the packetization of the encoded data in BLE packet and the parsing of the receiving data (for further information see Section 2.7.2.3 BlueVoiceOPUS transfer protocol).

2.7.2 How to use

2.7.2.1 *Initialization and configuration*

The BlueVoiceOPUS middleware can be used to implement a transmitter, a receiver or both in case of a full-duplex communication.

To be able to stream audio, BlueVoiceOPUS service and characteristics must be created by calling `BluevoiceOPUS_AddService` and `BluevoiceOPUS_AddChar` functions; both APIs require the UUIDs (chosen by the user) and return the relevant handlers.

The handle of the service must be passed to the `BluevoiceOPUS_AddChar` function. Alternatively, BlueVoiceOPUS characteristics can be added to a pre-existing service by calling `BluevoiceOPUS_AddChar` and passing the handle of that particular service as a parameter. If both functions returns `BV_OPUS_SUCCESS`, the BLE profile has been created correctly.

It is also possible to create the characteristics at user application level and pass the relevant handles using a structure `BV_OPUS_ProfileHandle_t` to the `BluevoiceOPUS_SetTxHandle` API. In the latter case, the following characteristics must be created:

- one related to the compressed audio data: `aci_gatt_add_char(ServiceHandle, UUID_TYPE_128, CharAudioUUID, 20, CHAR_PROP_NOTIFY, ATTR_PERMISSION_NONE, GATT_DONT_NOTIFY_EVENTS, 16, 1, CharAudioHandle);`

- one sending control information and command: `aci_gatt_add_char(ServiceHandle, UUID_TYPE_128, CharCtrlUUID, 20, CHAR_PROP_NOTIFY, ATTR_PERMISSION_NONE, GATT_DONT_NOTIFY_EVENTS, 16, 1, CharCtrlHandle);`

If the module has only to decode the received audio and will never stream data, the previous part is not necessary.

Then, Opus codec must be configured. According to the requested functions, an encoder and/or a decoder can be created by filling the relevant structure, `BV_OPUS_ENC_ConfigTypeDef` and `BV_OPUS_DEC_ConfigTypeDef`.

The default parameters used in the FP-AUD-BVLINK2 application are:

```
EncConfigOpus.application = OPUS_APPLICATION_VOIP;
EncConfigOpus.bitrate = 16000;                          /* bps */
EncConfigOpus.channels = AUDIO_CHANNELS_IN;             /* 1 channel, mono*/
EncConfigOpus.complexity = 0;
EncConfigOpus.ms_frame = AUDIO_IN_MS;                   /* 20 ms */
EncConfigOpus.sample_freq = AUDIO_IN_SAMPLING_FREQUENCY;  /* 16000 Hz */

DecConfigOpus.bitrate = 16000;                          /* bps */
DecConfigOpus.channels = AUDIO_CHANNELS_OUT;            /* 1 channel, mono*/
DecConfigOpus.ms_frame = AUDIO_OUT_MS;                  /* 20 ms */
DecConfigOpus.sample_freq = AUDIO_OUT_SAMPLING_FREQUENCY;  /* 16000 Hz */
```

Depending on the parameters chosen, the amount of memory returned by the relevant API must be allocated and passed to the configuration structure as:

```
uint32_t enc_size = BluevoiceOPUS_ENC_getMemorySize(&EncConfigOpus);
EncConfigOpus.pInternalMemory = (uint8_t *)malloc(enc_size);

uint32_t dec_size = BluevoiceOPUS_DEC_getMemorySize(&DecConfigOpus);
DecConfigOpus.pInternalMemory = (uint8_t *)malloc(dec_size);
```

Finally the encoder and decoder can be initialized by calling `BluevoiceOPUS_ENC_Init(&EncConfigOpus, &opus_err)` and `BluevoiceOPUS_DEC_Init(&DecConfigOpus, &opus_err)`. If both functions return `BV_OPUS_SUCCESS`, the BlueVoiceOPUS profile has been correctly configured.

If the initialization function return `BV_OPUS_INVALID_PARAM`, one or more parameters are not correct. The supported parameters are:

- application: `OPUS_APPLICATION_VOIP`, `OPUS_APPLICATION_AUDIO`, `OPUS_APPLICATION_RESTRICTED_LOWDELAY`
- bitrate [bps]: from 6000 to 510000
- channels: from 1 to 255
- complexity: from 0 to 10
- ms_frame [ms]: 2.5, 5, 10, 20, 40, 60
- sample_freq [Hz]: 8000, 12000, 16000, 24000, 48000

Other features can be set by calling specific BlueVoiceOPUS interface APIs:

- `BluevoiceOPUS_ENC_Set_VBR`: sets a variable bitrate (set by default)
- `BluevoiceOPUS_ENC_Set_CBR`: sets a constant bitrate
- `BluevoiceOPUS_ENC_Force_CELTmode`: forces Opus to use just the Celt codec (by default an hybrid mode is set)
- `BluevoiceOPUS_ENC_Force_SILKmode`: it forces Opus to use the Silk codec only (by default an hybrid mode is set)
- `BluevoiceOPUS_ENC_Set_Complexity`: it sets a new complexity without reinitialize the encoder
- `BluevoiceOPUS_ENC_Set_Bitrate`: it sets a new bitrate for the encoder

Note: *After calling this function, it is important to reallocate the encoder internal memory as explained above.*

The encoder and decoder can be deinitialized by calling `BluevoiceOPUS_ENC_Deinit` or `BluevoiceOPUS_DEC_Deinit`.

At application level, three callbacks must be called when the corresponding BlueNRG-MS event occurs:

- `BluevoiceOPUS_ConnectionComplete_CB` must be called when an `EVT_LE_CONN_COMPLETE` event occurs; it sets the connection handle.
- `BluevoiceOPUS_DisconnectionComplete_CB` must be called when an `EVT_DISCONN_COMPLETE` event occurs; it resets internal parameters.
- `BluevoiceOPUS_AttributeModified_CB` must be called when an `EVT_BLUE_GATT_ATTRIBUTE_MODIFIED` event occurs.

2.7.2.2   *Full-duplex audio transmission*

After connection setup, the module (Rx or Tx), which has discovered the BlueVoiceOPUS profile exposed by the other module, must enable the control notification by calling the `BluevoiceOPUS_EnableCtrl_Notif(rx_handle` API. The relevant handle must be passed to the function. The control notification will then be used to request start and stop streaming.

To start audio streaming, the transmitter module has to request the receiver to enable its audio notification by calling `BluevoiceOPUS_SendEnableNotifReq`. This API sends a notification, through the control characteristic, containing two bytes (`{BV_OPUS_FULLDUPLEX_ID, BV_OPUS_ENABLE_NOTIF_REQ}`). As soon as a node receives the request, it can enable the audio notification to the requestor, using the function `BluevoiceOPUS_EnableAudio_Notif(rx_handle);` with the relevant handle.

If the audio notification is correctly enabled, the module can start streaming audio. To set a full-duplex transmission up, the same procedure must be performed on both nodes.

BlueVoiceOPUS profile accepts, as input, an amount of PCM samples equal to the audio frame size set during Opus configuration. Every time an audio frame is ready, the API `BluevoiceOPUS_SendAudioData` should be called and it will automatically compress, packetize (see Section 2.7.2.3 BlueVoiceOPUS transfer protocol for further information) and send audio data.

For each audio notification received, `BluevoiceOPUS_ParseData` must be called and the status returned should be checked. In case of success, the `pcm_samples` parameter indicates if a complete audio frame is ready.

The parse function automatically rebuilds the Opus encoded frame, if it has been split into several BLE packets, and decodes it when it is ready.

The decompressed audio is available as PCM samples in the output buffer passed to the API.

2.7.2.3   *BlueVoiceOPUS transfer protocol*

By default, Opus encoder is configured with a variable bitrate, every encoded frame has a variable length according to the bitrate set during the initialization phase.

Since the maximum payload for the BLE packet is set to the highest limit of 20 Byte, the number of BLE packets where the encoded data fit may vary among different audio frames or depending on the Opus configuration.

The receiver must know the length of the compressed data to decode it; to this scope, a simple transfer protocol has been implemented (bluevoice_opus_transfer_protocol.c).

The BlueVoiceOPUS transfer protocol has the purpose of indicating when the encoded data starts and ends so that the receiver is able to rebuild the compressed buffer and decode it. To this goal, a single byte is added as first byte of each BLE packet, the remaining 19 bytes are filled with Opus encoded data.

The header byte can be one among list below:

- `BV_OPUS_TP_START_PACKET = 0x00`
- `BV_OPUS_TP_START_END_PACKET = 0x20`
- `BV_OPUS_TP_MIDDLE_PACKET = 0x40`
- `BV_OPUS_TP_END_PACKET = 0x80`

The transfer protocol is completely handled in the BlueVoiceOPUS service: at application level, the user only has to call `BluevoiceOPUS_SendAudioData` and `BluevoiceOPUS_ParseData` APIs.

Those functions internally calls:

- `BluevoiceOPUS_TP_Encapsulate`: splits the encoded buffer in groups of 19 bytes plus one byte for the tranfer protocol;
- `BluevoiceOPUS_TP_Parse`: extracts the payload and rebuilds the Opus encoded frame.

2.8 FP-AUD-BVLINK2 application description

FP-AUD-BVLINK2 central and peripheral applications are provided in the Projects directory. Ready-to-build projects are available for multiple IDEs and different development boards (Nucleo, SensorTile and BlueCoin).

To show full-duplex speech communication over BLE using the BlueVoiceOPUS Profile specification, central and peripheral application projects are included in the package. Depending on the start or stop event management, three types of communication could be set-up: simplex, half-duplex or full-duplex.

As involved in a bidirectional communication system, both modules can act as transmitters or receivers of voice communication:

- when a module is streaming, the application handles audio acquisition, data compression and packetization of the Opus compressed audio to be streamed over BLE, according to the BlueVoiceOPUS profile specification.
- when a module is receiving, the application is responsible for the parsing and decoding of Opus audio data received via BLE and for the USB or audio OUT streaming of decoded PCM samples.

After connection, the active communication channel is controlled by the user button on STM32Nucleo and BlueCoin, or by a double tap if a SensorTile is used.

Figure 7. FP-AUD-BVLINK2: central-peripheral communication diagram shows how the communication link is set and the audio streaming is performed. Once connected, pressing the user button starts and stops the audio streaming. If the node is receiving, a start streaming event creates full-duplex communication.

Depending on the status of the two modules, a LED shows:

- initialization: slow blinking
- connection: normal blinking
- streaming: fast blinking
- receiving: steady on (not blinking)
- full-duplex: both fast blinking

Figure 7. FP-AUD-BVLINK2: central-peripheral communication diagram
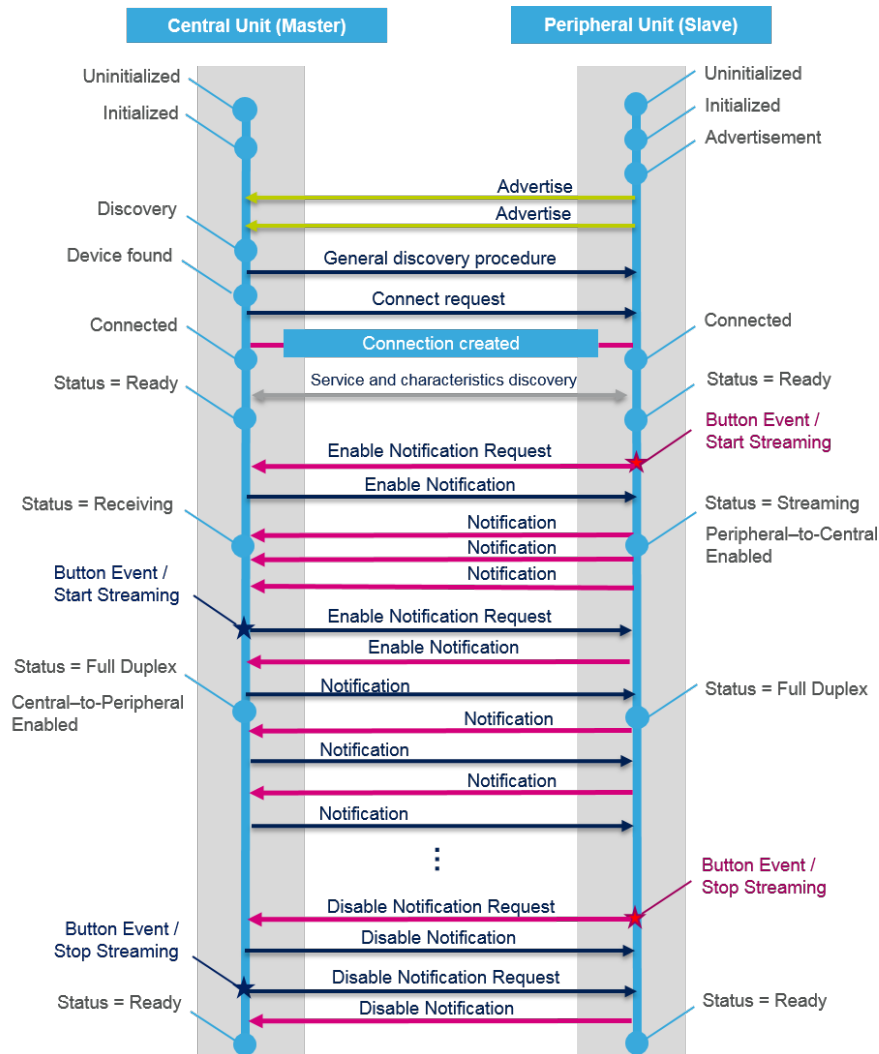
2.8.1 Initialization

During the first phase, all the hardware peripherals are initialized: BlueNRG, audio OUT or USB depending on the board used, digital MEMS microphone acquisition, button and LEDs. Afterwards, the BLE address and the generic access profile are configured.

BlueVoiceOPUS profile requires the configuration of an Opus encoder and decoder to act as a transmitter and a receiver. The BLE service and characteristics are then created using the relevant API (further information at Section 2.7.2.1 Initialization and configuration).

Finally, the audio acquisition is started and paused waiting for an user command to resume it.

In the infinite main loop the `HCI_Process` manages the BLE operation waiting for event and the `BV_APP_CEN_Process` state machine handles the application layer.

2.8.2 BLE link creation

After the initialization and configuration step, a BLE link must be created. Depending on central or peripheral role, BlueVoiceOPUS application is in advertise or discovery mode (`BV_APP_STATUS_DISCOVERY` or `BV_APP_STATUS_ADVERTISEMENT`). As soon as the peripheral module is discovered, the central module requests the connection.

A small connection interval has been chosen to minimize the overall audio latency:

- Max. connection interval: 10 ms
- Min. connection interval: 10 ms

The BLE link is now created, both the modules run a service and characteristic discovery to obtain the BlueVoiceOPUS profile handle exported by the other node (`BV_APP_STATUS_HANDLE_DISC`).

At the end of this procedure, the control characteristic is enabled on both nodes to allow start/stop streaming requests.

The application is now in `BV_APP_STATUS_READY` status waiting for a request that can create simplex, half-duplex or full-duplex communication.

2.8.3 Audio streaming

On the STM32 Nucleo stack or on the BlueCoin, the blue user button triggers a start/stop streaming. With a SensorTile, a double tap has the same effect. As soon as the button is pressed, a message is sent to request the audio notification enable by the receiver and the status is set to `BV_APP_STATUS_RECEIVING`.

The audio notifications are now enabled on the transmitter side, the status is set to `BV_APP_STATUS_STREAMING` and the audio acquisition is resumed, raising an interrupt each 20 ms of audio data acquired (the Opus frame size is set to 20 ms). The flag `audio_data_ready` is set to 1 and a software interrupt is generated. The audio data are encoded and sent by calling `BluevoiceOPUS_SendAudioData`.

On the receiver side the same procedure can be performed and both nodes switch to `BV_APP_STATUS_FULLDUPLEX`. It is possible to stop the audio streaming on one or both modules by pressing the user button, a disable notification request is sent and the audio acquisition is paused.

3 System setup guide

3.1 Hardware description

The FP-AUD-BVLINK2 application is based on:

- STM32 Nucleo: with ready-to-build projects available for NUCLEO-F446RE or NUCLEO-L476RG, together with a BlueNRG-MS expansion board (X-NUCLEO-IDB05A1) and a microphone expansion board (X-NUCLEO-CCA02M1);
- SensorTile: STEVAL-STLCS01V1 mounted on the cradle expansion STEVAL-STLCX01V1 which embeds an audio DAC and allows to connect a headset or a speaker;
- BlueCoin: STEVAL-BCNCS01V1 mounted on the STEVAL-BCNST01V1 Coin Station which embeds an audio DAC and allows a headset or a speaker connection.

The full-duplex demo can be set up by using a peripheral and a central node chosen among the different architectures described above.

Figure 8. **FP-AUD-BVLINK2 application system overview**
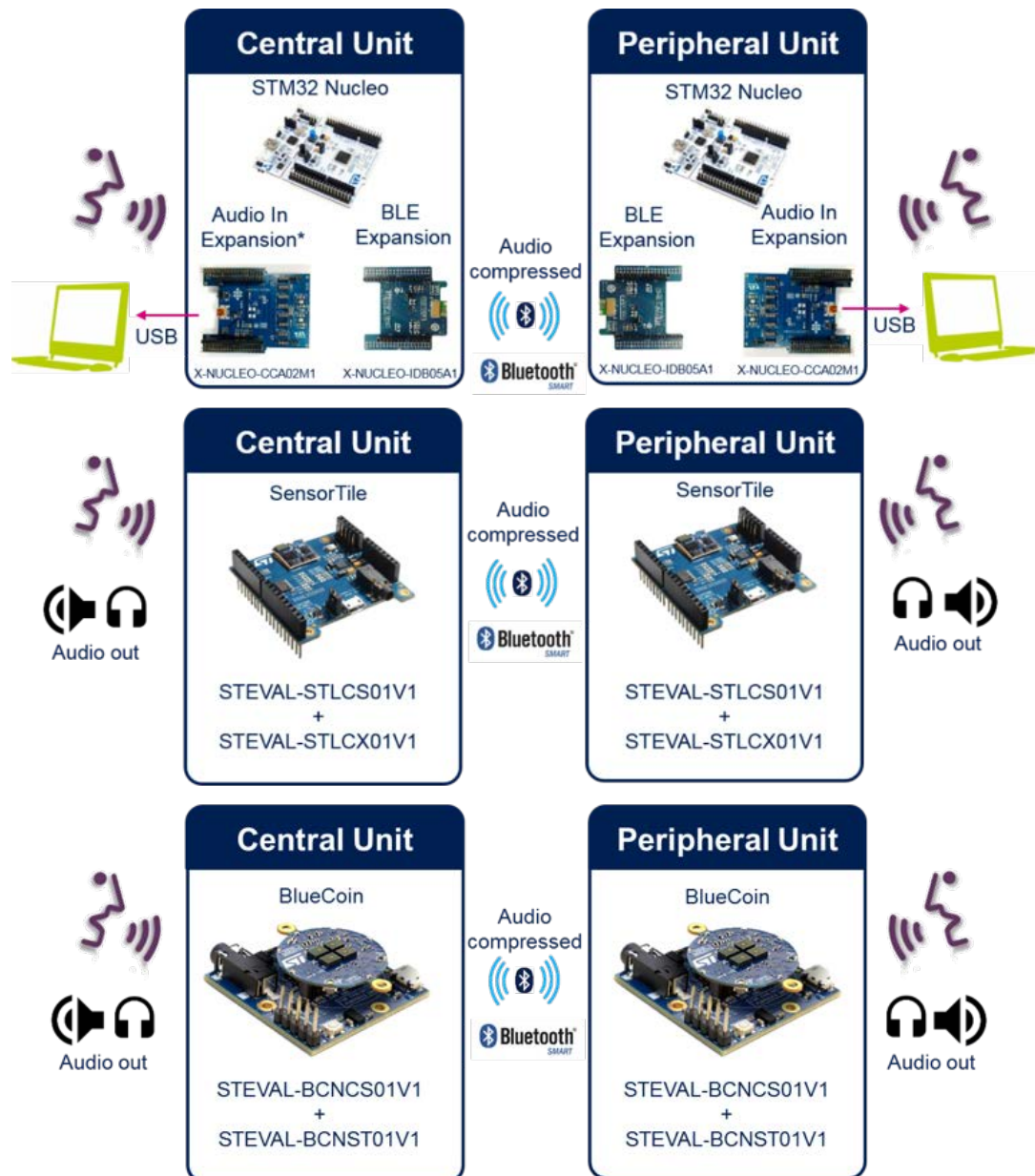
3.1.1 STM32 Nucleo platform

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

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

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/ programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples.

Figure 9. STM32 Nucleo board
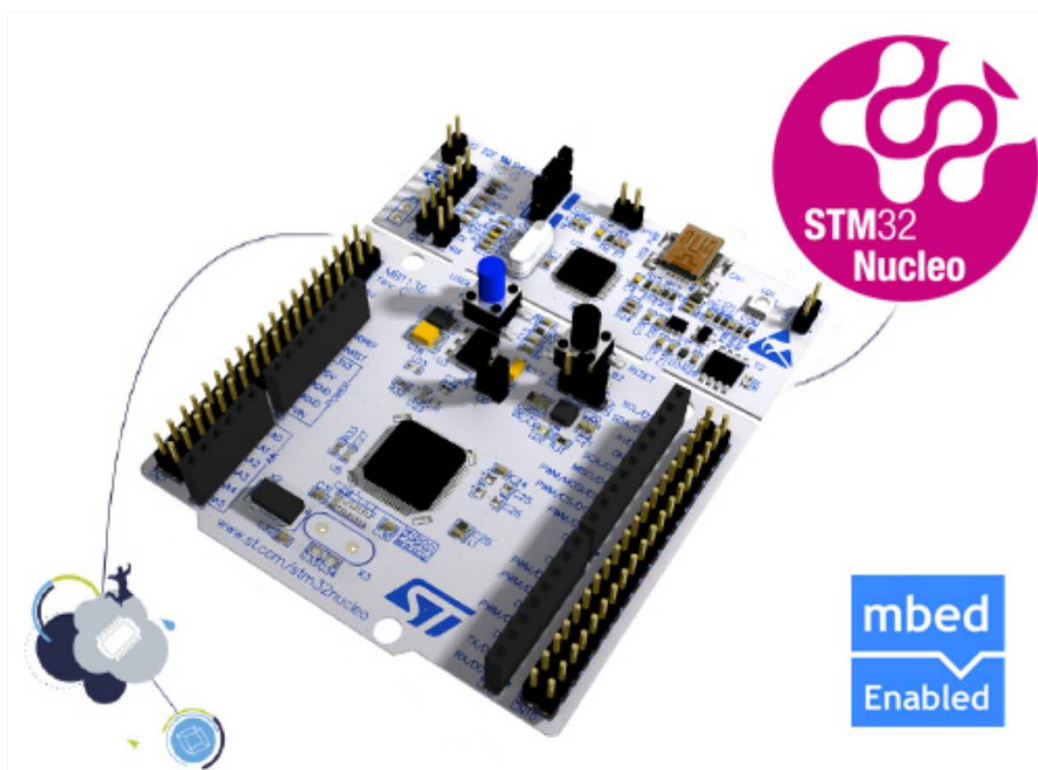
Information regarding the STM32 Nucleo board is available at www.st.com/stm32nucleo

3.1.2 X-NUCLEO-CCA02M1 expansion board

The X-NUCLEO-CCA02M1 is an expansion board based on digital MEMS microphones. It is compatible with the morpho connector layout, and is designed around STMicroelectronics MP34DT01-M digital microphones. There are two microphones soldered onto board and it offers the possibility to plug in additional microphones using MP32DT01 (or MP34DT01-M) based coupon evaluation board STEVAL-MKI129V3 (or STEVAL-MKI155V3).

The X-NUCLEO-CCA02M1 allows the acquisition of up to two microphones using the I²S bus and up to four coupon microphones using I²S and SPI together. In addition, it offers a USB output for the STM32 Nucleo board. It represents a fast and easy solution for the development of microphone-based applications as well as a starting point for audio algorithm implementation.

Figure 10. X-NUCLEO-CCA02M1 expansion board
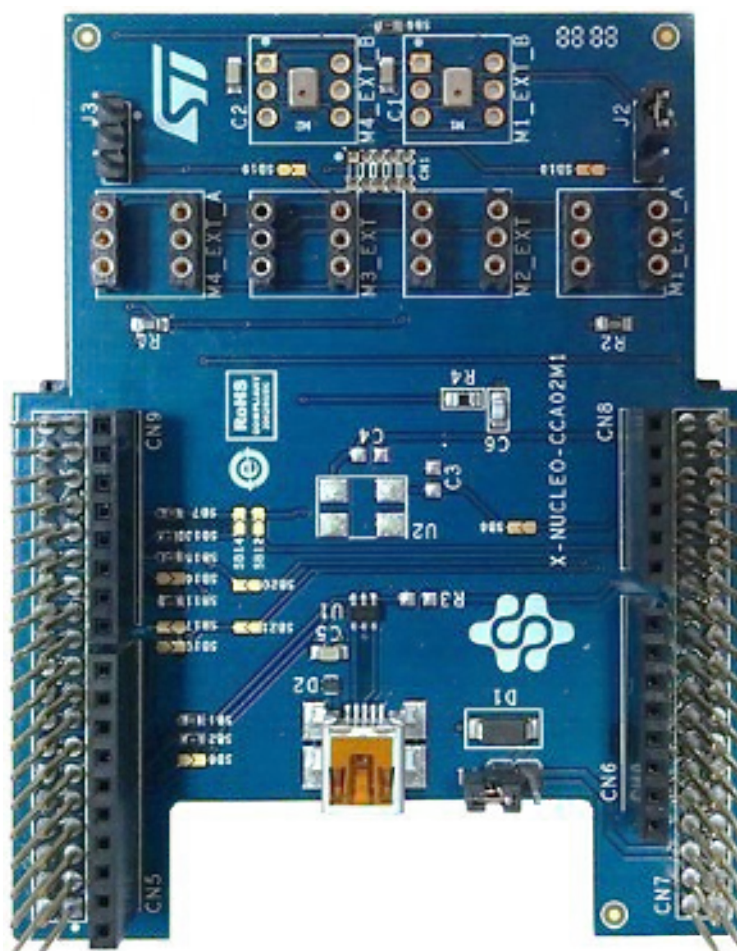
Information regarding the X-NUCLEO-CCA02M1 expansion board is available on www.st.com at http://www.st.com/x-nucleo.

3.1.3 X-NUCLEO-IDB05A1 expansion board

The X-NUCLEO-IDB05A1 is a Bluetooth low energy evaluation board based on the SPBTLE-RF BlueNRG-MS RF module to allow expansion of the STM32 Nucleo boards. The SPBTLE-RF module is FCC (FCC ID: S9NSPBTLERF) and IC certified (IC: 8976C-SPBTLERF). The BlueNRG-MS is a very low power Bluetooth low energy (BLE) single-mode network processor, compliant with Bluetooth specification v4.2. X-NUCLEO-IDB05A1 is compatible with the ST morpho and Arduino™ UNO R3 connector layout. This expansion board can be plugged into the Arduino UNO R3 connectors of any STM32 Nucleo board.

Figure 11. **X-NUCLEO-IDB05A1 expansion board**

Information about the X-NUCLEO-IDB05A1 expansion board is available on www.st.com at http://www.st.com/x-nucleo

3.1.4 STEVAL-STLKT01V1 SensorTile development kit

3.1.4.1 *Description*

The STEVAL-STLKT01V1 is a comprehensive development kit designed to support and expand the capabilities of the SensorTile and comes with a set of cradle boards enabling hardware scalability. The development kit simplifies prototyping, evaluation and development of innovative solutions. It is complemented with software, firmware libraries and tools, including a dedicated mobile App.

The SensorTile is a tiny, square-shaped IoT module that packs powerful processing capabilities leveraging an 80 MHz STM32L476JG microcontroller and Bluetooth low energy connectivity based on BlueNRG-MS network processor as well as a wide spectrum of motion and environmental MEMS sensors, including a digital microphone.

SensorTile can fit snugly in your IoT hub or sensor network node and become the core of your solution.

To upload new firmware onto the SensorTile, an external

SWD debugger (not included in the kit) is needed. It is recommended to use ST-LINK/V2-1 found on any STM32 Nucleo-64 development board.

3.1.4.2 *Features*

- Included in the development kit package:
  - SensorTile module (STEVAL-STLCS01V1) with STM32L476, LSM6DSM, LSM303AGR, LPS22HB, MP34DT04, BlueNRG-MS, BALF-NRG-01D3 and LD39115J18R
  - SensorTile expansion Cradle board equipped with audio DAC, USB port, STM32 Nucleo, Arduino UNO R3 and SWD connector
  - SensorTile Cradle with battery charger, humidity and temperature sensor, SD memory card slot, USB port and breakaway SWD connector
  - 100 mAh Li-Ion battery
  - Plastic box
  - SWD programming cable
- Software libraries and tools
  - STSW-STLKT01: SensorTile firmware package that supports sensors raw data streaming via USB, data logging on SDCard, audio acquisition and audio streaming.
  - FP-SNS-ALLMEMS1 and FP-SNS-MOTENV1: STM32 ODE functional packs

- – BlueMS: iOS and Android demo Apps
  - – BlueST-SDK: iOS and Android Software Development Kit
- CE certified
- RoHS and China RoHS compliant
- FCC (ID: S9NSTILE01) certified
- IC (IC: 8976C-STILE01) certified with PMN: STEVAL-STLKT01V1; HVIN: STEVAL-STLCS01V1; HMN: STEVAL-STLCX01V1; FVIN: bluenrg_7_1_e_Mode_2-32MHz-XO32K_4M.img
- TYPE certified (006-000482)

3.1.4.3 *Boards included in the kit*

Figure 12. STLCS01V1 board photo

STLCS01V1 SensorTile component board features

- Very compact module for motion, audio and environmental sensing and Bluetooth low energy connectivity with a complete set of firmware examples
- Supported by the STM32Cube and the STM32 ODE functional pack FP-SNS-ALLMEMS1 and FP-SNS-MOTENV1
- Mobile connectivity via the ST BlueMS app, available for iOS and Android
- Main components:
  - – STM32L476 – 32-bit ultra-low-power MCU with CortexM4F
  - – 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
  - – MP34DT04 – 64dB SNR Digital MEMS Microphone
  - – BlueNRG-MS – Bluetooth low energy network processor
  - – BALF-NRG-01D3 – 50 Ω balun with integrated harmonic filter
  - – LD39115J18R – 150 mA low quiescent current low noise LDO 1.8 V
  - – 2 V-5.5 V power supply range
  - – External interfaces: UART, SPI, SAI (Serial Audio Interface), I²C, DFSDM, USB OTG, ADC, GPIOs
- Pluggable or solderable interface
- SWD interface for debugging and programming capability

STLCS01V1 SensorTile component board description

STEVAL-STLCS01V1 (SensorTile) is a highly integrated reference design that can be plugged into form-factor prototypes to add sensing and connectivity capabilities to new designs through a smart hub solution. It can also easily support development of monitoring and tracking applications as standalone sensor node connected to iOS/ Android smartphone applications.

The SensorTile comes in a very small square shape 13.5 x 13.5 mm. All the electronic components are on the top side of the pcb, while the bottom side has a small connector through which it is possible to easily plug and unplug it from a motherboard. The connector pinout is also replicated on 18 pcb pads that render the SensorTile a solderable system on module as well.

The module comes with pre-loaded FP-SNS-ALLMEMS1 (former BLUEMICROSYSTEM2) software that initializes all the sensors and the Bluetooth low energy radio. The "ST BlueMS" app, available free of charge on Apple Store™ and Google Play™, is the easiest and fastest way to start using the SensorTile board and to experience a real activity monitoring system.

The SensorTile firmware package STSW-STLKT01, built on the STM32Cube software technology, includes all the low level drivers to manage the on-board devices and system-level interfaces. It has been designed in order to be easily extended and personalized as starting point for development and customization of new dedicated applications.

All the firmware packages are freely available on www.st.com.

The Bluetooth radio power output is set by default at 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 the change of this operating value will require an update of the FCC and IC certifications, with additional radio emission tests to be performed.

Figure 13. STLCR01V1 board photo
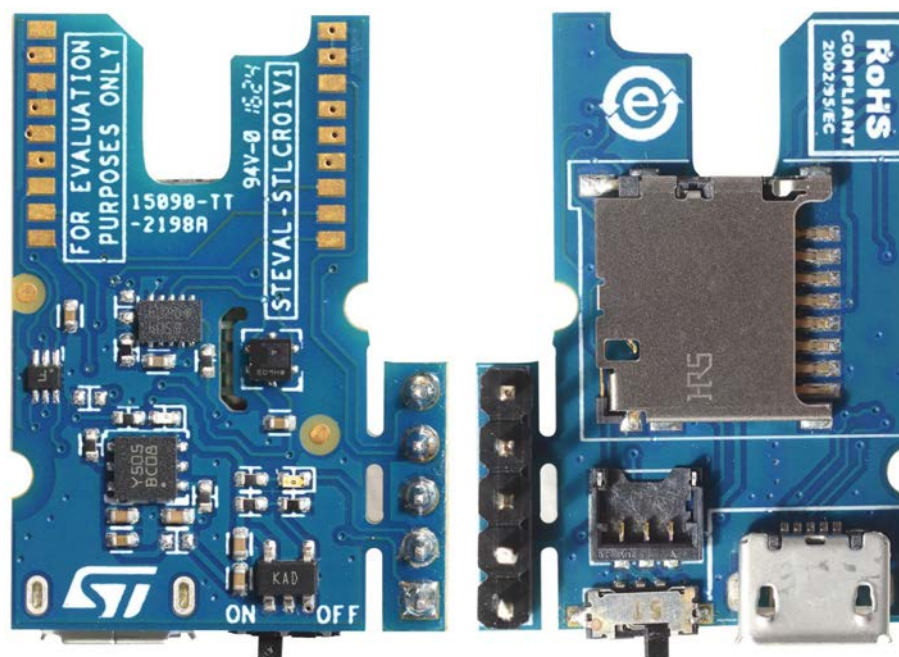

STLCR01V1 SensorTile component board features

- Sensortile Cradle board with SensorTile footprint (solderable)
- STBC08PMR – 800 mA standalone linear Li-Ion battery charger
- HTS221 – capacitive digital sensor for relative humidity and temperature
- LDK120M-R – 200 mA low quiescent current very low noise LDO
- STC3115 – Fuel gauge IC
- USBLC6-2P6 – very low capacitance ESD protection

- USB type A to Mini-B USB connector for power supply and communication
- microSD card socket
- SWD connector for programming and debugging

Figure 14. STLCX01V1 board photo
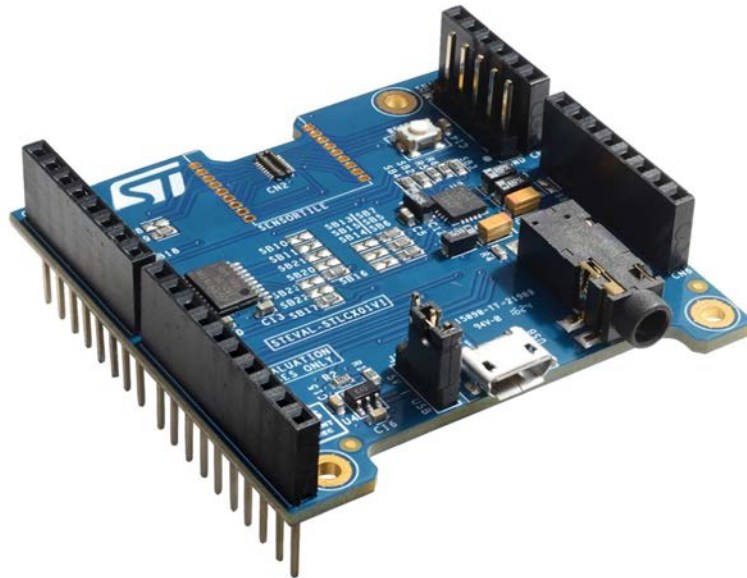
STLCX01V1 SensorTile component board features

- Sensortile Cradle expansion board with SensorTile plug connector
- Compatible with STM32 Nucleo boards through Arduino UNO R3 connector
- LDK120M-R – 200 mA low quiescent current very low noise LDO
- ST2378ETTR – 8-bit dual supply 1.71 V to 5.5 V level translator
- USBLC6-2P6 – very low capacitance ESD protection
- 16-Bit, low-power stereo audio DAC
- Micro-USB connector for power supply and communication
- Reset button
- SWD connector for programming and debugging

3.1.5 STEVAL-BCNKT01V1 BlueCoin development kit

3.1.5.1 *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.

3.1.5.2   *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 MP34DT04-C1, 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: STM32 ODE function pack 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

3.1.5.3   *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 MP34DT04-C1 – 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 15. STEVAL-BCNCS01V1 - BlueCoin Core System
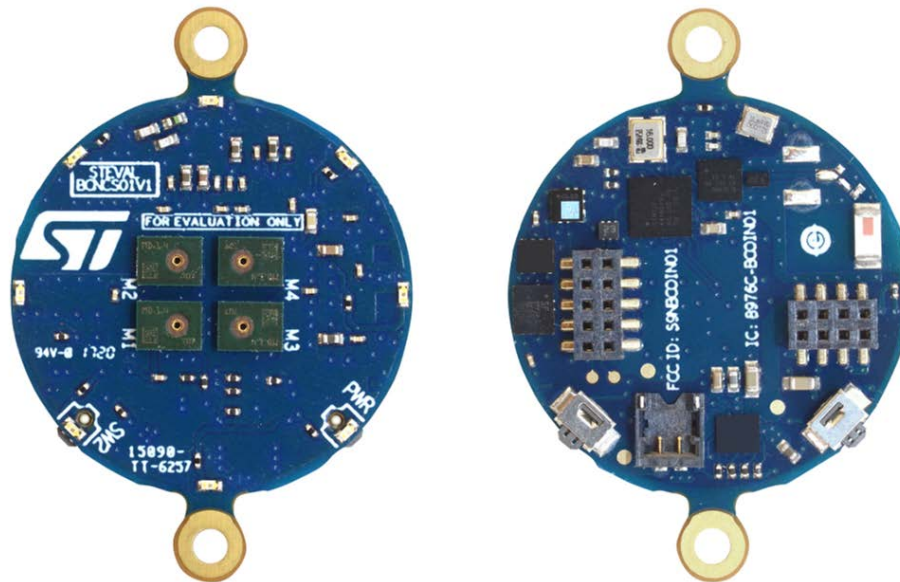
STEVAL-BCNCR01V1 - BlueCoin Cradle board features

- 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 16. STEVAL-BCNCR01V1 - BlueCoin Cradle board
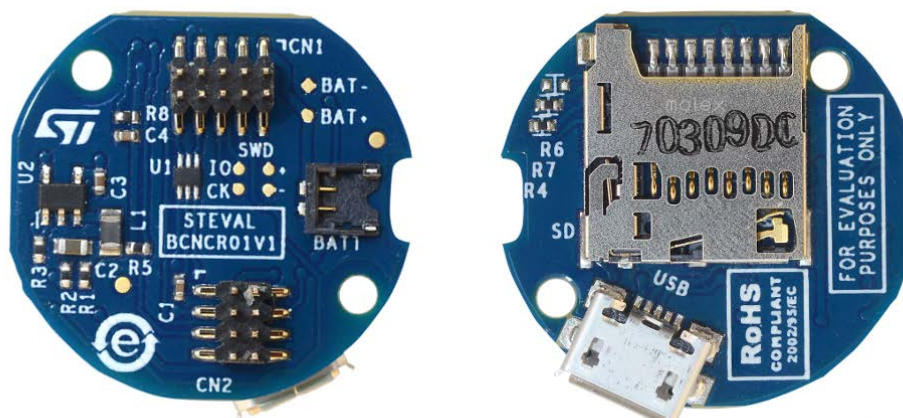
STEVAL-BCNST01V1 - CoinStation board features

- 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 17. STEVAL-BCNST01V1 - CoinStation board
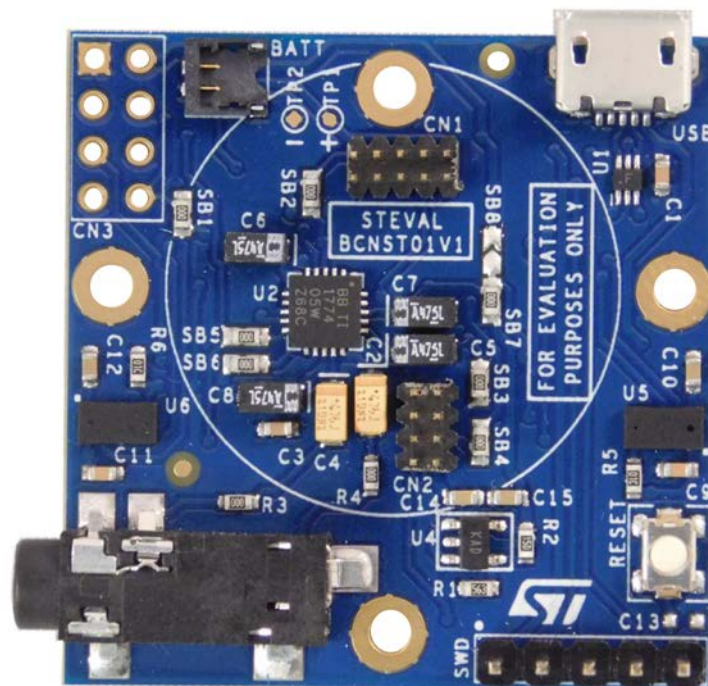
3.2 Software description

The following software components are required to set up a suitable development environment:

- FP-AUD-BVLINK2: an application based on STM32Cube that demonstrates voice communication over BLE. The FP-AUD-BVLINK2 firmware related documentation is available on www.st.com.
- One of the following development environments:
  – IAR Embedded Workbench for ARM® (EWARM) toolchain plus ST-LINK
  – RealView Microcontroller Development Kit (MDK-ARM) toolchain plus ST-LINK
  – Ac6 System Workbench for STM32 toolchain plus ST-LINK

3.3 Hardware and software setup

3.3.1 Hardware setup

The following hardware components are needed:

1. For full-duplex communication:
   – two development platforms among STM32 Nucleo (NUCLEO-F446RE or NUCLEO-L476RG), SensorTile or BlueCoin

2. If you choose two STM32 Nucleo boards, two digital MEMS microphone expansion boards (order code: X-NUCLEO-CCA02M1) and two BlueNRG-MS Bluetooth low energy expansion boards (order code: X-NUCLEO-IDB05A1) are needed.

3. One USB type A to Mini-B USB cable to power the Tx module up, for STM32 Nucleo; one USB type A to Micro-B USB for SensorTile or BlueCoin.

4. For the STM32 Nucleo, one USB type A to Mini-B USB cable to power the Rx module up and for USB streaming; one USB type A to Micro-B USB for SensorTile or BlueCoin, plus a headset or a speaker.

Both the STM32 Nucleo development board and the X-NUCLEO-CCA02M1 expansion board must be correctly configured to run the FP-AUD-BVLINK2 application.

3.3.1.1 STM32 Nucleo development board configuration

The Tx STM32 Nucleo board uses the external high-speed clock provided by the on-board ST-LINK MCU. The frequency is fixed at 8 MHz and connected to the STM32 microcontroller PF0/PD0/PH0-OSC_IN.

The STM32 Nucleo board has to be configured as shown below.

Figure 18. STM32 Nucleo board configuration
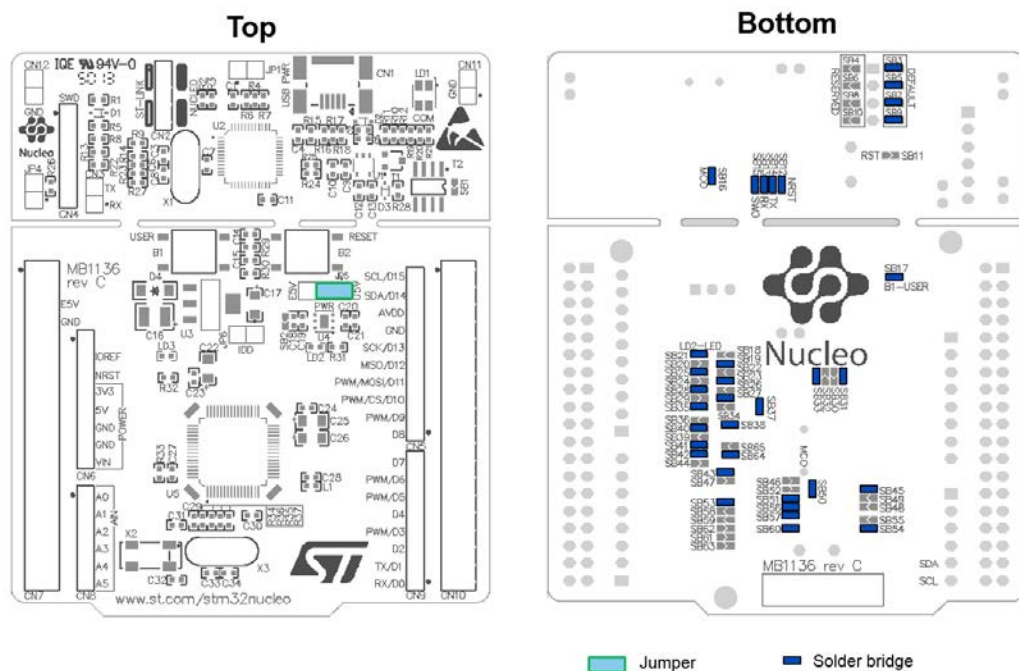
3.3.1.2 X-NUCLEO-CCA02M1 configuration

The X-NUCLEO-CCA02M1 board uses different configurations depending on the number of microphones acquired and on the peripheral used to acquired them (for all available configurations, refer to the X-NUCLEO-CCA02M1 user manual, UM1900, on www.st.com).

If NUCLEO-F446RE boards are used for full-duplex communication, only one microphone is acquired through the I²S.

Figure 19. X-NUCLEO-CCA02M1 hardware configuration for NUCLEO-F446RE
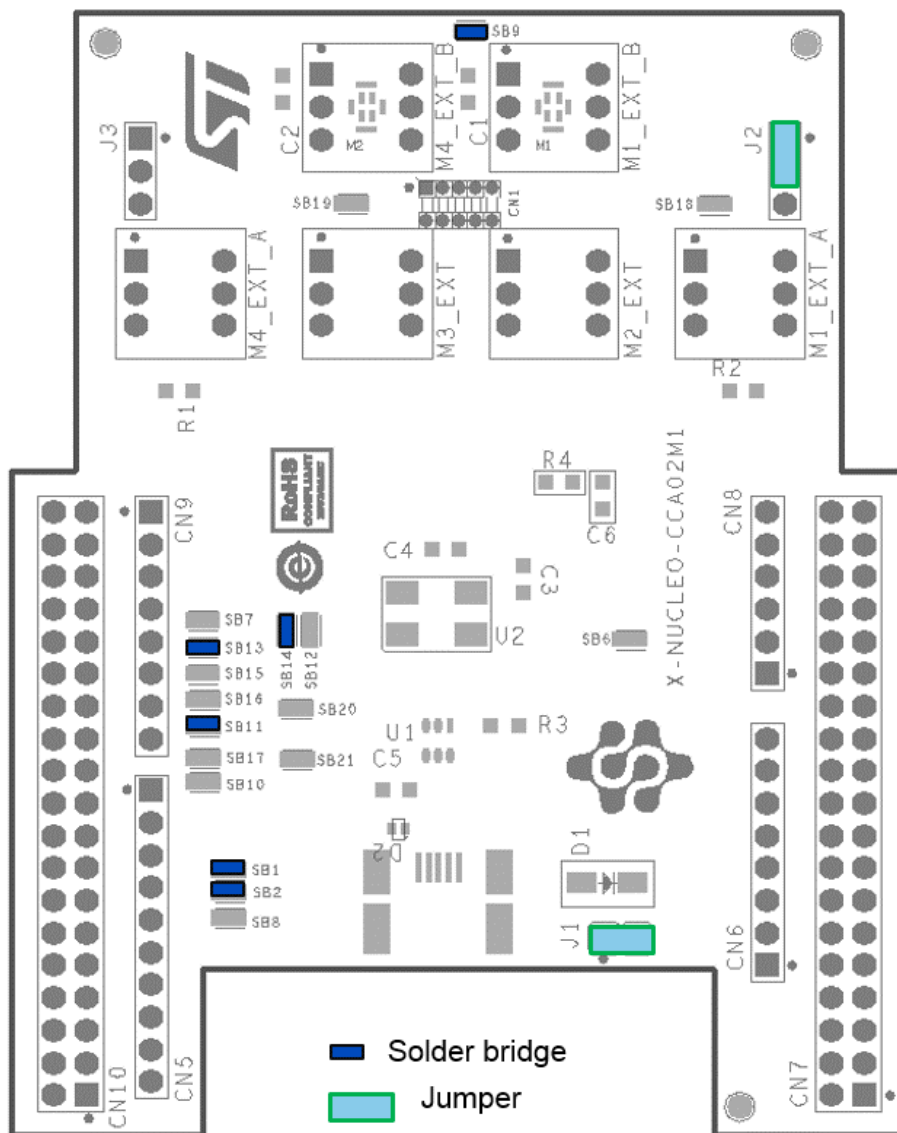
If two NUCLEO-L476RG boards are used for full-duplex communication, a different configuration is required; the clock is generated by the DFSDM peripheral and the PDM line of the microphone is routed to the MCU.

Figure 20. X-NUCLEO-CCA02M1 hardware configuration for NUCLEO-L476RG
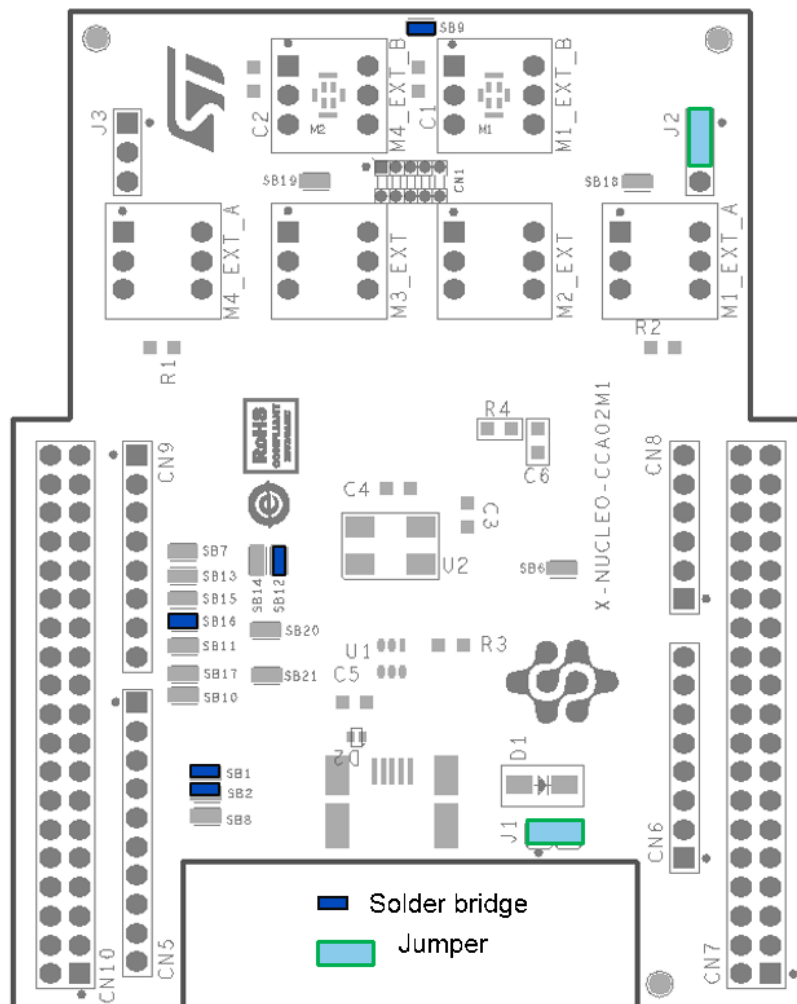
The device has to be recognized as a standard microphone to stream audio via USB; therefore, it needs to be interfaced directly to the X-NUCLEO-CCA02M1 USB mini connector.

Refer to the documentation available at http://www.st.com/x-nucleo for further information about the X-NUCLEO-CCA02M1 expansion board configuration.

3.3.2 Full-duplex software setup using STM32 Nucleo board

This section lists the minimum requirements to set the SDK up, run the sample testing scenario based on the previous description and customize applications running on STM32. The following section describes the demo setup in a Windows 7 environment.

3.3.2.1 *Development toolchains and compilers*

Select one of the integrated development environments supported by the STM32Cube expansion software. Read the system requirements and setup information provided by the selected IDE provider.

3.3.2.2 *Recognition of the device as a standard USB microphone in Windows 7*

When an STM32 Nucleo board stack is used as Tx or Rx module, both central and peripheral applications include an audio input USB driver that allows the device to be recognized as a standard USB microphone.

Once the firmware has been downloaded to the MCU Flash, move JP5 jumper to E5V and connect the X-NUCLEO-CCA02M1 to a PC via a mini USB cable. The board is recognized correctly in the Device Manager, as shown below.

Figure 21. Device Manager: STM32 Nucleo board microphone recognition
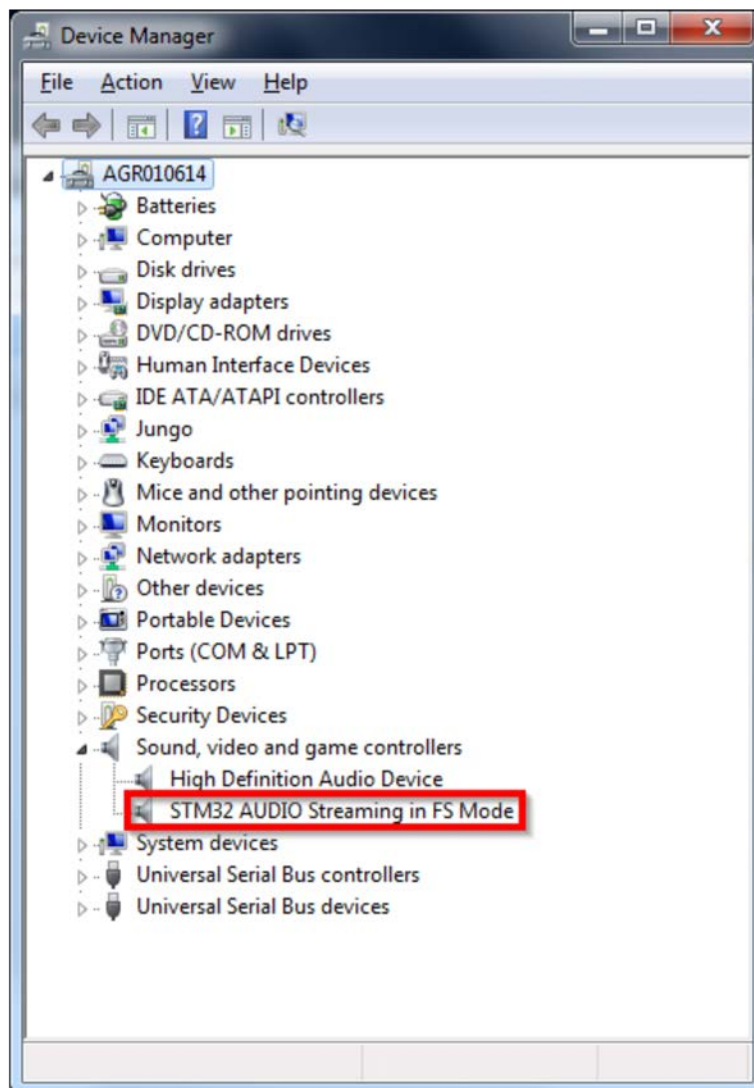
Right-click on the volume icon in the Windows task bar (on the bottom right side of the screen) and choose "recording device". Select STM32 microphone and click on Properties.

In the Advanced tab, a summary of the current device setup appears, showing the sampling frequency and number of channels. The module should be recognized as a "1 channel, 16000 Hz" microphone (if the audio sampling frequency set in the firmware is 16 kHz).

Note: *This procedure must be performed for both the central and peripheral modules.*

Figure 22. Advanced tab: microphone properties
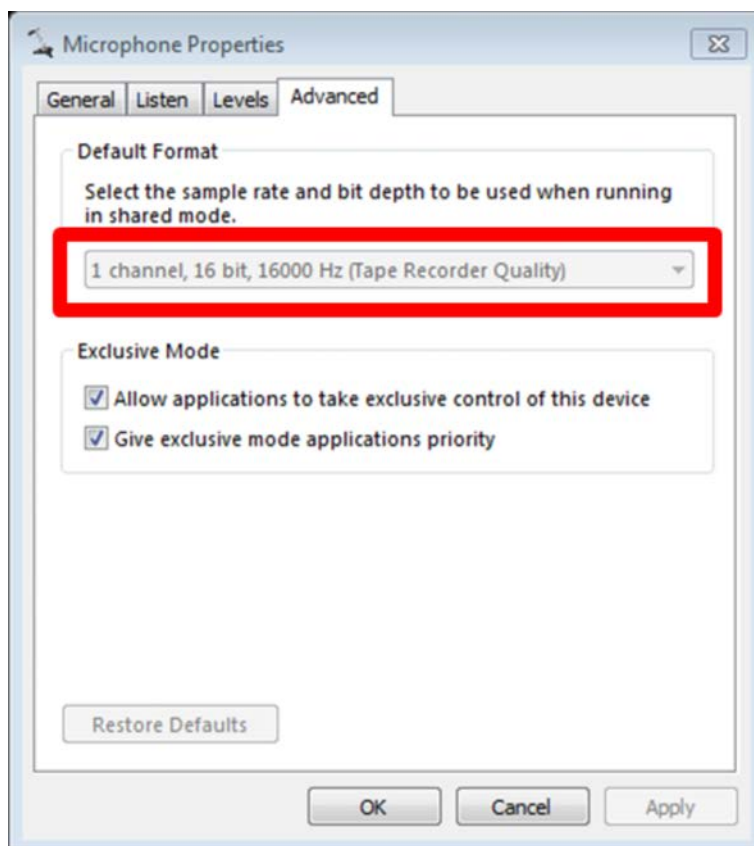
3.3.3 FP-AUD-BVLINK2 setup

This section describes how to set the FP-AUD-BVLINK2 full-duplex application up to demonstrate audio transmission over Bluetooth low energy.

Two BLE devices interact with each other creating point-to-point wireless communication.

One of the modules acts as central and the other as peripheral. Both nodes can stream audio at the same time.

3.3.3.1 STM32 Nucleo and expansion board setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer.

You can find the relevant version of the ST-LINK/V2-1 USB driver, STSW-LINK009, on www.st.com, or download the STM32 ST-LINK Utility (STSW-LINK004), a full-featured software interface for programming STM32 microcontrollers.

The X-NUCLEO-CCA02M1 microphone expansion board can be easily connected to the STM32 Nucleo development board through the ST morpho extension connector. The board can interface with the external STM32 Nucleo microcontroller via I²S or DFSDM and USB.

The X-NUCLEO-IDB05A1 BlueNRG-MS expansion board can be easily connected to the X-NUCLEO-CCA02M1 via the Arduino UNO R3 extension connector.

3.3.3.1.1 PC audio recording utility sample: Audacity

Audacity® is an open source, cross-platform program for recording and audio editing environment.

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

Figure 23. Audacity for Windows
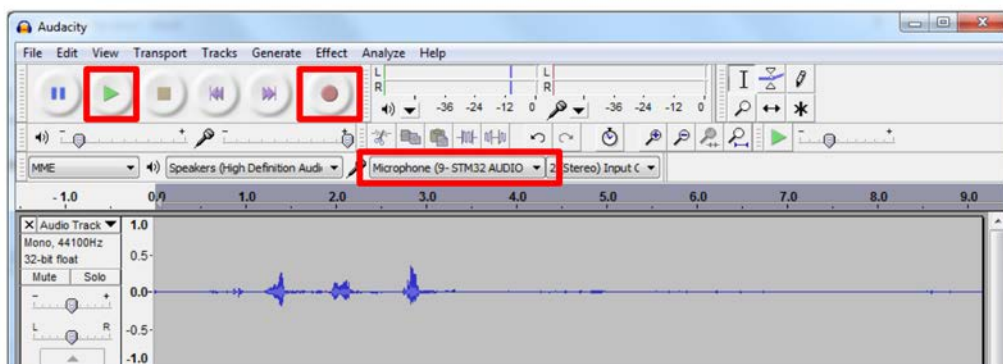
This program is used in the module setup to manage STM32Nucleo central and peripheral device microphones (see Section 3.3.3.1.2 Module setup, Section 3.3.3.1.3 Peripheral to central recording and Section 3.3.3.1.4 Central to peripheral recording).

3.3.3.1.2 Module setup

Step 1. Connect the X-NUCLEO-CCA02M1 USB connector of the central unit to the PC.

Figure 24. X-NUCLEO-CCA02M1 USB connection to the PC: module setup
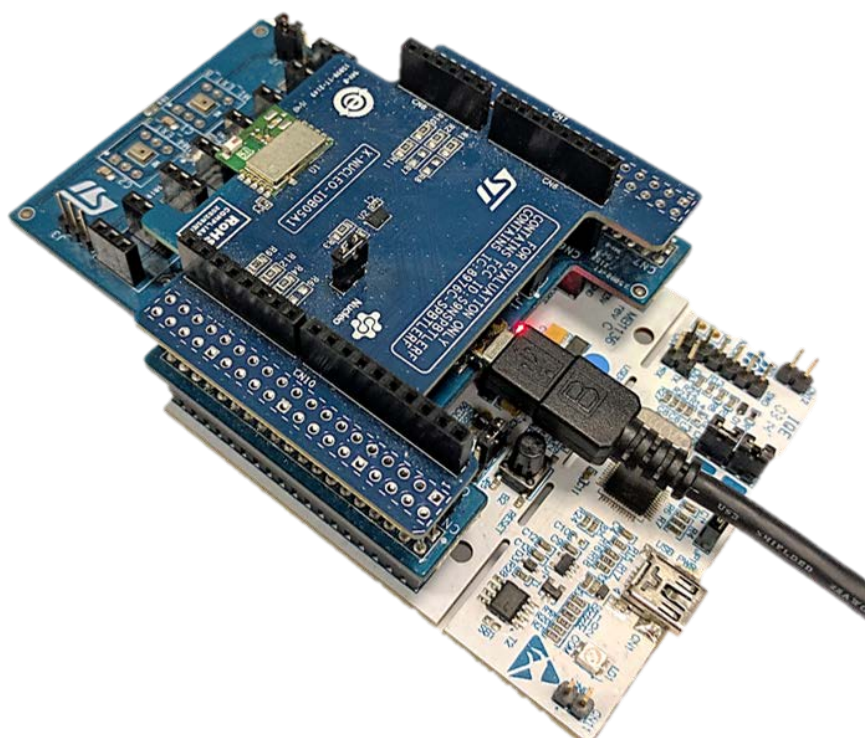
The STM32 Audio Streaming central appears in the Device Manager.

Figure 25. STM32 AUDIO Streaming peripheral
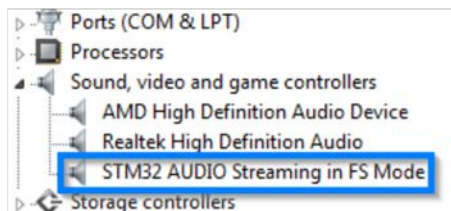
Step 2. Open Audacity

An STM32 microphone appears in the Input selector.

In the following image, the STM32 microphone 2 is considered the **Central unit** microphone. This microphone number may change different connections and PCs.

Figure 26. Central unit microphone in Audacity
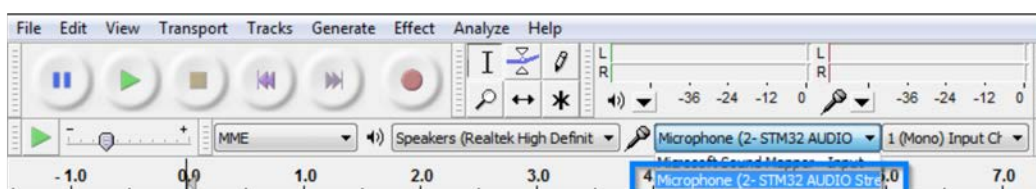
Step 3. Connect the X-NUCLEO-CCA02M1 USB connector of the peripheral unit to a PC.

A second STM32 AUDIO Streaming peripheral appears in Device Manager.

Figure 27. STM32 recognized as Audio Streaming peripheral
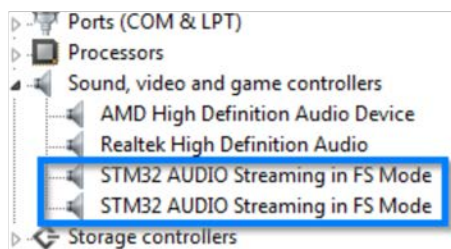
Step 4. In Audacity, click on Transport>Rescan Audio Devices

Two STM32 microphones appear in the Input selector.

In the following image, STM32 microphone 4 is considered the **Peripheral unit** microphone.

Figure 28. Peripheral unit microphone in Audacity
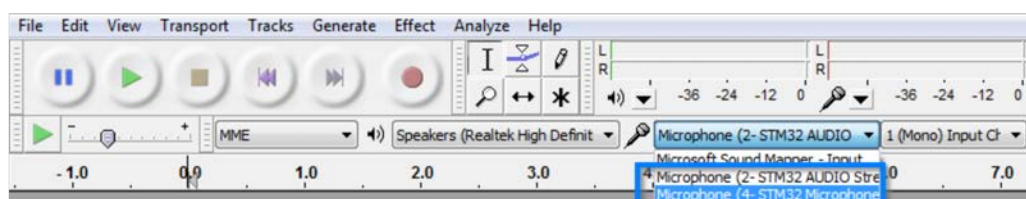
3.3.3.1.3 **Peripheral to central recording**

Step 1. Choose the Central unit microphone (microphone 2)

Step 2. Click Record to start silent recording

Figure 29. Audacity silent recording from Central unit USB stream
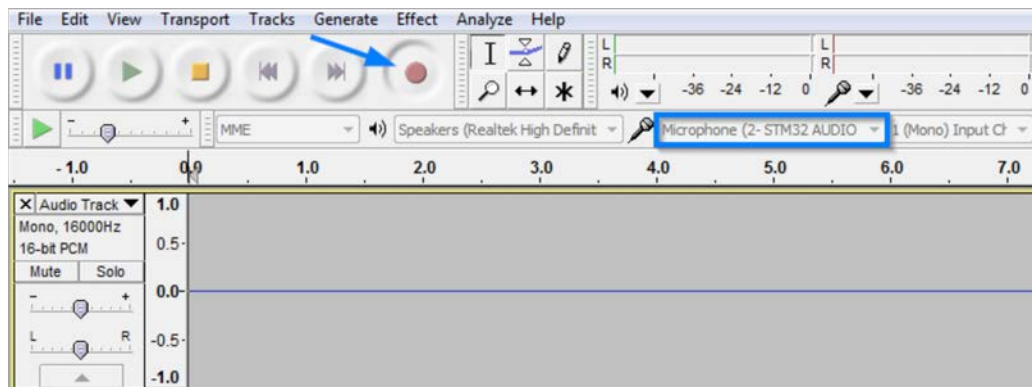
Step 3. Click Peripheral unit user button

Peripheral unit streams voice to the **Central unit**

Figure 30. Audacity recording voice coming from Peripheral unit
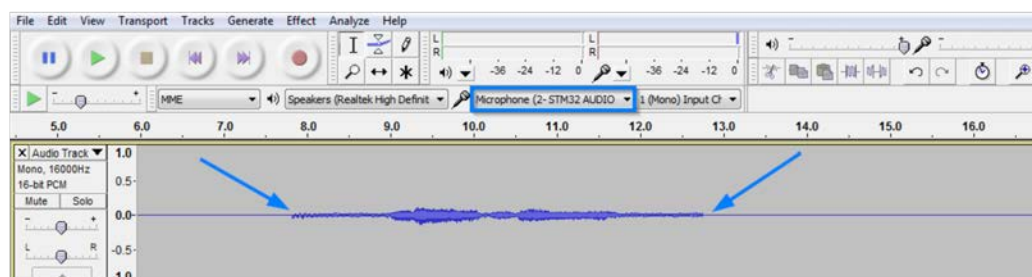
Step 4. Click the button to toggle the streaming status.

3.3.3.1.4 **Central to peripheral recording**

Step 1. Choose the Peripheral unit microphone (microphone 4).

Step 2. Click Record to start silent recording.

Figure 31. Audacity silent recording from Peripheral unit USB stream
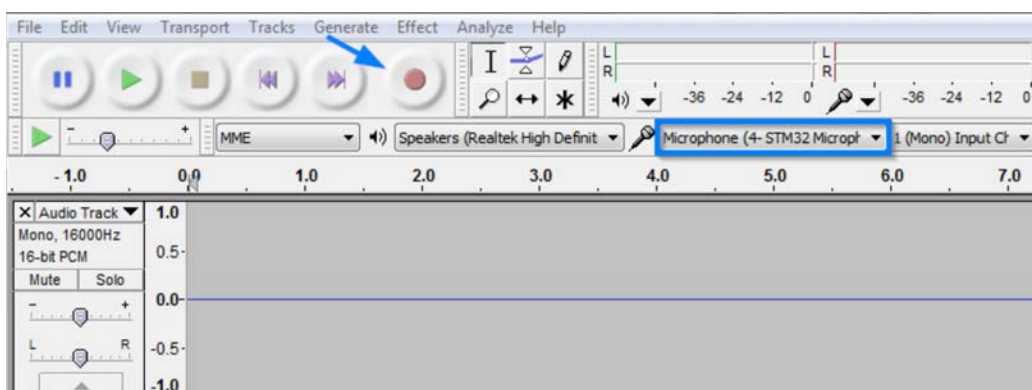
Step 3. Click central unit user button

Central unit streams voice to the **Peripheral unit**

Figure 32. Audacity recording voice coming from Central unit
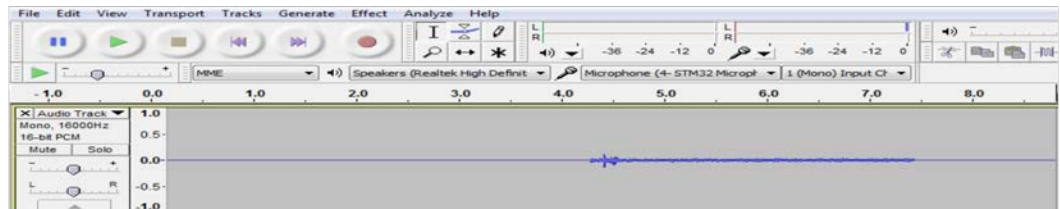
Step 4. Click the button to toggle the streaming status.

3.3.3.2 *SensorTile system setup*

The FP-AUD-BVLINK2 supports the STEVAL-STLKT01V1.

To create full-duplex communication, two SensorTile kits are necessary (each module consists of a STEVAL-STLCS01V1 and a STEVAL-STLCX01V1 development board).

3.3.3.2.1 **SensorTile module setup**

Two modules are needed to create a BLE link: one peripheral and one central.

The SensorTile core system (STEVAL-STLCS01V1) must be plugged to the cradle expansion (STEVAL-STLCX01V1) through the dedicated connector.

Note: *Be careful in matching the orientation (ST logo must have the same orientation on both boards).*

To program the board, connect an ST-LINK (included in STM32Nucleo boards) to the cradle SWD connector (a 5-pin flat cable is provided in the SensorTile Kit package), paying attention to the polarity of the connectors and ensuring that CN2 jumpers are OFF.

Pin 1 is identified by a small circle on the STM32Nucleo board and SensorTile cradle expansion PCB silkscreens.

Figure 33. SensorTile-STM32Nucleo: SWD connections with a 5-pin flat cable
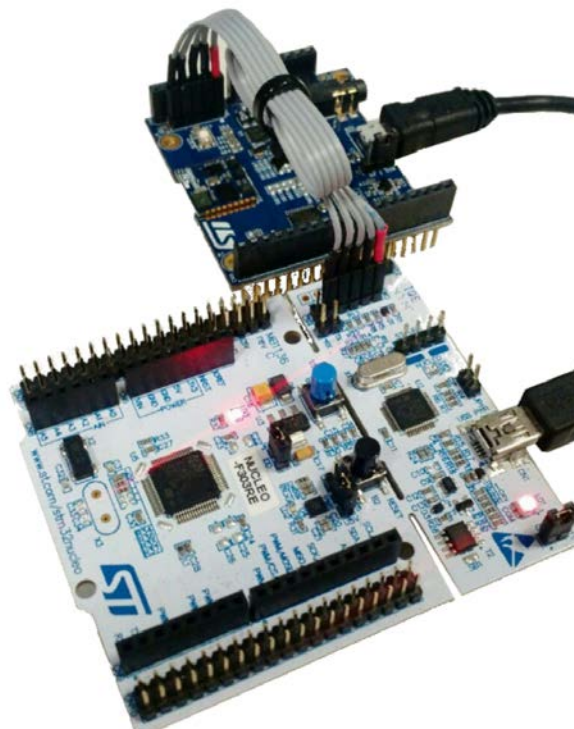
It is possible to reproduce the streamed audio with a headset or a speaker connected to the 3.5 mm phone jack.

The audio streaming from one module to the other is activated with a double tap on top of the SensorTile core system. The red LED starts blinking faster during the streaming and can be stopped by another double tap.

Note: *Only one module can stream at a time.*

Figure 34. SensorTile half-duplex system setup
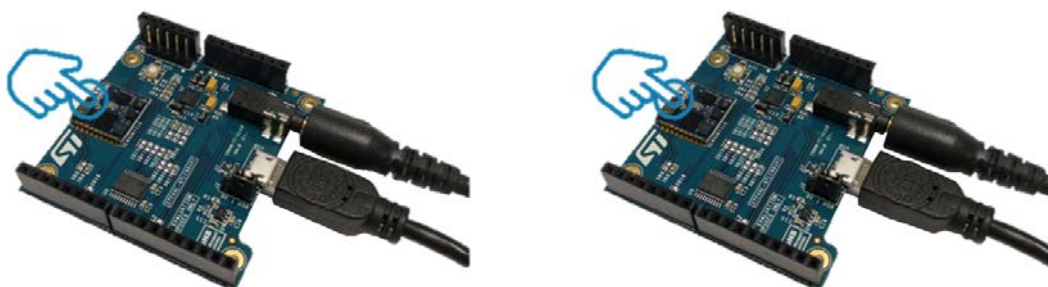
3.3.3.3 *BlueCoin system setup*

The FP-AUD-BVLINK2 also supports the STEVAL-BCNKT01V1.

To create a full-duplex communication, two BlueCoin kits are necessary (each kit consists of a STEVAL-BCNCS01V1 and a STEVAL-BCNST01V1 development board).

3.3.3.3.1 BlueCoin module setup

Two modules are needed in order to create a BLE link: one peripheral and one central.

The BlueCoin core system (STEVAL-BCNCS01V1) must be plugged to the cradle expansion (STEVAL-STLCX01V1) through the dedicated connector.

To program the board, connect an ST-LINK (included in STM32Nucleo boards) to the Coin Station SWD connector (a 5-pin flat cable is provided in the BlueCoin Kit package), paying attention to the polarity of the connectors and ensuring that CN2 jumpers are OFF.

Pin 1 is identified by a small circle on the STM32Nucleo board and Coin Station PCB silkscreens.

Figure 35. BlueCoin-STM32Nucleo: SWD connections with 5-pin flat cable

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

The audio streaming from one module to the other is activated by pressing the SW2 button. The red LED starts blinking faster during the streaming and can be stopped by pressing the button again.

Note: *Only one module can stream at a time.*

Figure 36. BlueCoin half-duplex system setup
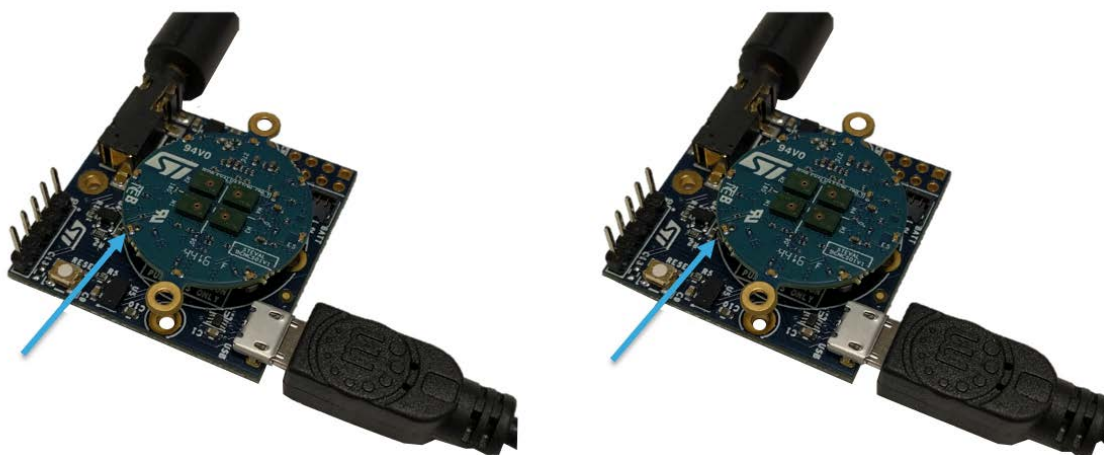
A References

1. Bluetooth Core Specification 4.1 on https://www.bluetooth.org
2. PDM audio software decoding on STM32 microcontrollers, Application note AN3998 on www.st.com
3. Opus Interactive Audio Codec
4. Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking: Townsend, K., Cufí, C., Akiba, Davidson, R., O'Reilly Media, 2014.

Revision history

Table 4. Document revision history

Date	Version	Changes
20-Mar-2018	1	Initial release.

Contents

List of tables

List of figures