Getting started with the STM32Cube function pack for STM32WB MCU featuring advanced audio streaming over Bluetooth 5.0 using Opus codec

Introduction

FP-AUD-BVLINKWB1 is an STM32Cube function pack that performs full-duplex voice streaming or stereo music streaming over BLE using the advanced Opus compression algorithm.

The application runs on P-NUCLEO-WB55 connected to an X-NUCLEO-CCA02M2 or on STM32WB5MM-DK and includes drivers and middleware for BLE and digital MEMS microphones. It also includes the complete Opus audio codec (v 1.3) as third-party middleware to perform bidirectional and simultaneous audio streaming between two STM32WB.

The peripheral module can also communicate in full-duplex mode (bidirectional audio at 16 kHz) with a mobile device running the STBLESensor app, or receive stereo music at 48 kHz from the same app.

The software with the suggested combination of STM32WB and ST devices can be used, for example, to develop wireless audio communication systems for smart home or wearable applications.

The Opus algorithm provides the flexibility to achieve high audio quality even at low bitrates, and the STM32WB has the low power capabilities to allow the development of applications featuring very low consumption.

RELATED LINKS

Visit the STM32Cube ecosystem web page on www.st.com for further information

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-BVLINKWB1 software description

2.1 Overview

The key features of the FP-AUD-BVLINKWB1 package are:

- Complete firmware to implement full-duplex communication or stereo music streaming over Bluetooth 5.0 using Opus codec for both encoding and decoding
- 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 v1.3 (downloadable from https://www.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
- Sample implementation available for X-NUCLEO-CCA02M2 connected to a P-NUCLEO-WB55 and for STM32WB5MM-DK
- Compatibility with STBLESensor app, to perform full-duplex audio streaming at 16 kHz and speech-to-text or to receive stereo music at 48 kHz from devices supporting BLE 4.2 or higher
- 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 the MEMS microphone expansion board, P-NUCLEO-WB55 and STM32WB5MM-DK, as well as middleware components for audio acquisition, communication with other BLE devices, USB streaming of recorded signals and a dedicated profile for audio streaming over BLE (*bvopus_service_stm*).

The third party Opus (v1.3) 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 messages. 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 service is responsible for audio encoding and periodic data transmission on the server side and for received data decoding 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, which 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. Both the central and the peripheral modules can act as a transmitter and a receiver at the same time, enabling full-duplex voice streaming.

The peripheral module can also perform full-duplex communication with a mobile device running the STBLESensor app or receive and decode stereo music at 48 kHz streamed by the same app.

2.2 Architecture

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

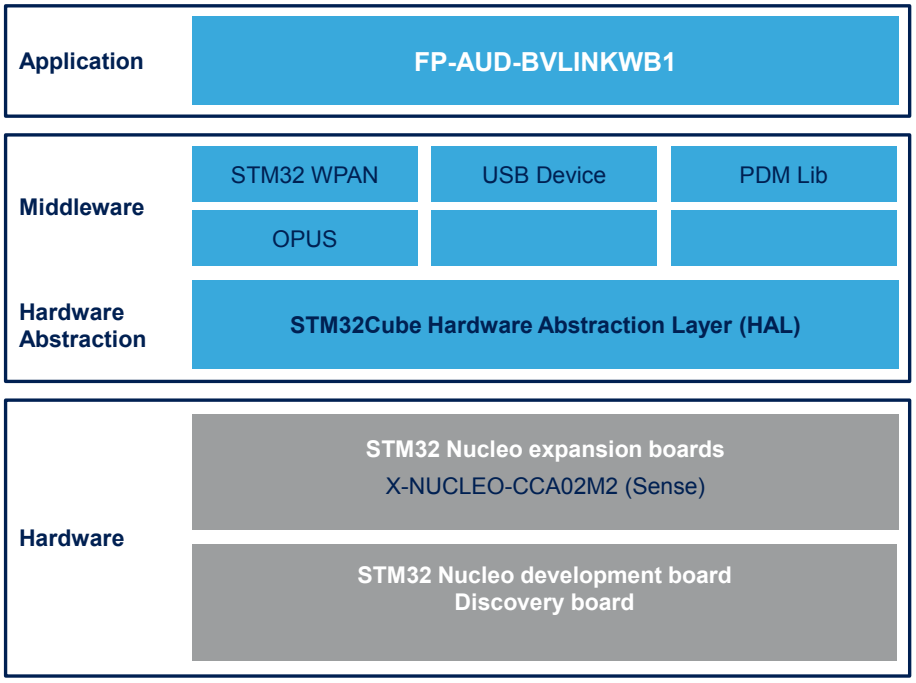
The FP-AUD-BVLINKWB1 provides a board support package (BSP) for the digital MEMS microphone expansion board (X-NUCLEO-CCA02M2), as well as for the STM32WB Nucleo board and the STM32WB Discovery kit.

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:

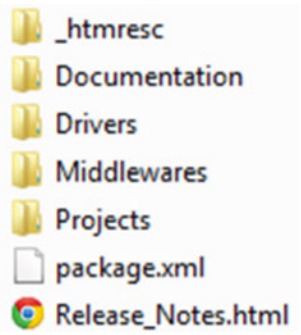
- **STM32Cube HAL layer**: provides a simple, generic and multi-instance set of generic and extension APIs (application programming interfaces) to interact with the upper application, library and stack layers. These generic and extension APIs are built on a common architecture and allow layers built 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 SPI, ADC, LEDs and user buttons. For the microphone acquisition board (X-NUCLEO-CCA02M2), it provides APIs for audio acquisition and processing.

Figure 1. FP-AUD-BVLINKWB1 software architecture



2.3 Folders

Figure 2. FP-AUD-BVLINKWB1 package 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, Bluetooth 5 profiles and services and the Opus codec source code that can be downloaded from Opus official website (https://www.opus-codec.org/)
- **Projects** contains three different projects: BVLCentral and BVLPeripheral allow performing a full-duplex audio streaming over BLE; BVLPeripheral_FullBand acts as stereo music receiver when connected to the Android ST BLE Sensor app. The projects are supplied for the P-NUCLEO-WB55 development board and STM32WB5MM-DK with the IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM-STR) and STM32CubeIDE

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-BVLINKWB1 function pack processing components have been designed to create a wireless audio link between a trasmitter (Tx) and a receiver (Rx) module. The whole speech processing chain begins with MEMS digital microphone acquisition or stereo music track and culminates in playthrough via USB, creating a full-duplex speech or stereo music streaming between two STM32WB devices or an STM32WB and a smartphone.

BLE has been configured to send packets with a maximum size of 150 bytes, exploiting if possible the BT5 LE 2M PHY feature. Depending on the application, encoded bytes can be above this threshold, so the compressed buffer must be split into multiple BLE packets. Moreover, the encoded buffer size can change every audio frame and the receiver has to know its length to rebuild it; a simple transfer protocol has been implemented for this scope (for further information about the transfer protocol, refer to Section 2.7.2.4 BlueVoiceOPUS transfer protocol).

The full-duplex application requires two different modules composed by a X-NUCLEO-CCA02M2 plugged on a P-NUCLEO-WB55 or by an STM32WB5MM-DK: a peripheral and a central node, both acting as transmitter and receiver at the same time.

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, it is transferred to the compression algorithm: the encoded buffer size returned by the Opus encoder can significantly change according to the Opus encoder parameters.

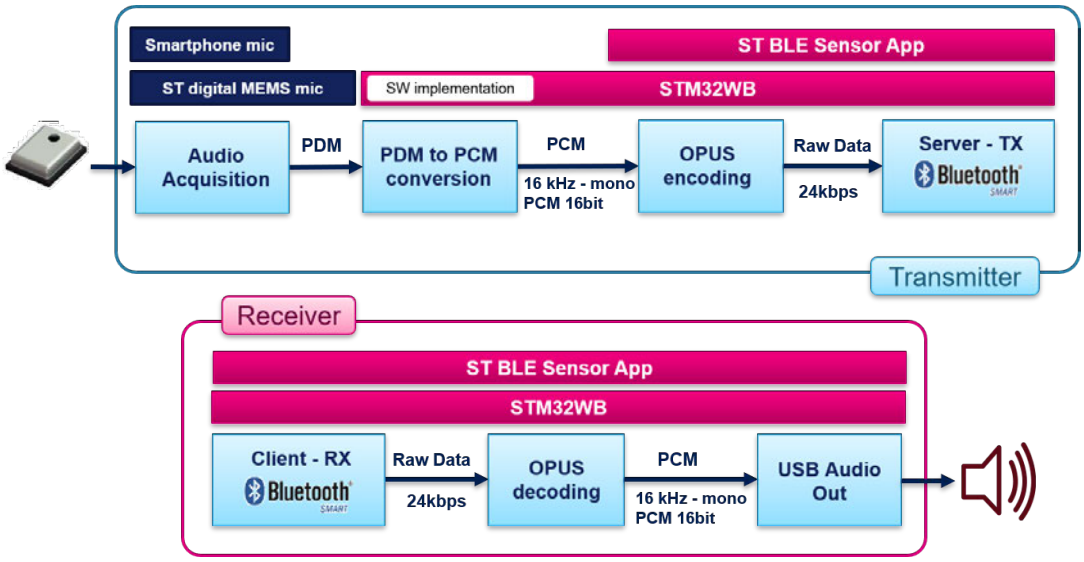
As soon as the BLE packets are received by the Rx module, they are parsed to rebuild the encoded buffer. After that, the Opus decoder returns the decompressed PCM audio data that are then sent via USB, configured as audio out interface. Any freeware or commercial audio recording software can be used to record and listen to the received audio (for example, Audacity).

The function pack implements full-duplex communication as both the central and the peripheral nodes can act as a transmitter or a receiver.

The full-duplex application included in the functional pack is configured to acquire audio with a sampling frequency of 16 kHz, to encode data each 20 ms (audio frame size) and stream them at the resulting 24 kbps bitrate, which is an optimum compromise in terms of audio quality, radio transmission time and power consumption. In case of simplex or half-duplex streaming, the bitrate can be increased.

The peripheral module can also be connected to the STBLESensor app for Android and iOS to perform full-duplex speech communication with the smartphone. Both nodes can stream audio at 16 kHz at the same time.

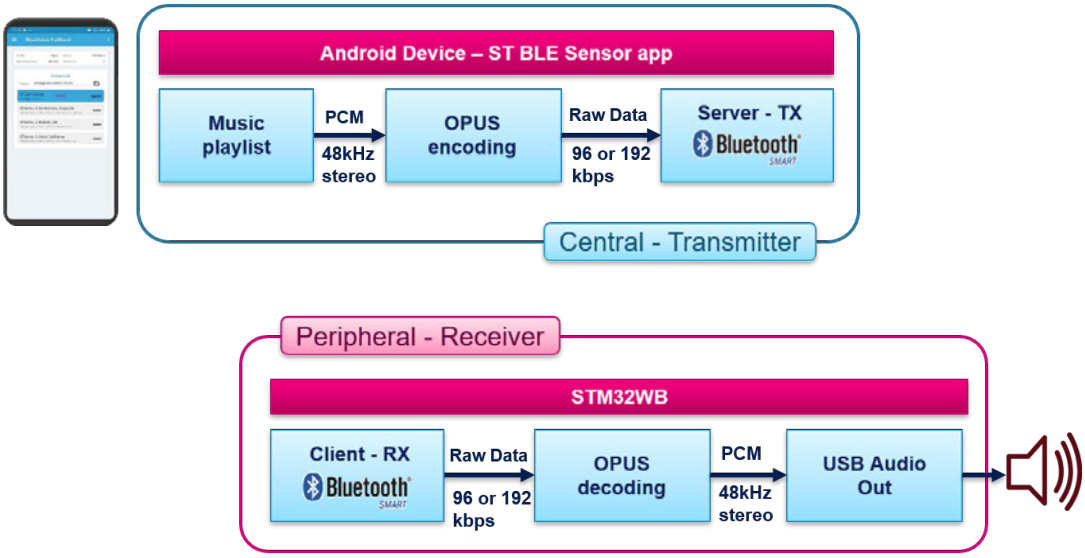
Figure 3. **FP-AUD-BVLINKWB1 full-duplex processing chain**



An easy-to-use set of APIs is available in the STM32_WPAN middleware to customize every encoder and decoder parameter allowing to develop a full-band application.

The full-band application consists in a stereo music streaming from an Android device (supporting BLE 4.2 or higher) running the STBLESensor app to a STM32WB acting as peripheral node. The receiver must be connected to a PC via USB and it is recognized as a 48 kHz-2channel microphone. The mobile device processes a 48 kHz stereo wav file; every 10 ms, audio is encoded using Opus and sent over BLE through a dedicated service and characteristic exported by the app. The peripheral node parses the received data, decompresses and sends them via USB to the PC; any freeware or commercial audio recording software can be used to record and listen to the received stereo music.

Figure 4. **FP-AUD-BVLINKWB1 full-band 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 to deliver a single output bit at the high sampling rate. The noise shaping mechanism ensures appropriate quantization of inconsistent noise density 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 an intermediate step between PDM and the compressed audio data actually sent over BLE. The decimation filters typically used to convert the PDM stream to PCM data is available as a middleware library in the FP-AUD-BVLINKWB1 function pack.

The conversion library (PDMLib) consists of a decimation filter that converts 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.

2.5.2 Opus

Opus is an open, royalty-free, and highly versatile audio codec. It can be used for different kinds of applications like 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 bit-rate narrow-band speech at 6 kbit/s to 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, LP techniques such as code-excited linear prediction (CELP) codify low frequencies more efficiently than in transform domain techniques such as MDCT.

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 bit-rate narrow-band speech to very high quality stereo music. Supported features are:

- Bit-rates from 6 kbps to 510 kbps
- Sampling rates from 8 kHz (narrow-band) to 48 kHz (full-band)
- Frame sizes from 2.5 ms to 60 ms
- Support for constant bit-rate (CBR) and variable bit-rate (VBR)
- Audio bandwidth from narrow-band to full-band
- Support for speech and music
- Support for mono and stereo
- Support for up to 255 channels (multi-stream frames)
- Dynamically adjustable bit-rate, audio bandwidth and frame size
- Good loss robustness and packet loss concealment (PLC)
- Floating point and fixed-point implementation

In the FP-AUD-BVLINKWB1 an Opus encoder and a decoder are created for the central and the peripheral module.

The default parameters for the full-duplex application are:

- for the encoder (in the firmware as well as in ST BLE Sensor app):
  - Opus application type: OPUS_APPLICATION_VOIP
  - Bitrate: 24000 bps
  - Channels: 1 (mono)
  - Opus complexity: 0
  - Audio frame size: 20 ms
  - Audio sampling frequency: 16000 Hz
- for the decoder (in the firmware as well as in ST BLE Sensor app):
  - Bitrate: 24000 bps
  - Channels: 1 (mono)
  - Audio frame size: 20 ms
  - Audio sampling frequency: 16000 Hz

The default parameters for the full-band application are:

- for the encoder (STBLESensor app):
  - Opus application type: OPUS_APPLICATION_AUDIO
  - Bitrate: 96000 or 192000 bps
  - Channels: 2 (stereo)
  - Opus complexity: 4
  - Audio frame size: 10 ms
  - Audio sampling frequency: 48000 Hz
- for the decoder (FW BVLPeripheral_FullBand):
  - Bitrate: 96000 or 192000 bps
  - Channels: 2 (stereo)
  - Audio frame size: 10 ms
  - Audio sampling frequency: 48000 Hz

2.6 BlueVoiceOPUS profile description

2.6.1 Generic Access Profile (GAP)

The FP-AUD-BVLINKWB1 application deploys a connection-based communication paradigm providing a permanent point-to-point link between two devices, managed by the Generic Access Profile (GAP). Data sent via BLE are organized through the Generic Attribute Profile (GATT), an additional protocol layer.

The Bluetooth Specification 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 5. 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.

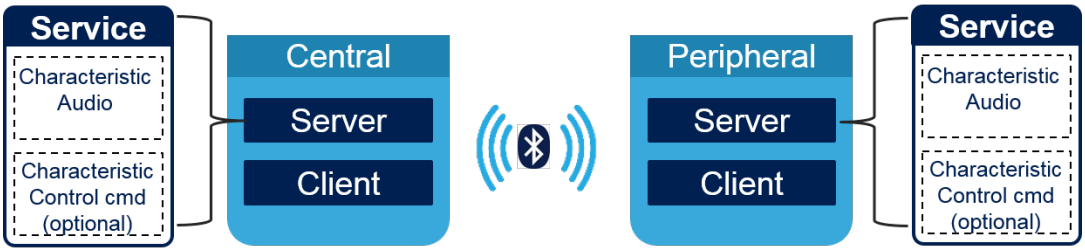
GATT defines client and server roles for interacting devices independent of the GAP master/central and slave/peripheral roles:

- **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 6. BlueVoice Profile GATT role assignment in a bidirectional system

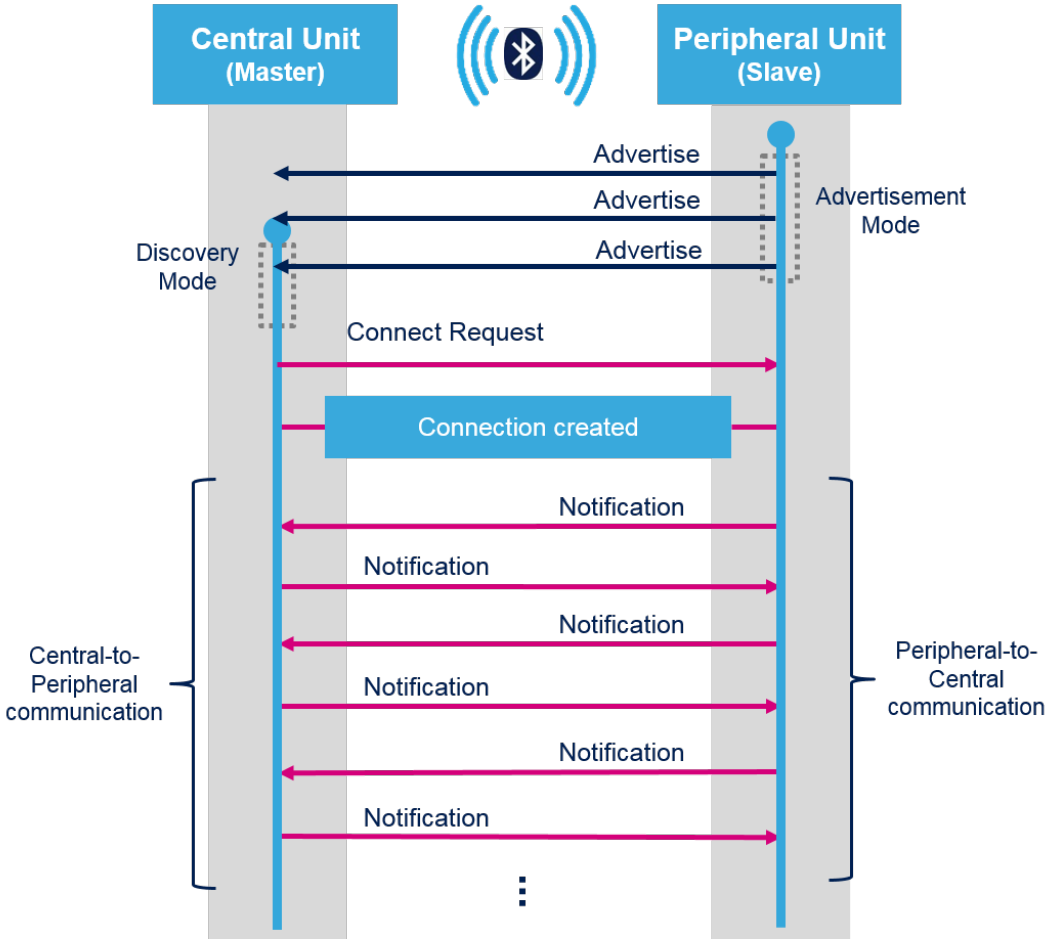


In both directions, streaming audio data transmission is based on periodic server-to-client notifications which do not require a request or response from the receiving device. Server-initiated updates are sent as asynchronous notification packets which include a characteristic value attribute along with 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 7. 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-BVLINKWB1 defines a vendor-specific service named BlueVoiceOPUS service which exposes a user voice to a client device. Depending on the running application, a characteristic changes between audio or music characteristic.

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, for full-duplex application (BVLPeripheral and BVLCentral projects), consists of an Audio characteristic to expose actual compressed audio data and a Ctrl characteristic to expose control information used to implement optional commands. The same service is exposed by the ST BLE Sensor app if used in full-duplex mode.

- **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: 00000001-0002-11e1-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: 00000002-0002-11e1-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.

The BlueVoceOPUS service is also used also for the full-band demo, shown by the ST BLE Sensor app but the audio characteristic is replaced by the music characteristic used to send compressed audio stereo at 48 kHz.

- **Music Characteristic**
  - Att1 contains the music 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: 00000011-0002-11e1-ac36-0002a5d5c51b).
  - Att2 contains the characteristic value that is the compressed stereo music data. Its attribute fields are:
    - UUID: the same UUID present in the last 16 bytes of the characteristic definition attribute value.
    - Permissions: None.
    - Value: the actual control data.
  - Att3 contains the client characteristic configuration, defining how the characteristic can 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 2. BlueVoiceOPUS UUID summary table

UUID name	UUID
bv_opus_service_uuid	00000000-0001-11e1-9ab4-0002a5d5c51b
bv_opus_audio_char_uuid	00000001-0002-11e1-ac36-0002a5d5c51b
bv_opus_ctrl_char_uuid	00000002-0002-11e1-ac36-0002a5d5c51b
bv_opus_music_char_uuid	00000011-0002-11e1-ac36-0002a5d5c51b

2.7 BlueVoiceOPUS service description

2.7.1 Overview

BlueVoiceOPUS is a BLE service which implements a vendor-specific profile for audio streaming over BLE and is included in the *STM32_WPAN* middleware.

The Opus audio codec is used to compress acquired audio obtaining a low bitrate and mantaining a optimum audio quality.

The BVOPUS service is provided as source code and requires two more modules: the STM32_WPAN middleware included in the STM32CubeWB software package and the official Opus audio codec package (available at https://www.opus-codec.org) included in the FP-AUD-BVLINKWB1 as third-party middleware.

BlueVoiceOPUS includes:

- *bvopus_service_stm*: the core of BlueVoiceOPUS profile which manages all the functions related to BLE, from the creation of the service and characteristics to the data sending and receiving. It inlcudes also a simple protocol for the encoded data packetization and parsing (for further information see Section 2.7.2.4 BlueVoiceOPUS transfer protocol).
- *opus_interface_stm*: 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.

2.7.2 How to use

2.7.2.1 Initialization and configuration

The BlueVoiceOPUS profile can implement a transmitter, a receiver or both in case of full-duplex communication.

To stream audio, the BlueVoiceOPUS service and characteristics must be created by calling `BVOPUS_STM_Init`, which includes the `BluevoiceOPUS_AddService` and `BluevoiceOPUS_AddChar` functions; the UUIDs are defined in *bvopus_service_stm.c* file.

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 the function returns `BV_OPUS_SUCCESS`, the BLE profile has been correctly created.

Note: If the module has to decode the received audio only and never stream data, the previous part is not necessary (e.g. in the FullBand example).

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

The default parameters used for full-duplex application are:

```
EncConfigOpus.application = OPUS_APPLICATION_VOIP;
EncConfigOpus.bitrate = 24000;                              /* 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 = 24000;                              /* 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 */
```

The default parameters used for the full-band application are:

```
DecConfigOpus.bitrate = bitrate; /* 96000 or 192000 bps */
DecConfigOpus.channels = AUDIO_CHANNELS_OUT; /* 2 channel, stereo*/
DecConfigOpus.ms_frame = AUDIO_OUT_MS; /* 10 ms */
DecConfigOpus.sample_freq = AUDIO_OUT_SAMPLING_FREQUENCY; /* 48000 Hz */
```

The encoder and decoder can be initialized by calling `BVOPUS_CodecEncInit(&EncConfigOpus)` and `BVOPUS_CodecDecInit(&DecConfigOpus)`. If both functions return `BV_OPUS_SUCCESS`, the BlueVoiceOPUS profile has been correctly configured. Depending on the parameters chosen, the init functions allocate an amount of memory returned by the relevant API called internally:

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

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

If the initialization function returns `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:

- `OPUS_IF_ENC_Set_VBR`: sets a variable bitrate (set by default)
- `OPUS_IF_ENC_Set_CBR`: sets a constant bitrate
- `OPUS_IF_ENC_Force_CELTmode`: forces Opus to use the Celt codec only (by default an hybrid mode is set)
- `OPUS_IF_ENC_Force_SILKmode`: forces Opus to use the Silk codec only (by default an hybrid mode is set)
- `OPUS_IF_ENC_Set_Complexity`: sets a new complexity without reinitialize the encoder
- `OPUS_IF_ENC_Set_Bitrate`: sets a new bitrate for the encoder

Note: After calling this function, it is important to reinitialize the encoder by calling `BVOPUS_CodecEncInit`.

The encoder and decoder can be deinitialized by calling `OPUS_IF_ENC_Deinit` or `OPUS_IF_DEC_Deinit`.

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(void)` API. The control notification is then 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_CONTROL, BV_OPUS_ENABLE_NOTIF_REQ}`). As soon as a node receives the request, it can enable the audio notification to the requestor with the function `BluevoiceOPUS_EnableAudio_Notif(void);`.

If the audio notification is correctly enabled, the module can start streaming audio. To set up a half-duplex or full-duplex transmission, 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 automatically compresses, packetizes (see Section 2.7.2.4 BlueVoiceOPUS transfer protocol for further information) and sends 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 complete.

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

2.7.2.3 *Full-Band receiver application*

The Full-Band application consists in a peripheral module to be connected to an Android device that supports BLE 4.2, running STBLESensor app and acting as a central node. If supported, the 2M PHY feature is enabled to reach a higher bitrate (96000 or 192000 bps, accordingly).

The STM32WB discovers the BlueVoiceOPUS profile exposed by the central node and enables its characteristics. Then user can start and stop music streaming from the mobile device.

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 complete.

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

2.7.2.4 *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.

The maximum payload for the BLE packet is set to 150 Bytes and 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 achieve this, a simple transfer protocol has been implemented and included in the *bvopus_service_stm.c* file.

The BlueVoiceOPUS transfer protocol indicates when the encoded data starts and ends so that the receiver is able to rebuild the compressed buffer and decode it: a single byte is added as first byte of each BLE packet, the remaining 19 bytes or more, depending on the MTU chosen, 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 audio encoded data plus one byte for the transfer protocol;
- `BluevoiceOPUS_TP_Parse`: extracts the payload and rebuilds the Opus encoded frame.

2.8 FP-AUD-BVLINKWB1 application description

FP-AUD-BVLINKWB1 central and peripheral applications are provided in the Projects directory. Ready-to-build projects are available for P-NUCLEO-WB55 or STM32WB5MM-DK and different IDEs.

To show full-duplex speech communication over BLE using the BlueVoiceOPUS Profile specification, BVLCentral and BVLPeripheral 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.

Being 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 decoded PCM streaming over USB

After connection, the active communication channel is controlled by the SW1 user button on the P-NUCLEO-WB55.

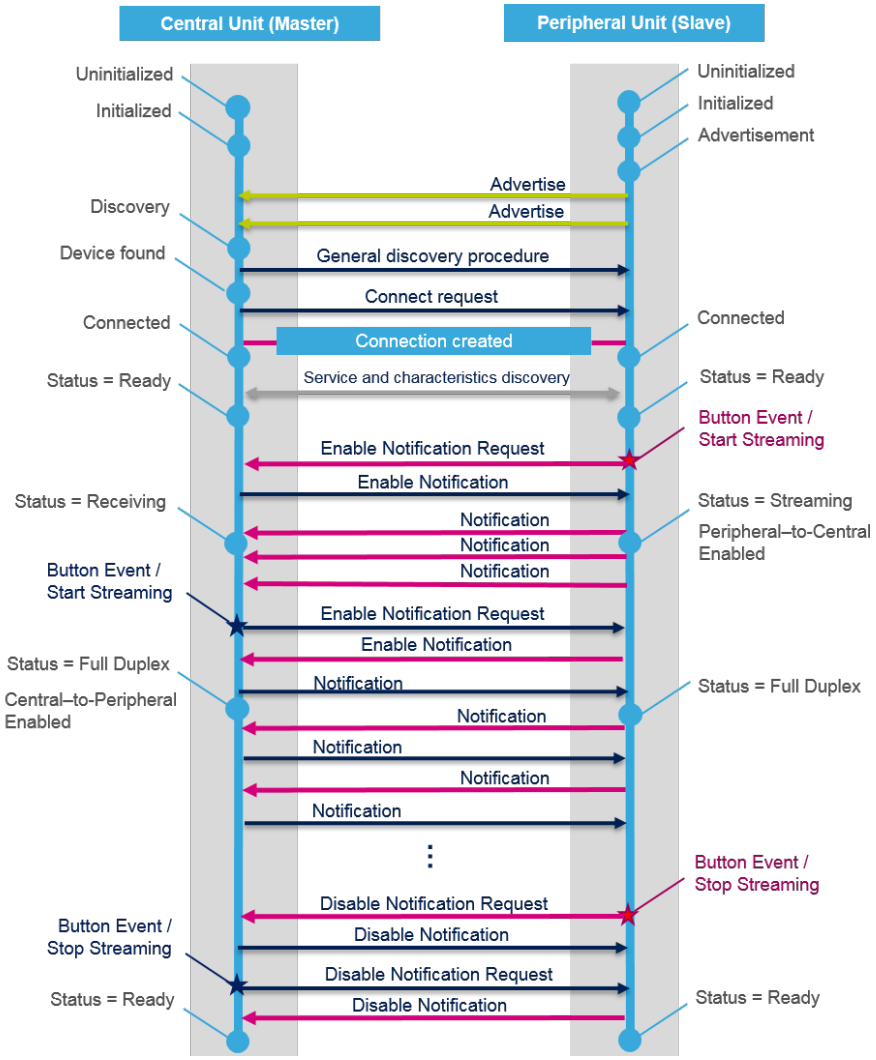
The diagram below shows how the communication link is set and the audio streaming is performed. Once connected, pressing the user button starts and stops the audio streaming. If the node is receiving, a start streaming event creates full-duplex communication.

Depending on the status of the two modules, the LEDs show:

- advertising/discovery: green LED blinking
- connection: blue LED slow blinking
- streaming: blue LED normal blinking
- Receiving: blue LED steady on (not blinking)
- Full-duplex: blue LED fast blinking on both modules

The BVLCentral module can be replaced by a mobile device running ST BLE Sensor app (4.9.0 or higher) performing a full-duplex communication between the board and a smartphone.

Figure 8. FP-AUD-BVLINKWB1: central-peripheral communication diagram



To set up a stereo music streaming, the BVLPeripheral_FullBand project is included in the package. The STM32WB acts as peripheral and sends advertising messages waiting for the STBLESensor app to create the connection (green LED is blinking).

On the mobile device the connection is created by choosing the BVFBAND device (blue LED is blinking slow): STM32WB is ready to receive audio, decode and send it via USB; the blue LED blinks faster while audio data are being received.

2.8.1 Initialization

During the first phase, all the hardware peripherals needed are initialized: BLE, USB audio output, 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/or decoder to act as a transmitter and/or a receiver. In the full-duplex application, the BLE service and characteristics are then created using the relevant API (further information at Initialization and configuration).

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

Each application is structured in different tasks for each BLE functionality, in particular:

- BVOPUS_CEN:
  - CFG_TASK_START_SCAN_ID sets the device in discovery mode
  - CFG_TASK_LINK_CONFIG_ID creates a BLE connection with the discovered BVOPUS_PER device
  - CFG_TASK_SEND_DATA_ID encodes PCM data and sends them via BLE
- BVOPUS_PER:
  - CFG_TASK_START_ADV_ID sets the device in advertising mode
  - CFG_TASK_LINK_CONFIG_ID sets the MTU and discovers BlueVoiceOPUS service and characteristics if exported by the central module
  - CFG_TASK_SEND_DATA_ID encodes PCM data and sends them via BLE
- BV_FULLBAND:
  - CFG_TASK_START_ADV_ID sets the device in advertising mode
  - CFG_TASK_GAP_CONFIG_ID runs gap procedures, setting PHY and updates connection parameters
  - CFG_TASK_GATT_CONFIG_ID runs gatt procedures, setting MTU preferences and discovering BlueVoiceOPUS service and characteristics exported by the STBLESensor app

In the infinite main loop, the UTIL_SEQ_Run manages the scheduler.

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 advertising or discovery mode. 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 and meet Android requirements:

- Full-duplex:
  - min. connection interval: 10 ms
  - max. connection interval: 21.25 ms
- Full-band:
  - min. connection interval: 11.25 ms
  - max. connection interval: 11.25 ms

As soon as the connection is established, both modules run a service and characteristic discovery to obtain the BlueVoiceOPUS profile handle exported by the other node.

At the end of this procedure, the control characteristic is enabled on both nodes to allow starting and stopping streaming requests, in case of full-duplex configuration; both the characteristics are enabled in case of full-band configuration.

The application is now in BV_STATUS_READY status waiting for a request that can create simplex, half-duplex or full-duplex communication or to receive stereo music.

2.8.3 Full duplex audio streaming

User button 1 triggers a start/stop streaming. As soon as the button is pressed, a message is sent to request the audio notification enabling to the receiver and its 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.

2.8.4 Profiling

Profiling in terms of memory size and computational power is shown below.

The following pictures consider just the Opus codec, configured according to the two applications available in the FP-AUD-BVLINKWB1.

Figure 9. Full-duplex configuration – Opus memory footprint

Compiler: IAR v8.32
Optimization: High-size
Code: Opus codec
*VLA: Variable Lenth Array

Audio channel = 1	Flash(bytes) ro code + ro data	RAM (bytes)			
		RW data	Stack	Heap	
Encoder	158500	140	~15000	38436	VLA* ~15000
Decoder				17776	

Figure 10. Full-duplex configuration – computational power

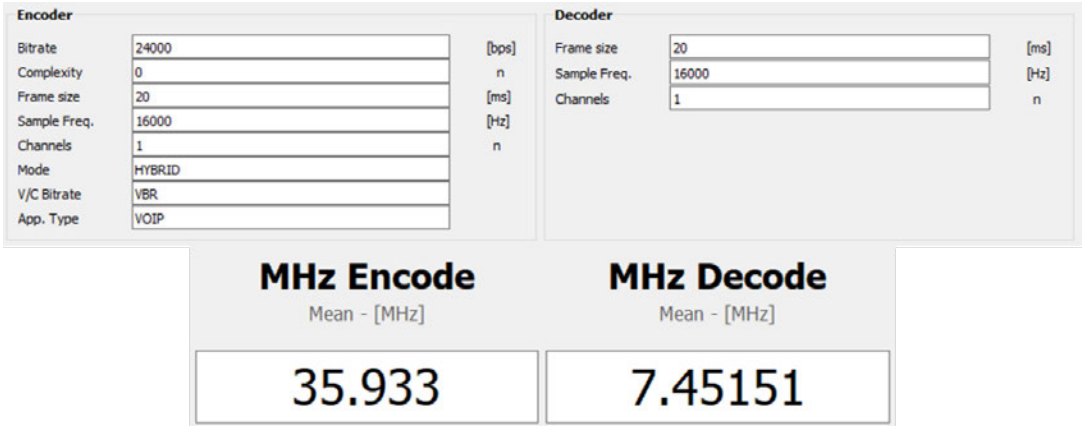


Figure 11. Full-Band configuration – Opus memory footprint

Compiler: IAR v8.32
Optimization: High-size
Code: Opus codec
*VLA: Variable Lenth Array

Audio channel = 2	Flash(bytes) ro code + ro data	RAM (bytes)			
		RW data	Stack	Heap	
Decoder	77900	--	~4000	26496	VLA* ~5000

Figure 12. **Full-Band configuration (bitrate 96 kbps) – computational power**

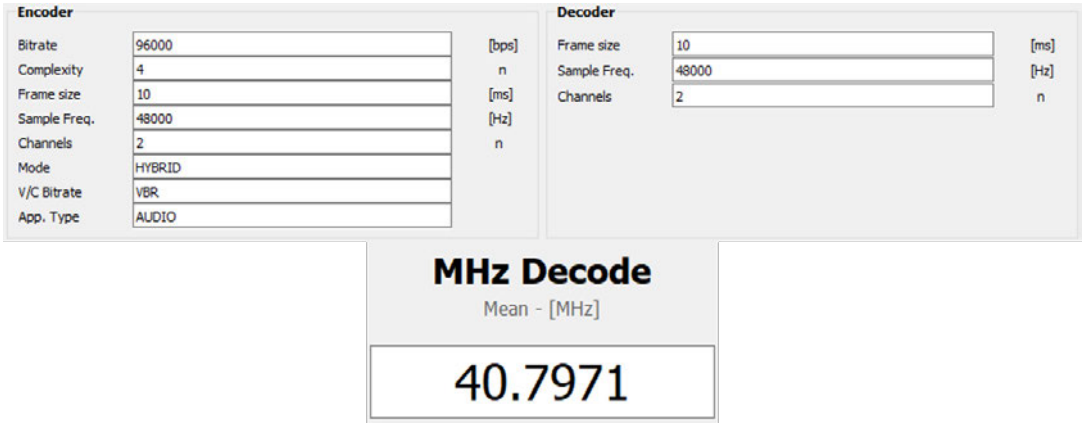
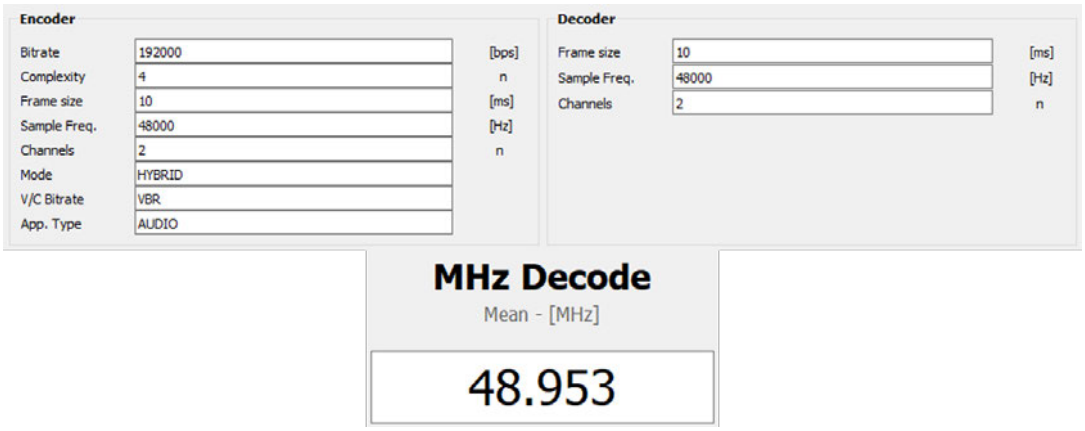


Figure 13. **Full-Band configuration (bitrate 192 kbps) – computational power**



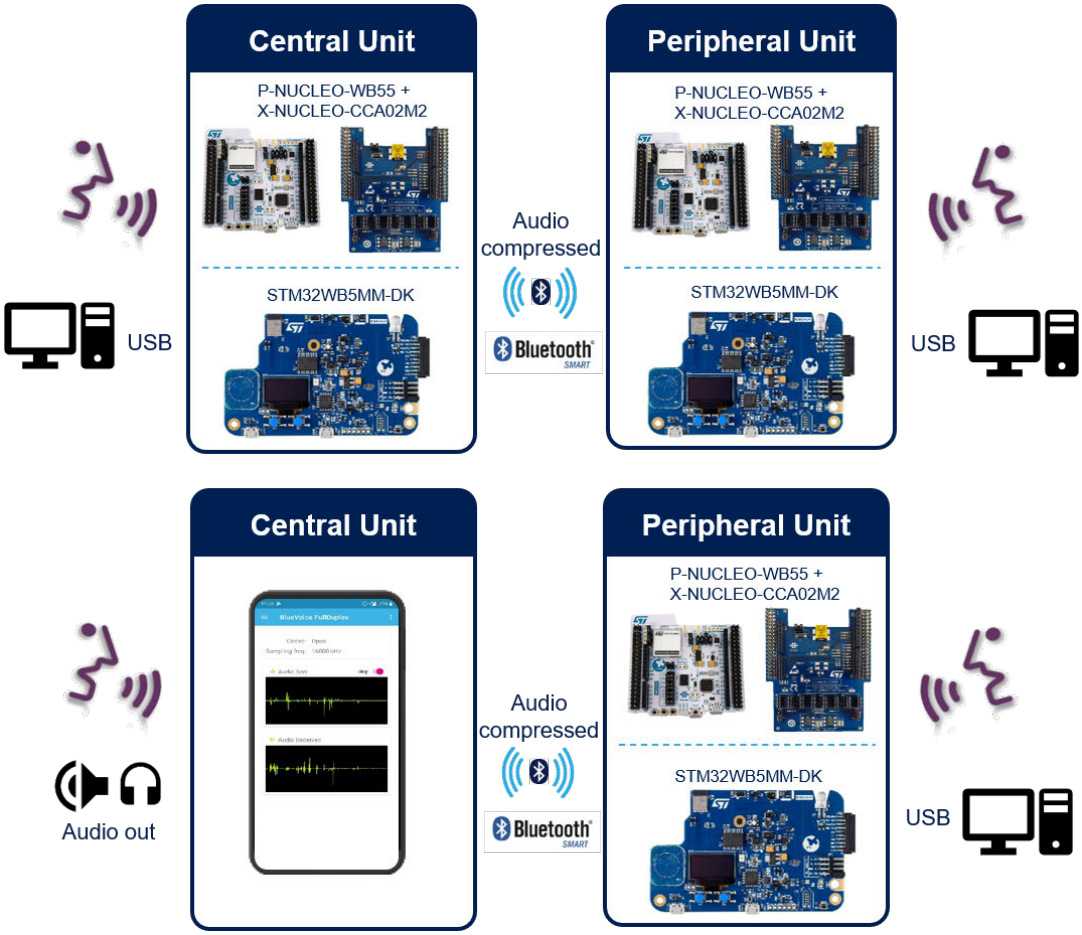
3 System setup guide

3.1 Hardware description

The FP-AUD-BVLINKWB1 application is based on ready-to-build projects available for P-NUCLEO-WB55, together with a microphone expansion board (X-NUCLEO-CCA02M2) or for STM32WB5MM-DK.

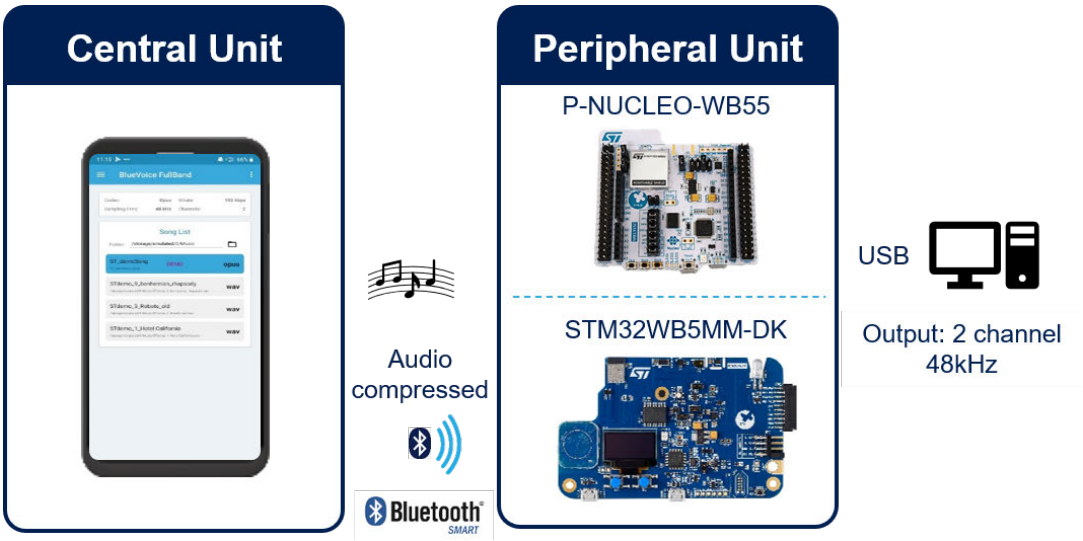
The full-duplex demo can be set up by using a BVLPeripheral and a BVLCentral node assembled as shown below. Alternatively, a mobile device running STBLESensor app can be used as central unit.

Figure 14. FP-AUD-BVLINKWB1 full-duplex system overview



The full-band demo can be set up by using the BVLPeripheral_Fullband node and an Android device running STBLESensor app, as shown below.

Figure 15. **FP-AUD-BVLINKWB1 full-band system overview**



3.1.1 P-NUCLEO-WB55

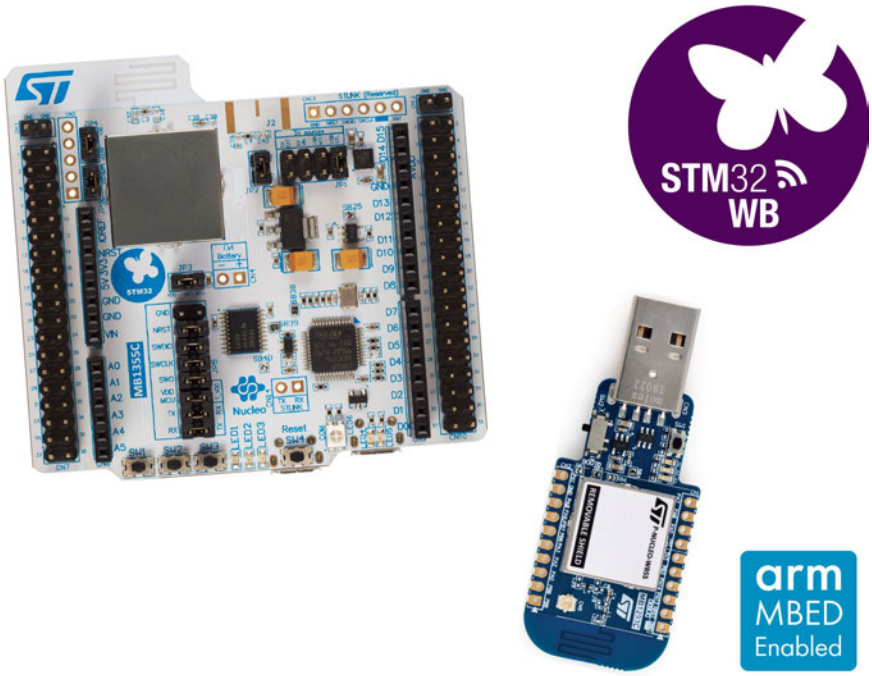
The P-NUCLEO-WB55 pack is a multi-protocol wireless and ultra-low-power device embedding a powerful and ultra-low-power radio compliant with the Bluetooth® Low Energy (BLE) SIG specification v5.0 and with IEEE 802.15.4-2011.

The pack consists of a Nucleo-68 development board and a USB dongle.

It features:

- STM32WB microcontroller in a VFQFPN68 package
- 2.4 GHz RF transceiver supporting Bluetooth® specification v5.0 and IEEE 802.15.4-2011 PHY and MAC
- Dedicated Arm® 32-bit Cortex® M0+ CPU for real-time Radio layer
- Three user LEDs
- Three user buttons and one reset button
- Board connector: USB user with Micro-B
- Arduino™ Uno V3 connector
- ST morpho connectors
- Integrated PCB antenna or footprint for SMA connector
- Flexible power-supply options: ST-LINK USB VBUS or external sources
- On-board socket for CR2032 battery
- On-board ST-LINK/V2-1 debugger/programmer with USB re- enumeration capability: mass storage, virtual COM port and debug port
- Comprehensive free software libraries and examples available with the STM32Cube package
- Support of a wide choice of Integrated Development Environments (IDEs), including IAR™, Keil®, GCC-based IDEs, Arm® Mbed™

Figure 16. **P-NUCLEO-WB55 development pack**



3.1.2 X-NUCLEO-CCA02M2 expansion board

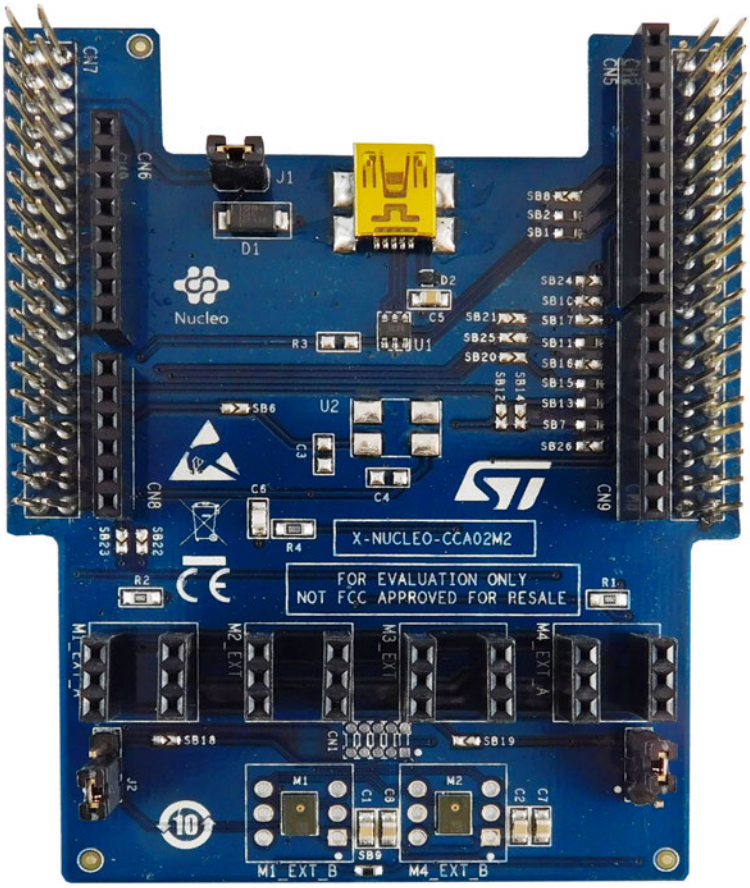
The X-NUCLEO-CCA02M2 expansion board has been designed around MP34DT06J digital MEMS microphone.

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 17. **X-NUCLEO-CCA02M2 expansion board**



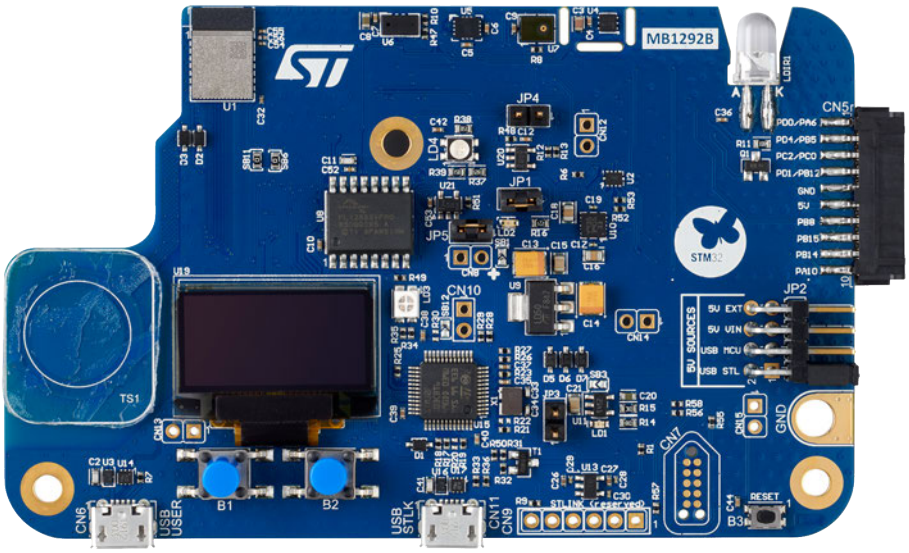
3.1.3 STM32WB5MM-DK

The STM32WB5MM-DK Discovery kit is a development platform for the STM32W5MMG module based on the Arm® Cortex®-M4 and Arm® Cortex®-M0+ cores.

The STM32 device is a multi-protocol wireless and ultra-low-power device embedding a powerful and ultra-low power radio compliant with the Bluetooth® Low Energy (BLE) SIG specification v5.2 and with IEEE 802.15.4-2011.

An ST-LINK/V2-1 is integrated, as well as an embedded in-circuit debugger and programmer for the STM32 MCU and the USB Virtual COM port bridge.

Figure 18. **STM32WB5MM-DK Discovery kit**



3.2 Software description

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

- FP-AUD-BVLINKWB1: an application based on STM32Cube that demonstrates audio streaming over BLE. The FP-AUD-BVLINKWB1 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-STR) toolchain plus ST-LINK
  - STM32CubeIDE plus ST-LINK

3.3 Hardware and software setup

3.3.1 Hardware setup

The following hardware components are needed for full-duplex communication between two boards. Just one module is requested if the connection is established with a mobile device running ST BLE Sensor app.

1. Two P-NUCLEO-WB55 development platforms or two STM32WB5MM-DK
2. If the P-NUCLEO-WB55 is used, two digital MEMS microphone expansion boards (order code: X-NUCLEO-CCA02M2)
3. Two USB type A to Micro-B USB cables to power up the modules and for audio streaming over USB

The following hardware components are needed for full-band application:

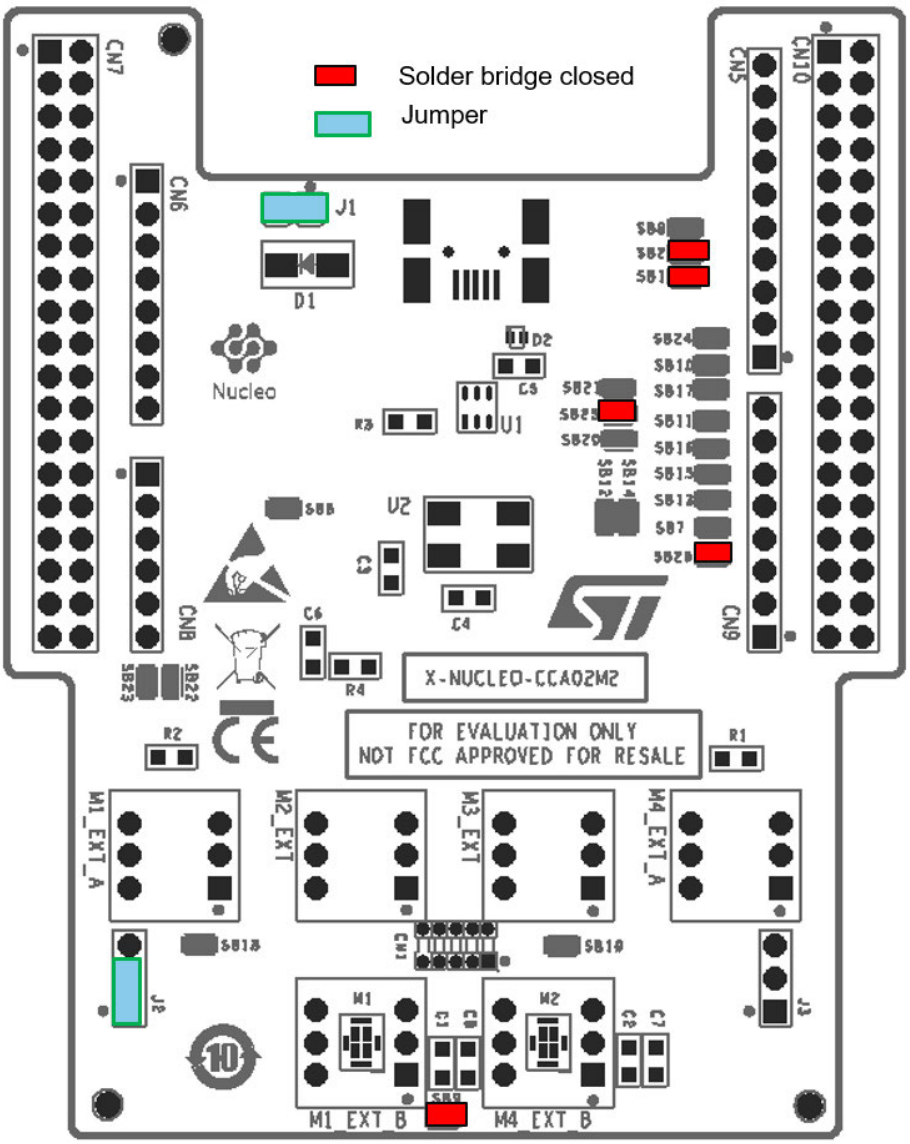
1. one P-NUCLEO-WB55 development platform or one STM32WB5MM-DK
2. one USB type A to Micro-B USB cables to power up the module and for audio streaming over USB

3.3.1.1 X-NUCLEO-CCA02M2 configuration

The X-NUCLEO-CCA02M2 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-CCA02M2 user manual, UM2631, on www.st.com).

With the P-NUCLEO-WB55 used for full-duplex communication, only one microphone is acquired through the SAI.

Figure 19. **X-NUCLEO-CCA02M2 hardware configuration for P-NUCLEO-WB55**



3.3.2 Software setup

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 STM32WB. The following section describes the demo setup in a Windows 10 environment.

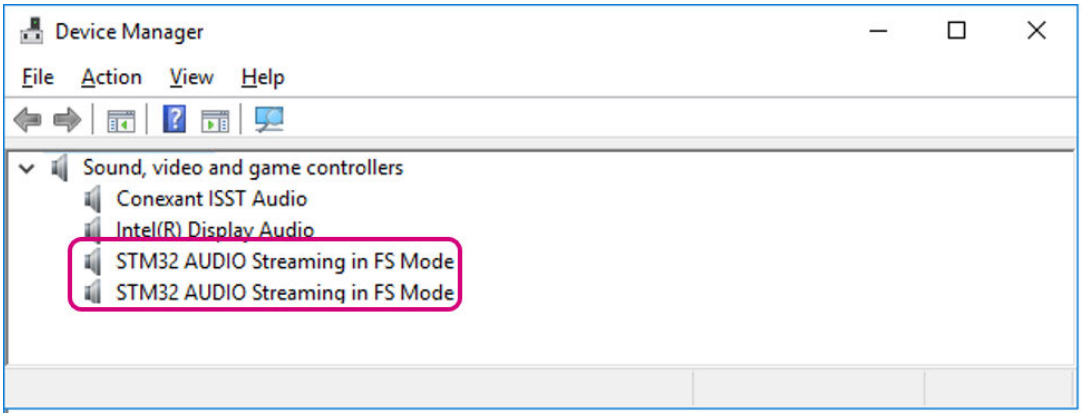
3.3.2.1 Development toolchains and compilers

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 10*

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 jumper (JP1 on the P-NUCLEO-WB55, JP2 on STM32WB5MM-DK) from USB STL to USB MCU and connect the USB_USER to a PC via a micro USB cable. The two nodes are recognized correctly in the Device Manager, as shown below.

Figure 20. Device Manager: STM32WB Nucleo board microphone recognition

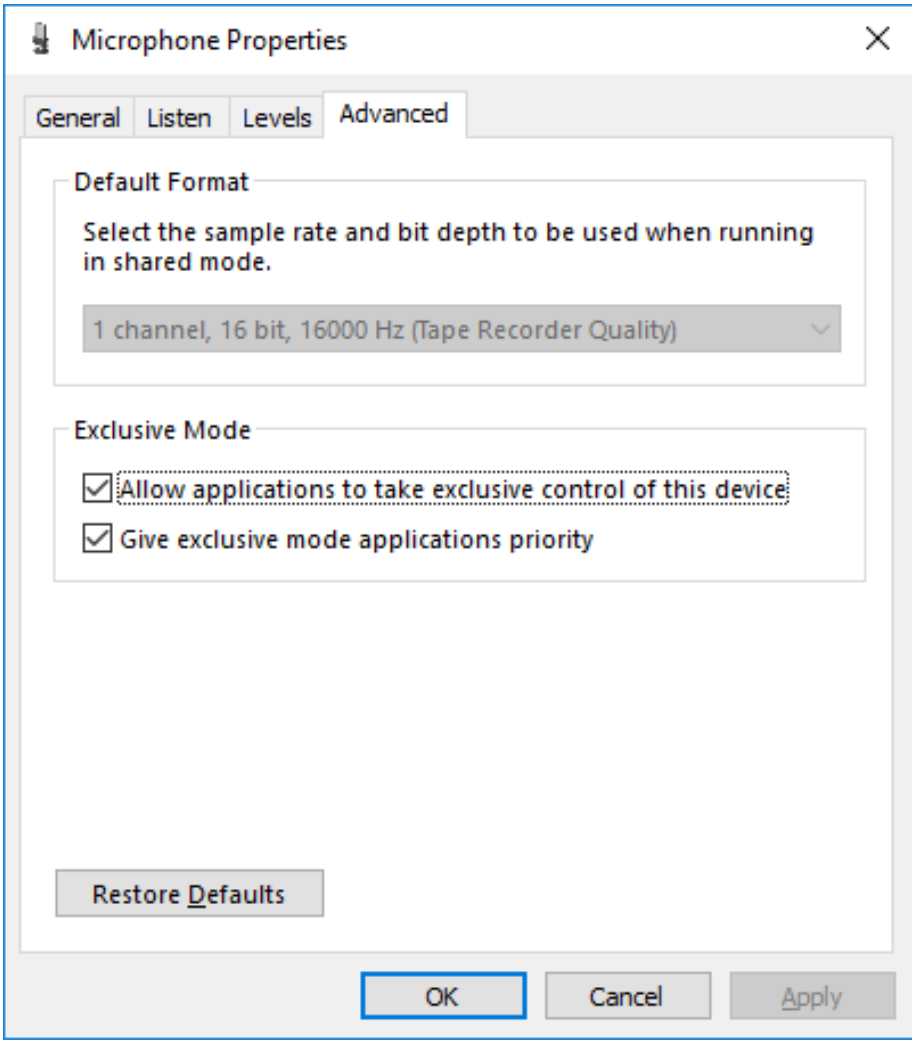


Right-click on the volume icon in the Windows task bar (on the bottom right side of the screen) and choose “recording device”. Select STM32 microphone and click on Properties.

In the Advanced tab, a summary of the current device setup appears, showing the sampling frequency and number of channels. The module running full-duplex application should be recognized as a "1 channel, 16000 Hz" microphone (if the audio sampling frequency set in the firmware is 16 kHz); instead, a full-band module should be recognized as a "2 channel, 48000 Hz" microphone.

Note: If the audio sampling frequency is changed, for example if you re-flash the same board from full-duplex to full-band demo, the audio USB driver must be uninstalled from the device manager. After that, you have to unplug and replug the USB cable and the driver is automatically reinstalled with the correct settings.

Figure 21. Advanced tab: microphone properties



3.3.3 Full-duplex demo setup for two STM32WB boards

This section describes how to set the FP-AUD-BVLINKWB1 full-duplex application up to demonstrate audio transmission over Bluetooth low energy between two STM32WB boards.

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.

The full-duplex streaming can be performed between two P-NUCLEO-WB55 or two STM32WB5MM-DK, one flashed with the BVLPeripheral project and the other with BVLCentral project included in the software package.

3.3.3.1 STM32WB development board and expansion board setup

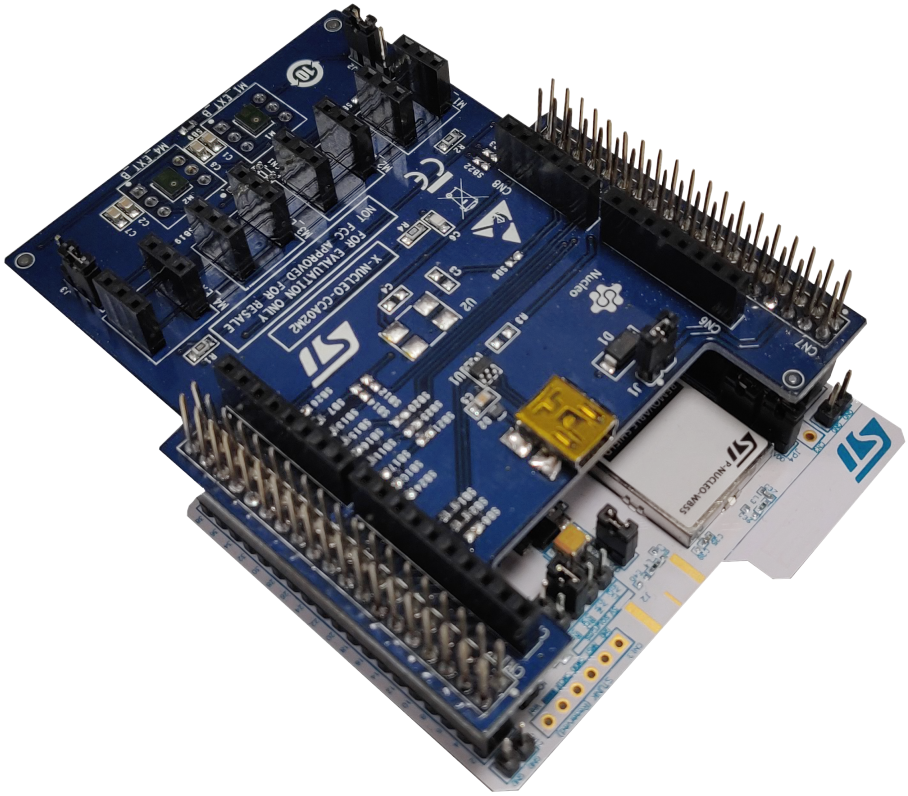
The P-NUCLEO-WB55 and STM32WB5MM-DK integrate the ST-LINK/V2-1 debugger/programmer.

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.

To flash the MCU, move JP1 on the P-NUCLEO-WB55 or JP2 on the STM32WB5MM-DK to USB STL and connect the USB STLK. To make the board to be recognized as a standard microphone, move the jumper to USB MCU and connect the USB USER.

The X-NUCLEO-CCA02M2 microphone expansion board can be easily connected to the P-NUCLEO-WB55 board through the ST morpho extension connector.

Figure 22. X-NUCLEO-CCA02M2 connected to P-NUCLEO-WB55



3.3.3.1.1 PC audio recording utility sample: Audacity

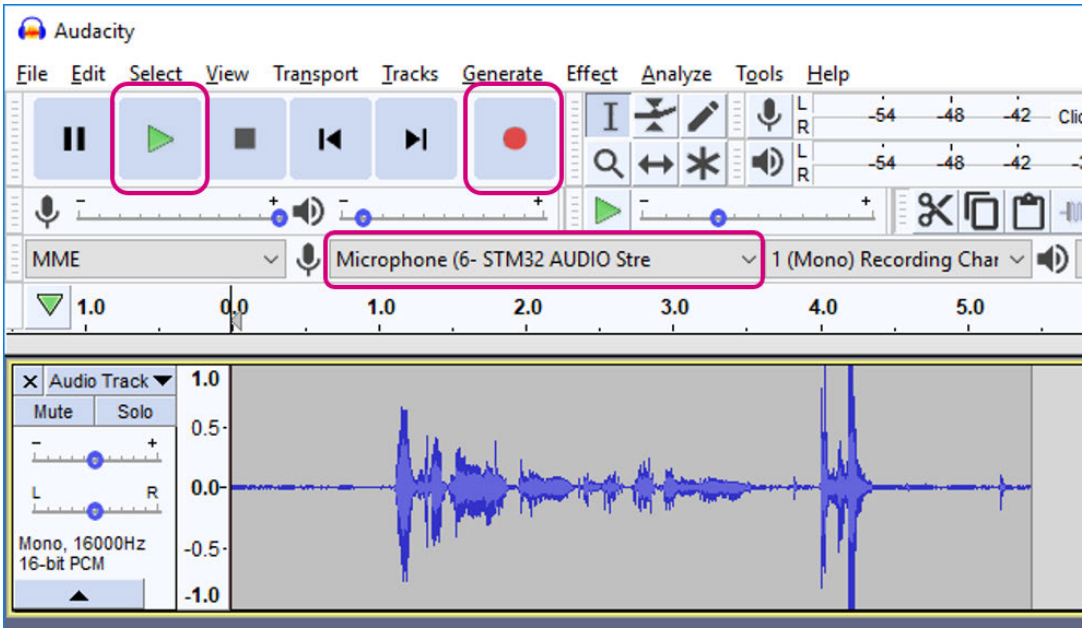
Audacity® is an open source, cross-platform program for recording and audio editing environment.

To start audio recording, first check if the audio input device is STM32 AUDIO streaming in FS mode. Then set the default audio settings depending on the STM32 microphone.

In Audacity, open [**Edit/Preferences**] and select [**Quality**] tab, choose the correct sampling frequency and sample format (16 kHz 16-bit for voice or 48 kHz 16-bit for music).

Audacity is now ready to start recording, if you want to hear the audio while recording it open [**Edit/Preferences**], select [**Recording**] tab and check [**Software playthrough of input**].

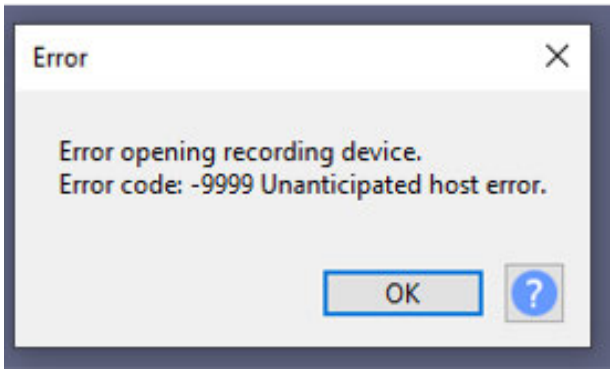
Figure 23. Audacity for Windows



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

Note: If the following error appears, delete the STM32 microphone driver from Windows Device Manager, then unplug and replug the USB cable from the board. The driver will automatically and correctly be reinstalled.

Figure 24. Audacity for Windows



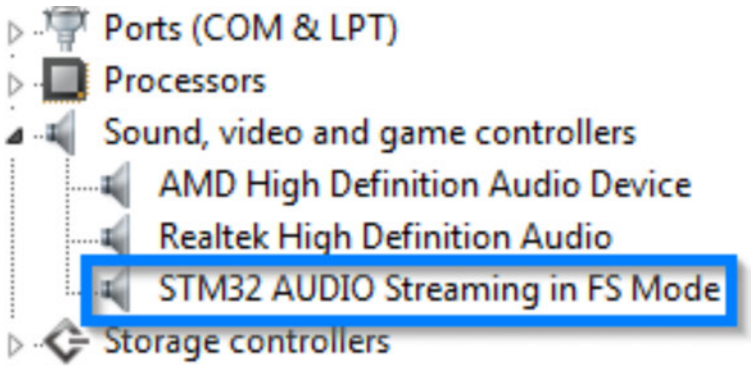
3.3.3.1.2 Module setup

Step 1. Flash the module with BVLCentral binary by connecting USB ST_LINK (for P-NUCLEO-WB55, JP1 on USB STL; for STM32WB5MM-DK, JP2 on USB STL)

Step 2. Connect the USB_USER connector (on P-NUCLEO-WB55, move JP1 to USB MCU; on STM32WB5MM-DK, move JP2 to USB MCU) of the Central node to the PC.

The STM32 Audio Streaming device appears in the Device Manager.

Figure 25. STM32WB recognized as Audio Streaming

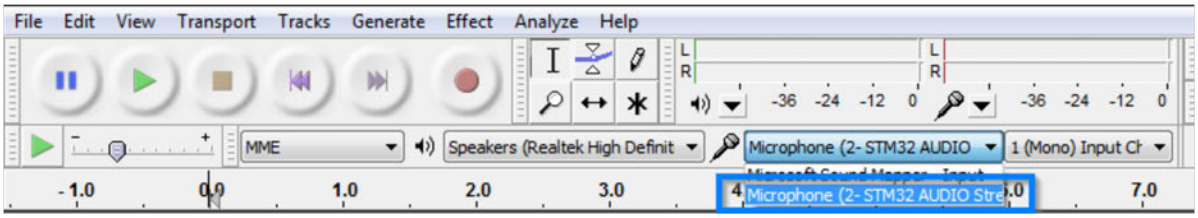


Step 3. Open Audacity.

An STM32 microphone appears in the Input selector.

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

Figure 26. Central unit microphone in Audacity

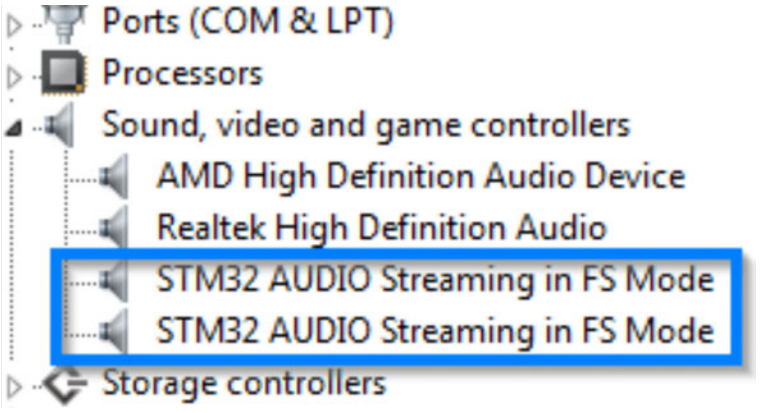


Step 4. Flash the other module with BVLPeripheral binary by connecting USB ST_LINK (for P-NUCLEO-WB55, JP1 on USB STL; for STM32WB5MM-DK, JP2 on USB STL).

Step 5. Connect the USB_USER connector (for P-NUCLEO-WB55, move JP1 on USB MCU, on STM32WB5MM-DK, move JP2 on USB MCU) of the Peripheral node to the PC.

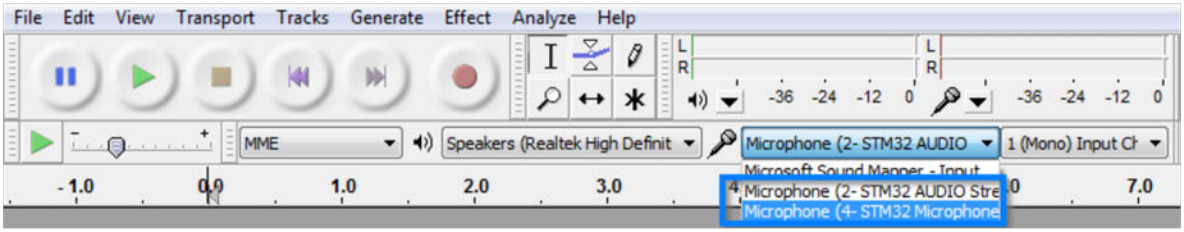
A second STM32 AUDIO Streaming device appears in Device Manager.

Figure 27. STM32 recognized as Audio Streaming



Step 6. 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

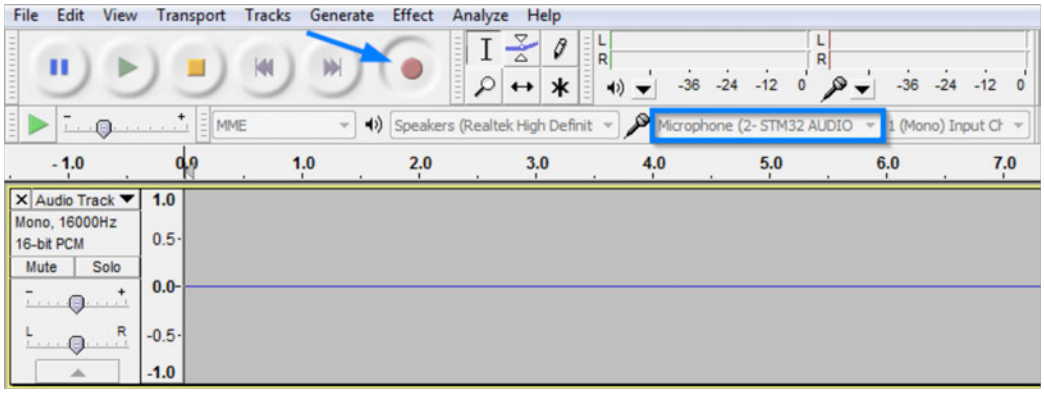


3.3.3.1.3 Peripheral to central recording

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

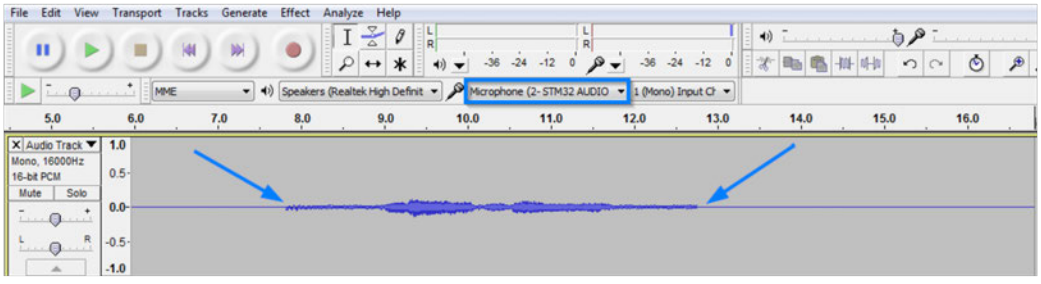
Step 2. Click Record to start silent recording

Figure 29. Audacity silent recording from Central unit USB stream



Step 3. Click Peripheral unit user button 1
Peripheral unit streams voice to the **Central unit**

Figure 30. Audacity recording voice coming from Peripheral unit



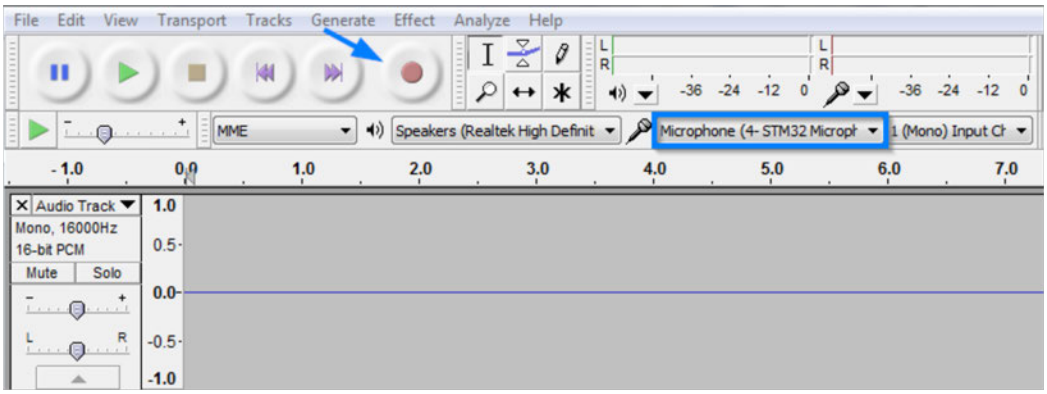
Step 4. Click the button to toggle the streaming status.

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.

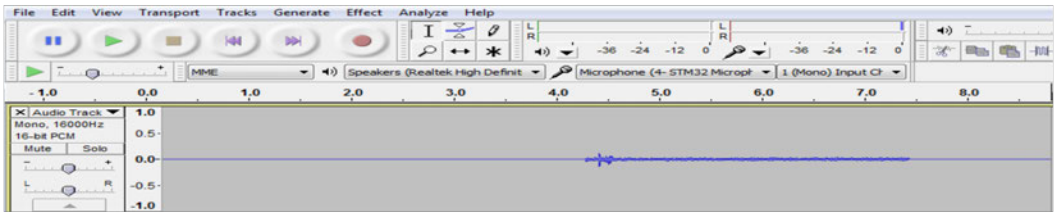
Figure 31. Audacity silent recording from Peripheral unit USB stream



Step 3. Click central unit user button 1

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.

3.3.4 Full-Duplex demo setup for STM32WB and a mobile device

The BlueVoiceOPUS profile is compatible with the STBLESensor app (version 4.9.0 or higher) available on Android/iOS stores.

A full-duplex audio stream can be created between an STM32WB board and an Android/iOS device. In this scenario, the ST module acts as the peripheral node, whereas the mobile device is the central node.

The FP-AUD-BVLINKWB1 BVLPeripheral application can be used to stream the audio acquired at 16 kHz and compressed with Opus to the mobile device.

At the same time, the STM32WB can receive compressed audio at 16 kHz, decode and send it to a PC via USB.

The setup requires an ST module that acts as peripheral node composed of an X-NUCLEO-CCA02M2 connected to a P-NUCLEO-WB55 or an STM32WB5MM-DK, and a mobile device running ST BLE Sensor app.

Step 1. Prepare a mobile device running STBLESensor app version 4.9.0 or higher

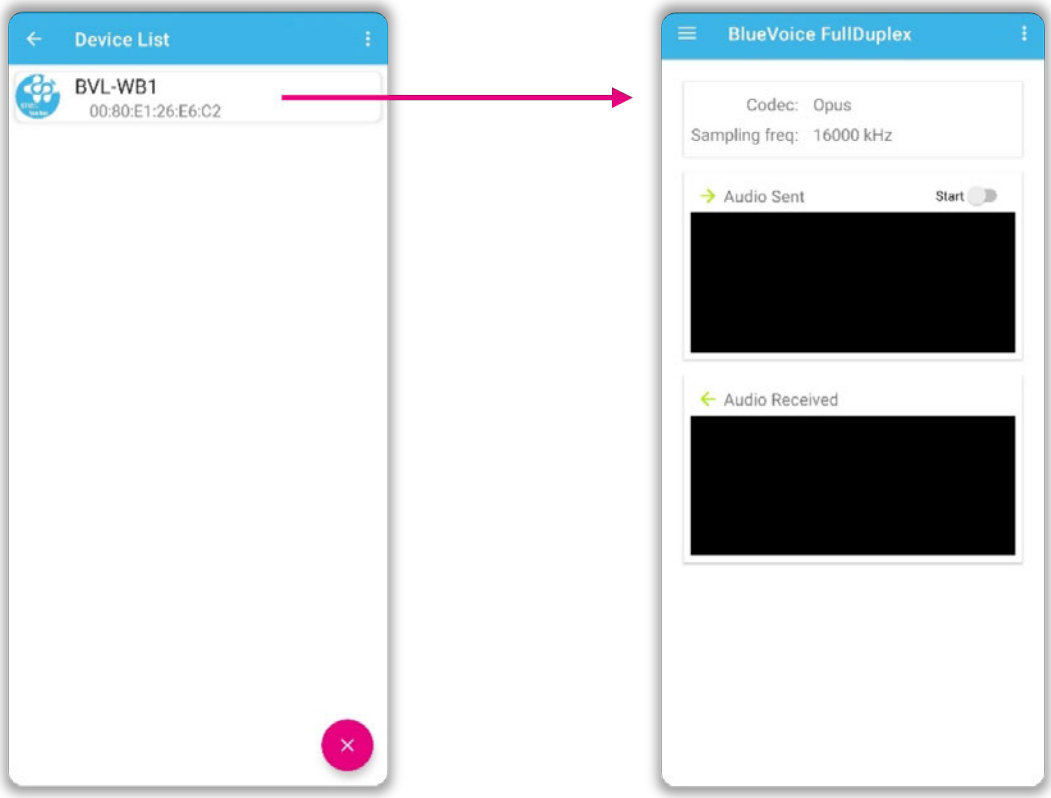
Step 2. Prepare the peripheral node by flashing BVLPeripheral firmware on a P-NUCLEO-WB55 or an STM32WB5MM-DK, make sure that is recognized by Windows as a standard microphone and set up Audacity, or another audio recording software, as explained in Section 3.3.3 .

Step 3. Check that JP1 for P-NUCLEO-WB55 or JP2 for STM32WB5MM-DK are on USB MCU. Then, connect the USB_USER connector of the peripheral node to the PC.

The STM32 Audio Streaming device appears in Windows Device Manager, the peripheral node is in advertising mode and the green LED blinks.

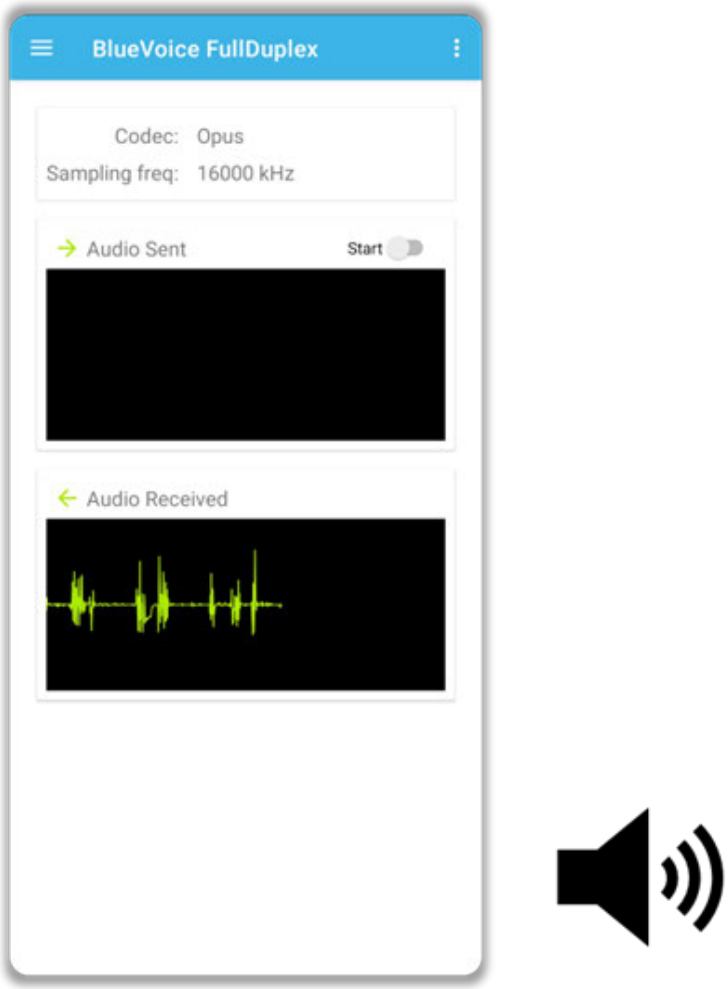
Step 4. Use ST BLE Sensor application to detect the peripheral node (BVL-WB1) and connect to it. The BlueVoice FulDuplex page is shown and the blue LED on the board starts blinking slow.

Figure 33. BVL-WB1 connection



Step 5. Push User button 1 on the board to start audio streaming to the app.
You can hear audio from the smartphone speaker and the blue LED starts blinking faster.

Figure 34. Audio streaming from the board to ST BLE Sensor app

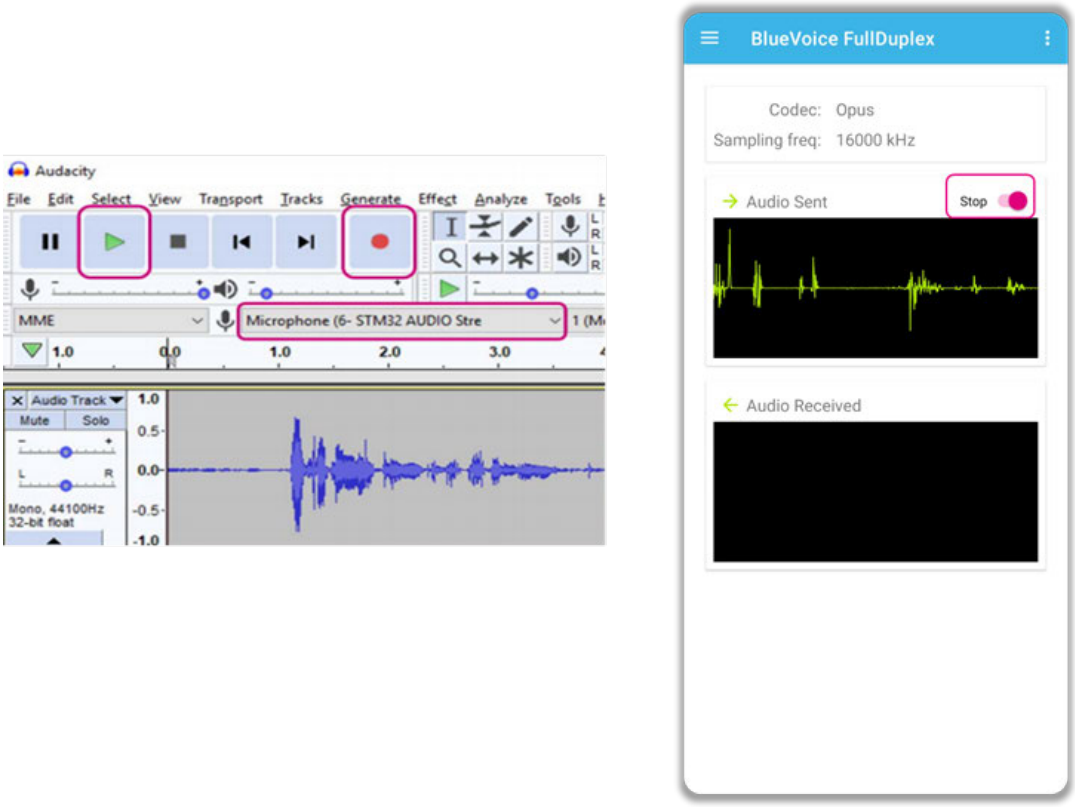


Step 6. Enable the [**Start**] switch on the app to start audio streaming from the smartphone to the board.

Step 7. Open Audacity on the PC, check if the audio input device is “STM32 AUDIO streaming” and then press the recording button.

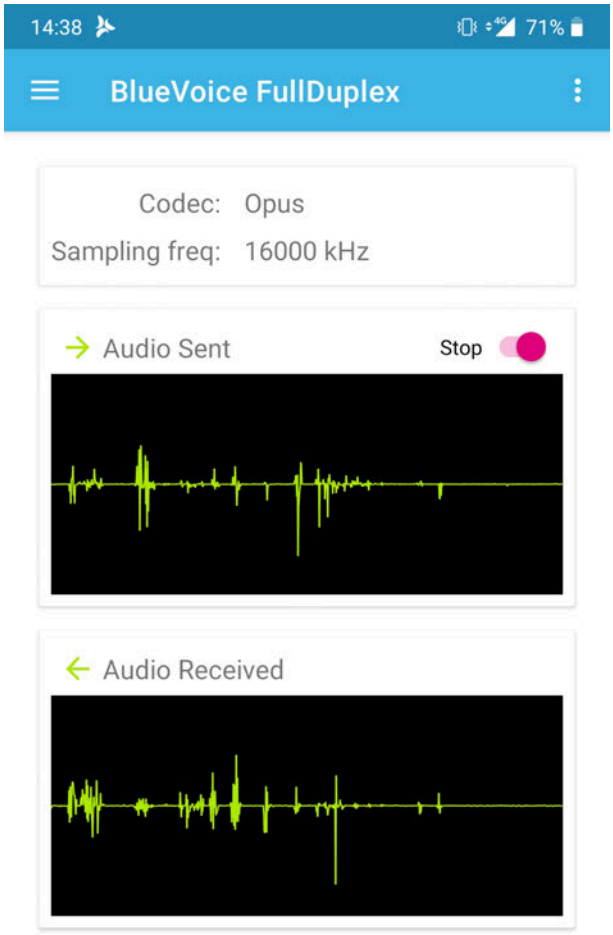
On the board, the blue LED remains steady.

Figure 35. Audio streaming from the ST BLE Sensor app to the board



Step 8. Enable the streaming on both sides to perform a full-duplex communication.
The blue LED on the board starts blinking fast.

Figure 36. Full-duplex streaming



Important:
To avoid Larsen effect, keep the board and smartphone at a minimum distance of 40 cm or connect an headset to your mobile device.

Note: If you are using the STM32WB5MM-DK, information about the demo are displayed on the LCD.

3.3.5 FP-AUD-BVLINKWB full-band demo setup

To set up the FP-AUD-BVLINKWB1 full-band application to demonstrate stereo music streaming over Bluetooth low energy from an Android device to STM32WB, you need two BLE devices interacting with each other and creating point-to-point wireless communication. One of the modules acts as central and the other as peripheral. In this application, the STM32WB is the peripheral while the mobile device is the central node.

Step 1. Prepare the peripheral node by flashing BVLPeripheral_FullBand firmware on a P-NUCLEO-WB55 or STM32WB5MM-DK.

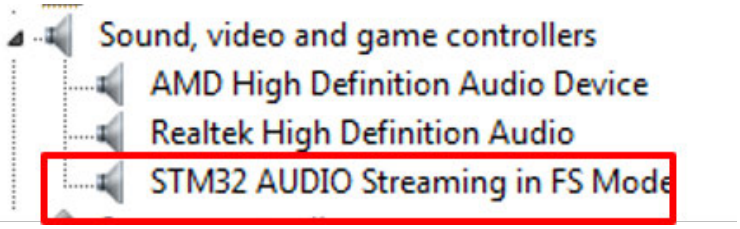
Step 2. Be sure to have installed an audio recording software such as Audacity on your PC.

Step 3. Prepare an Android device, that supports BLE 4.2, running STBLEsensor app.

Step 4. Move JP1 on P-NUCLEO-WB55 or JP2 on STM32WB5MM-DK, from USB STL to USB MCU.

Step 5. Connect the USB_USER connector of the peripheral node to the PC.
The STM32 Audio Streaming device appears in the Device Manager.

Figure 37. STM32WB recognized as audio streaming

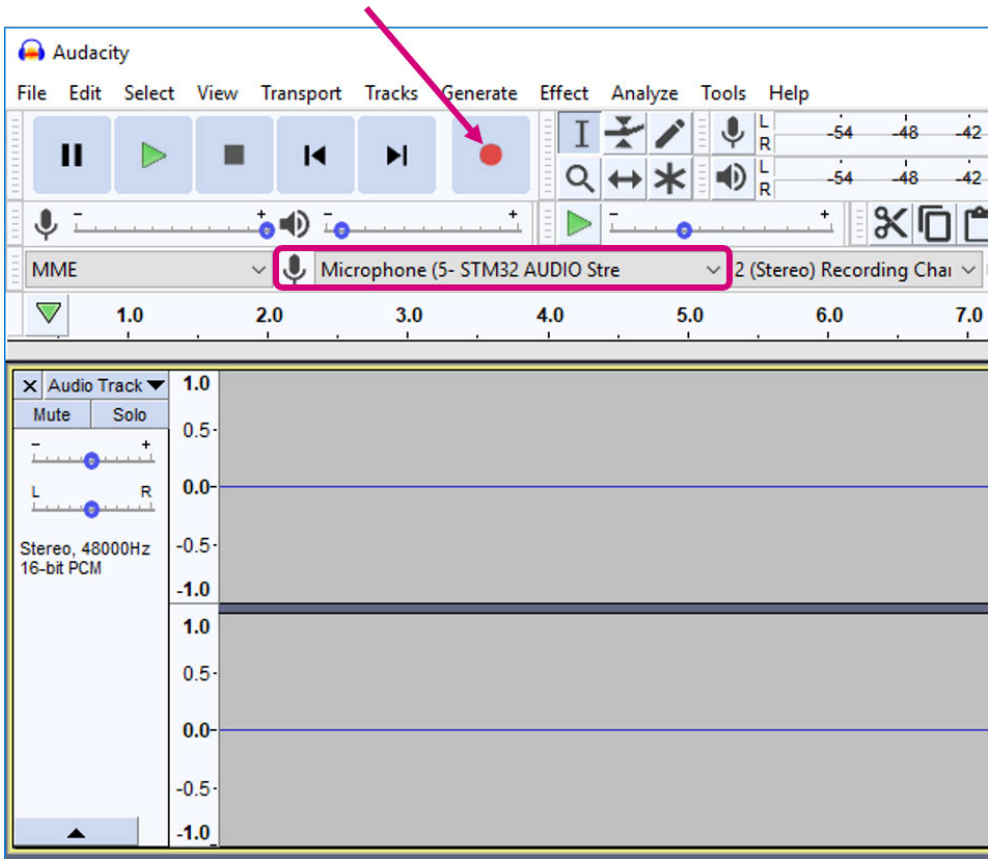


Step 6. Open Audacity.

Step 7. Select the STM32 microphone that appears in the input selector.

Step 8. Select 2 recording channels and press [**Record**] to start recording.

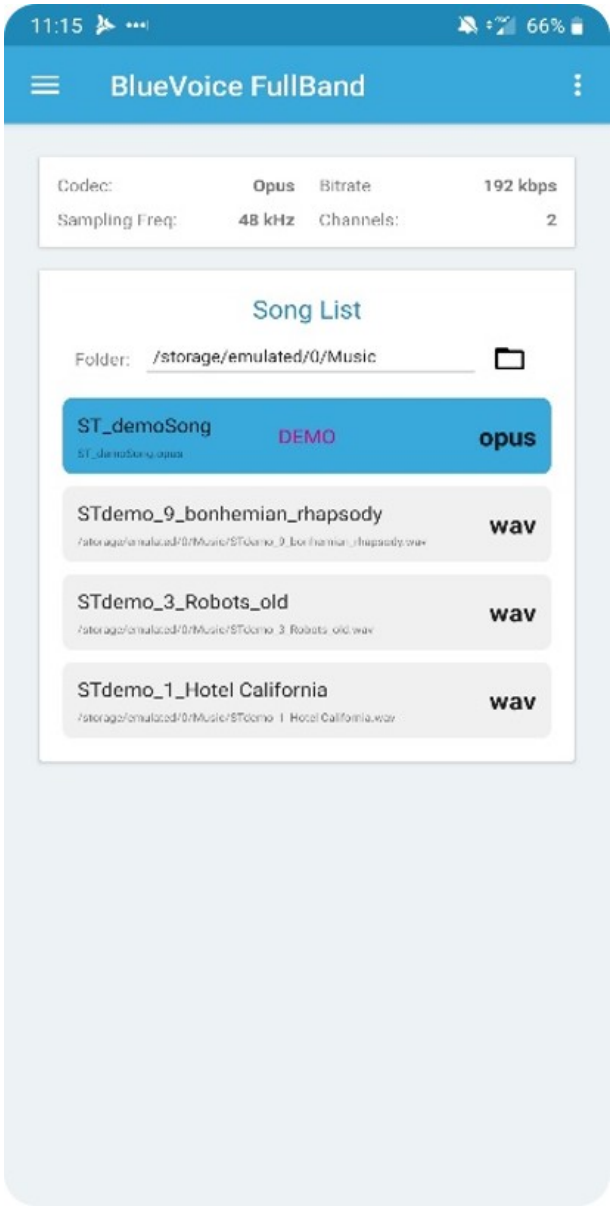
Figure 38. Audacity recording



Step 9. Open the STBLESensor app and connect to the BVFBAND from the device list.

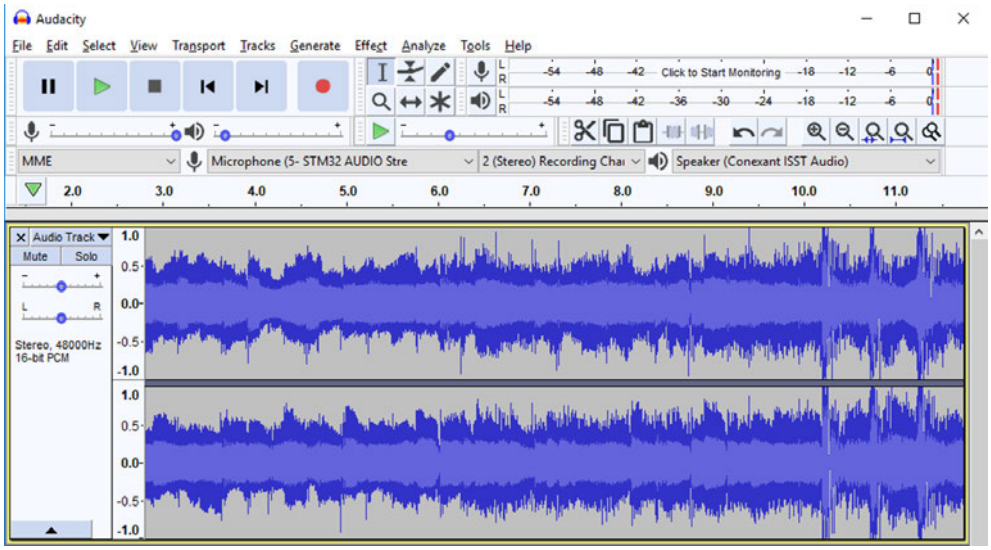
The BlueVoice FullBand page appears showing codec details and the song list. A demo song (ST_demoSong) is already available and can be streamed to the STM32WB by pressing the Play button; moreover, other songs (wave file, 2 channels, 48 kHz) could be load from the smartphone choosing the relevant folder.

Figure 39. BlueVoice FullBand page



You can now listen to high quality stereo music received via BLE on your PC.

Figure 40. **Audacity stereo music**



Appendix A References

1. Bluetooth Core Specification 5 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 https://opus-code.org/
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 3. **Document revision history**

Date	Version	Changes
28-Jun-2019	1	Initial release.
25-Nov-2019	2	Updated all content to reflect v.1.1.0 software release.
02-Mar-2021	3	Updated all content to reflect v.2.0.0 software release.

Contents

List of tables

List of figures