UM2196
User manual

Getting started with the FP-AUD-BVLINK1 STM32 ODE function pack based on half-duplex voice streaming over BLE

Introduction

FP-AUD-BVLINK1 is an STM32 ODE function pack that performs voice streaming over Bluetooth low energy in a half-duplex configuration. The application runs on the STM32 Nucleo and includes drivers and middleware for Bluetooth low energy (BlueNRG-MS) and MP34DT01-M or MP34DT04-C1 digital MEMS microphones.

The expansion is built on STM32Cube software technology to ease portability across different STM32 microcontrollers. The software comes with sample implementations of the drivers for X-NUCLEO-IDB05A1 plus X-NUCLEO-CCA02M1, when connected to a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L053R8 board.

FP-AUD-BVLINK1 is also compatible with SensorTile (STEVAL-STLKT01V1) and BlueCoin (STEVAL-BCNKT01V1).

Information regarding STM32Cube is available on www.st.com at: *http://www.st.com/stm32cube*.

Contents

List of tables

List of figures

1 Acronyms and abbreviations

Table 1: Acronyms and abbreviations

Term	Description
MEMS	Micro electro-mechanical systems
PDM	Pulse density modulation
PCM	Pulse code modulation
ADPCM	Adaptive differential pulse code modulation
USB	Universal serial bus
BSP	Board support package
HAL	Hardware abstraction layer
BLE	Bluetooth low energy
IDE	Integrated development environment
UUID	Universally unique identifier
ATT	Attribute protocol
GATT	Generic attribute profile
GAP	Generic access profile

2 FP-AUD-BVLINK1 software description

2.1 Overview

The key features of the FP-AUD-BVLINK1 package are:

- BlueVoiceADPCM, half-duplex voice-over-Bluetooth low energy communication profile
- Very low power Bluetooth low energy single-mode network processor, compliant with Bluetooth specification 4.1
- Complete middleware to build applications using digital MEMS microphones (MP34DT01-M or MP34DT04-C1)
- Digital audio signal acquisition and processing
- Audio input class USB driver to allow a device to be recognized as a standard USB microphone
- Easy portability across different MCU families thanks to STM32Cube
- Free, user-friendly license terms
- Sample implementation available for board X-NUCLEO-IDB05A1 plus X-NUCLEO-CCA02M1 connected to a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L053R8 board
- Sample implementation available for SensorTile (STEVAL-STLKT01V1) and BlueCoin (STEVAL-BCNKT01V1)
- Compatibility with ST BlueMS app (v3.0.0 or higher), available for Android™ and iOS™

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, MP34DT01-M MEMS microphone expansion boards, SensorTile and BlueCoin; middleware components for audio acquisition, communication with other Bluetooth LE devices, USB streaming of recorded signals and a dedicated profile for half-duplex speech transmission over BLE (BlueVoiceADPCM library).

The BlueVoiceADPCM profile defines a BLE service including one characteristic for audio transmission and one for synchronization. In a half-duplex system, both sides of the communication (central and peripheral) can act as servers of information. Periodic notifications containing compressed audio data are sent from one server to one client depending on the selected channel: central to peripheral or peripheral to central. BlueVoiceADPCM middleware is responsible for audio encoding and periodic data transmission on the server side and for decoding of received voice data on the client side.

BlueNRG-MS is a very low power Bluetooth low energy (BLE) single-mode network processor, compliant with Bluetooth specification core 4.1.

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 acquisition, compression and transmission over Bluetooth low energy of voice data from the module acting as the transmitter to the one acting as the receiver. The receiver is responsible for audio decompression and USB streaming of audio data to a PC. Any freeware or commercial audio recording software can be used to interface with the system.

The Peripheral module can also stream audio to an Android™ or iOS™ device running the ST BlueMS app v3.0.0 or higher.

2.2 Architecture

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

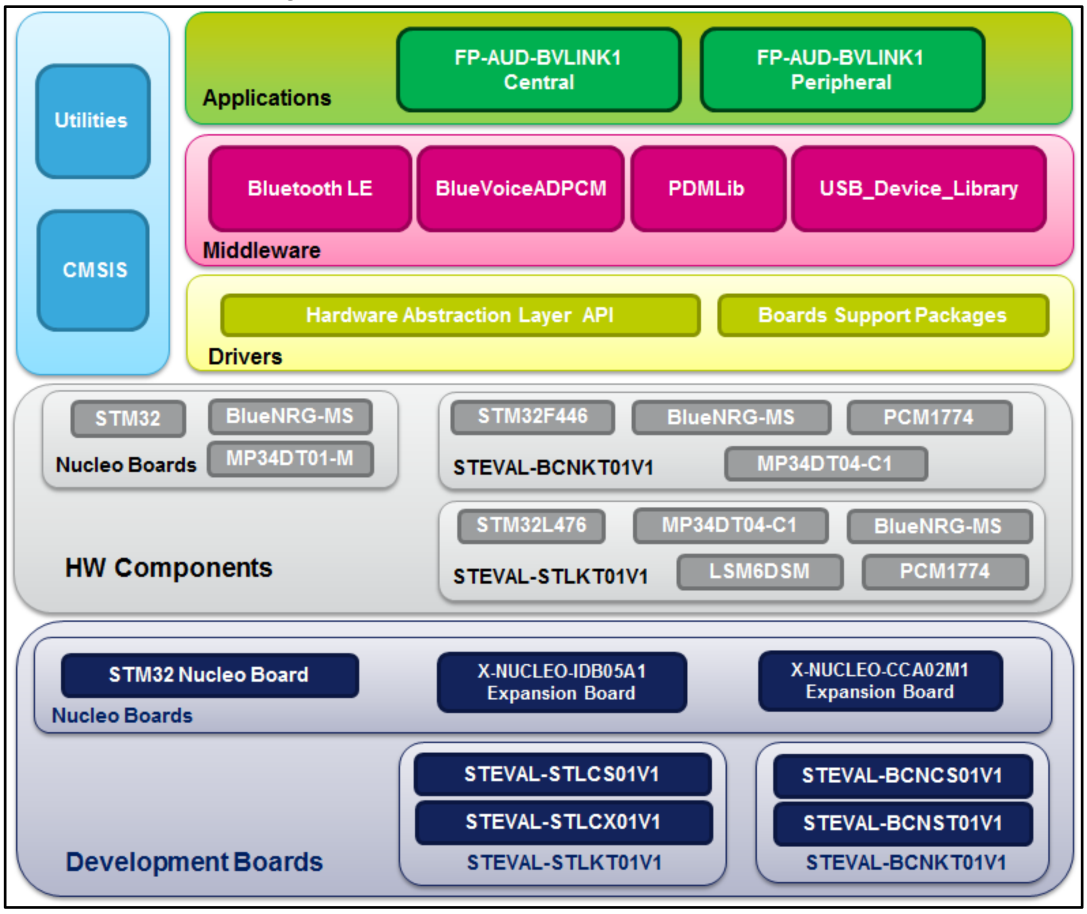
The FP-AUD-BVLINK1 provides a board support package (BSP) for the Bluetooth low energy (X-NUCLEO-IDB05A1) and the digital MEMS microphone (X-NUCLEO-CCA02M1) expansion boards. It includes also BSP for SensorTile (STEVAL-STLKT01V1) and BlueCoin (STEVAL-BCNKT01V1) for a complete, platform independent compatibility.

The BSP comes together with a set of middleware components for audio acquisition, voice transmission over BLE and USB audio in class implementation.

The application works with the following software layers:

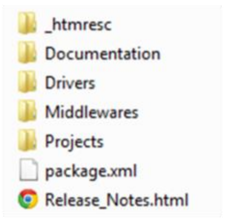
- **STM32Cube HAL layer**: provides a simple, generic and multi-instance of APIs (application programming interfaces) to interact with the upper application, library and stack layers. These generic and extension APIs are built on a common architecture and allow layers built on top of them (like middleware) 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 Nucleo board, apart from the MCU. This limited set of APIs provides a programming interface for board-specific peripherals like LED and user button; 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 enables also audio streaming exploiting an audio DAC, mounted on the SensorTile motherboard (STEVAL-STLCX01V1) or on the Coin Station (STEVAL-BCNST01V1)

Figure 1: FP-AUD-BVLINK1 software architecture



2.3 Folder structure

Figure 2: FP-AUD-BVLINK1 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, BlueNRG Bluetooth low energy network module and the BlueVoiceADPCM library.
- **Projects**: contains central and peripheral applications to demonstrate voice transmission over BLE. The projects are supplied for the NUCLEO-F401RE,

NUCLEO-L476RG, NUCLEO-L053R8, 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 BlueVoice profile description

2.5.1 Bluetooth low energy

2.5.1.1 Generic access profile (GAP)

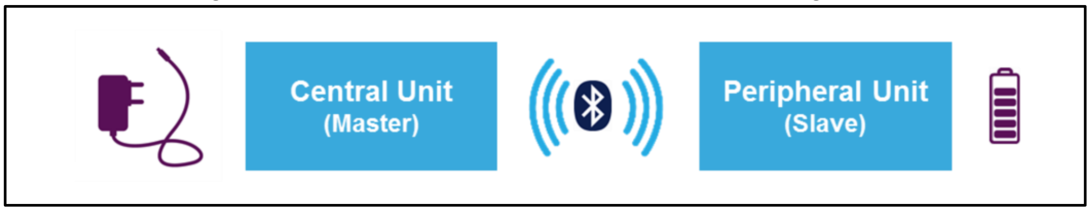
Bluetooth low energy 4.0 communication can be either broadcast or connection-based.

The BlueVoice 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 3: BlueVoice 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.

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 half-duplex communication scheme, BlueVoice role assignment is therefore decoupled via "transmitter" or "receiver" functionality.

2.5.1.2 Generic attribute profile (GATT)

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

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

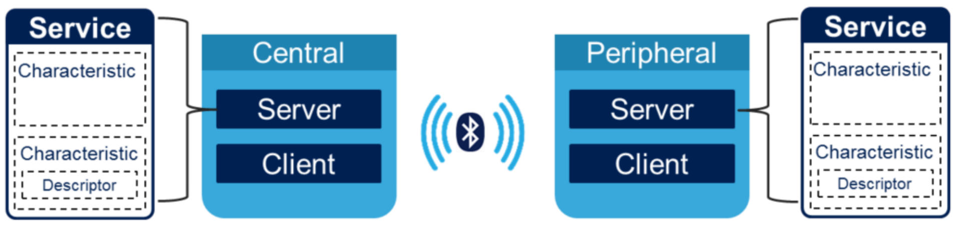
GATT defines client and server roles for interacting devices independent of the GAP master/central and slave/peripheral roles:

- **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 4: BlueVoice Profile GATT role assignment in a bidirectional system

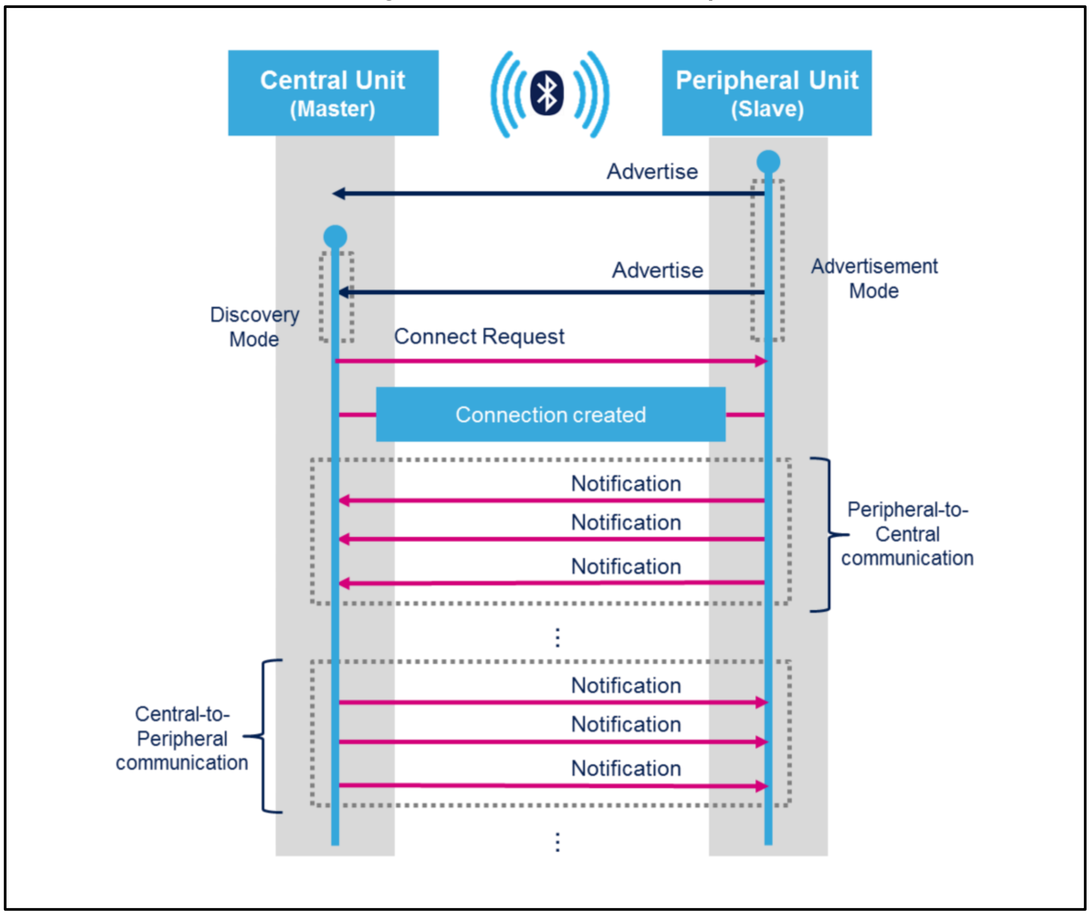


In both directions, streaming audio data transmission is based on periodic server-to-client notifications which do not require a request or response from the receiving device. Server-initiated updates are sent as asynchronous notification packets which include the handle of a characteristic value attribute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

2.5.1.3 Establishing a BLE connection

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 a slave device. After connection, notifications carrying audio data are periodically sent from the server to the client, according to the selected peripheral-to-central or central-to-peripheral direction.

Figure 5: BLE connection setup



2.5.1.4 BlueVoice 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 BlueVoice.
- **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 BlueVoice application defines a vendor-specific service named BlueVoice service which exposes a user voice to a client device.

Table 2: BlueVoice service definition

Type	Handle	UUID	Permissions	Value
Service	Att1	0x2800	READ	audio_adpcm_serv_uuid
Audio characteristic	Att1	0x2803	READ	NOT|0x0012|Audio UUID
	Att2	audio_adpcm_char_uuid	NONE	Audio data
	Att3	0x2902	READ/WRITE	Client characteristic configuration
Sync characteristic	Att1	0x2803	READ	NOT|0x0015|Sync UUID
	Att2	audio_adpcm_synch_char_uuid	NONE	Sync data
	Att3	0x2902	READ/WRITE	Client characteristic configuration

From the table, the BlueVoice service is described by the following attributes:

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

The BlueVoice service is composed of an Audio characteristic to expose actual compressed audio data and a Sync characteristic to expose collateral information used to implement a synchronization mechanism.

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: 08000000-0001-11e1-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

Sync Characteristic

- Att1 contains the synchronization characteristic declaration. Its attribute fields are the following:

  - 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 Sync-Peripheral Data (UUID: 40000000-0001-11e1-ac36-0002a5d5c51b).
- Att2 contains the characteristic value, in this case sync data. Its attribute fields are the following:
  - UUID: the same UUID present in the last 16 bytes of the characteristic definition's attribute value.
  - Permissions: None.
  - Value: the actual sync 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: BlueVoice UUID summary table

UUID name	UUID
audio_adpcm_serv_uuid	00000000-0001-11e1-9ab4-0002a5d5c51b
audio_adpcm_char_uuid	08000000-0001-11e1-ac36-0002a5d5c51b
audio_adpcm_sync_char_uuid	40000000-0001-11e1-ac36-0002a5d5c51b

Given the hierarchical architecture of BLE services, further characteristics may be added to the BlueVoice service in future releases of the BlueVoice application, such as allowing a client to configure parameters like volume, enabling/disabling of processing algorithms, etc...

The BlueVoice profile exported by the server exposes data type, format and access details to client devices.

2.5.2 Audio processing

The audio processing component of the BlueVoiceADPCM application is designed to achieve an audio sampling frequency of 8 or 16 kHz at the receiver side, with a trade-off between audio quality and bandwidth occupation for voice signals. Audio signals transmitted over the BLE link are compressed via ADPCM (adaptive differential pulse code modulation) to fit in the available data rate while minimizing radio transmission time and power consumption. In a half-duplex channel (two simplex communication channels in opposite directions), one node acts as the Tx module and the other acts as the Rx module.

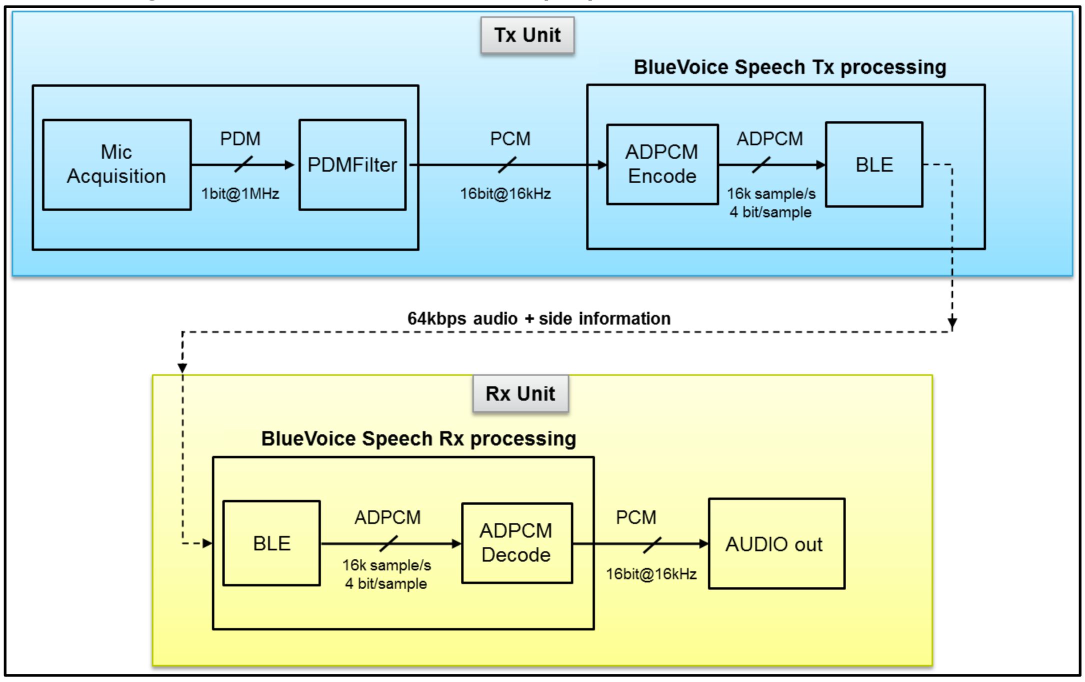
Figure 6: "FP-AUD-BVLINK1 chain1 – peripheral to central communication" shows the speech processing chain in a complete communication system with Tx and Rx.

On the Tx side, the library receives an audio signal which is typically acquired by a digital MEMS microphone as a 1-bit PDM signal at 1 MHz and converted by a PDM to PCM conversion filter into 16-bit PCM samples at 16 kHz.

The library can be provided with 1, 2, 5 or 10 ms of audio data. When the compressed output buffer is ready, a flag is set and audio data is streamed via BLE together with collateral ADPCM information.

The resulting communication bandwidth is 64 kbps (with 16 kHz audio sampling frequency) or 32 kbps (with 8 kHz audio sampling frequency) of audio data plus 300 bps of collateral information. In a half-duplex implementation, the same processing chain is implemented and the same bandwidth is used for communication in the opposite direction.

Figure 6: FP-AUD-BVLINK1 chain1 – peripheral to central communication



2.5.2.1 PDM filter

The digital MEMS microphone is designed to give a digital signal 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 quantization of noise density that is not constant 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.

In this application the conversion is performed exploiting a conversion library named PDMLib. It 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.

Further information can be found in the relevant application note available on www.st.com.

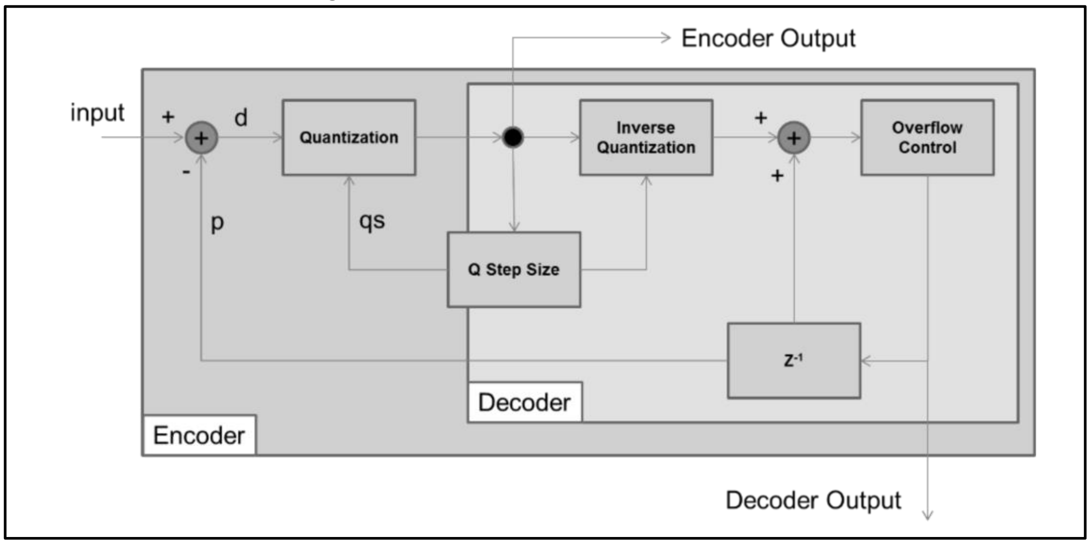
2.5.2.2 ADPCM compression

The ITU-T G.726 adaptive differential pulse code modulation (ADPCM) standard is applied to save bandwidth. This audio algorithm for loss waveform coding predicts the current signal value from previous values, and transmits the difference between the real and the predicted value, quantized with an adaptive quantization step.

The ADPCM algorithm used by BlueVoiceADPCM application compresses digital voice signals encoded as:

- Audio format: PCM
- Audio sample size: 16 bits
- Channels: 1 (mono)
- Audio sample rate: 16-8 kHz

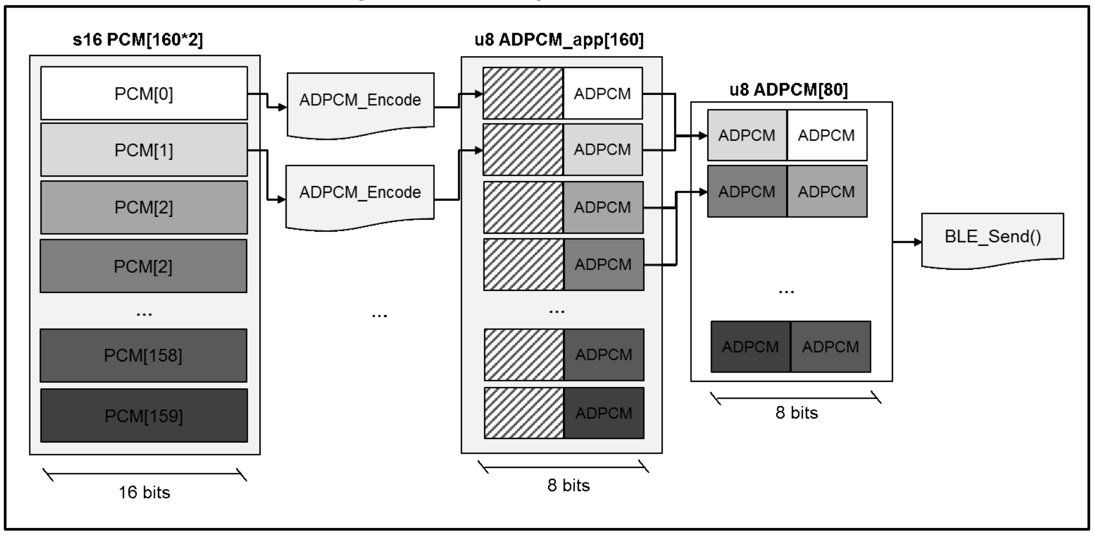
Figure 7: ADPCM encode-decode schema



BlueVoiceADPCM implements a modified version of the compression algorithm with improved communication robustness through an additional low data rate channel with collateral information is added to the ADPCM quantized values, slightly increasing the overall bit-rate to an average 64.3 Kbps.

The internal buffering required by ADPCM compression is shown in *Figure 8: "ADPCM packet mechanism"*. 16-bit input PCM samples are encoded in 8-bit temporary samples with 4-bit actual data (u8 ADPCM_app buffer) and then encapsulated in 8-bit samples containing information of two PCM samples (u8 ADPCM buffer).

Figure 8: ADPCM packet mechanism



ADCPM decoding on the Rx side is performed in a symmetric manner.

2.5.3 Data packets

The Tx data rate for streaming data is obtained from:

- the connection interval
- the number of packets per connection interval and user data payload for each packet

The FP-AUD-BVLINK1 application implements:

- a constant bitrate allocated to audio data through the chosen ADPCM compression
- a small connection interval to minimize the overall audio latency

The following connection intervals are set for Android™ and iOS™ compliance:

- conn_min_int=10 ms
- conn_max_int=21.25 ms

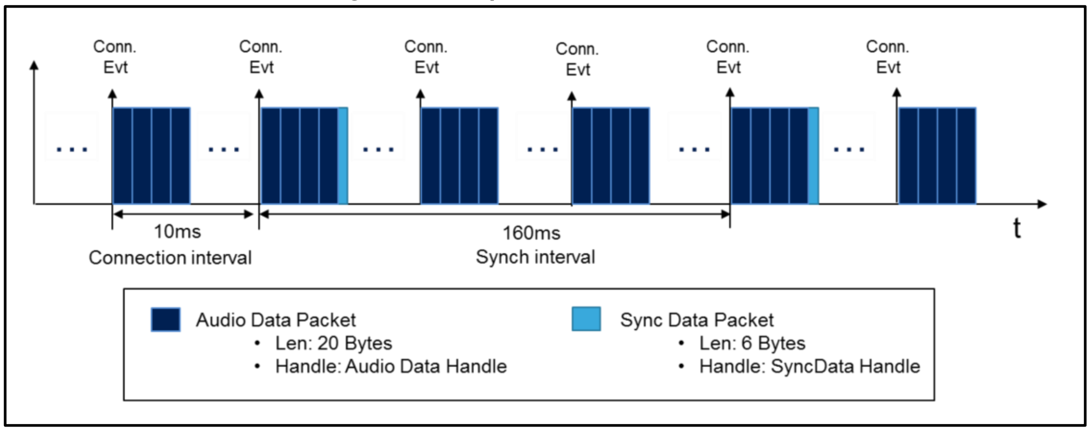
If the streaming is performed between two STM32 Nucleo modules, the selected connection interval is the minimum value (10 ms). The 64 kbps constant data rate target is achieved by sending 80 bytes of ADPCM data (640 bits) at each connection event.

With maxPayload = 20 bytes per packet (160 bit), four4 packets (two if an audio sampling frequency of 8 kHz is set) of 20 bytes are sent per average connection event.

In addition, ADPCM collateral information is sent at a lower frequency via an additional smaller packet sent at regular intervals.

The following figure shows a 16 kHz compressed audio streaming example, where four data packets of 20 bytes are sent at each connection interval of 10 ms, while ADPCM collateral information is sent as an additional packet once every 160 ms.

Figure 9: BLE packets for 16 Hz audio



2.6 BlueVoiceADPCM library software description

2.6.1 Overview

BlueVoiceADPCM implements a vendor-specific profile that expands the standard Bluetooth low energy profiles already supported by BlueNRG-MS.

The BlueVoiceADPCM engine is provided as a library which allows derivative firmware images to run on STM32 Nucleo only.

The CRC (cyclic redundancy check) calculation unit is used to unlock the library. Therefore it is mandatory to enable the CRC unit before initializing the BlueVoiceADPCM library.

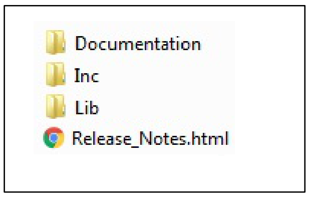
The library is implemented as middleware ready to be integrated in projects based on STM32Cube and Bluetooth low energy BlueNRG-MS network module.

Specifically, BlueVoiceADPCM requires the STM32 Bluetooth low energy middleware component included in the X-CUBE-BLE1 software expansion package.

2.6.2 Folder structure

The folder structure of the package STM32_BlueVoiceADPCM_Library in the Middlewares folder is shown below.

Figure 10: BlueVoiceADPCM library folder structure



The following files and folders are included in the software package:

- **Documentation**: with a compiled HTML file generated from the source code detailing the software components and APIs.
- **Inc**: contains the header file of the BlueVoiceADPCM library
- **Lib**: contains the BlueVoiceADPCM library binary code compiled for Cortex-M0, Cortex-M4 (with and without FPU activated), each one for the three supported tool-chains

2.6.3 Using the BlueVoiceADPCM library

2.6.3.1 Initialization and configuration

First call the `BluevoiceADPCM_Initialize` API to initialize the library and check if the CRC unit is enabled. If the return value is `BV_ADPCM_SUCCESS`, the library is ready to use, if the return value is `BV_ADPCM_LOCKER_ERROR`, the CRC unit must be enabled.

Afterwards the library must be configured according to the audio acquisition implemented in the application.

```
BV_ADPCM_Config_t BV_ADPCM_Config;
BV_ADPCM_Config.sampling_frequency = FR_16000;
BV_ADPCM_Config.channel_in = 1;
BV_ADPCM_Config.channel_tot = 2;
BluevoiceADPCM_SetConfig(&BV_ADPCM_Config);
```

A configuration structure must be filled in by setting the audio sampling frequency (FR_16000 or FR_8000), the total number of channels given as input for the library, and which channel (between 1 and channel_tot) shall be used for the voice streaming over BLE. If the return value is `BV_ADPCM_SUCCESS`, the library is configured correctly.

The BlueVoiceADPCM library can be reset by recalling `BluevoiceADPCM_SetConfig`.

The library includes three callbacks that must be called when the corresponding BlueNRG-MS event occurs:

- `BluevoiceADPCM_ConnectionComplete_CB` must be called when an EVT_LE_CONN_COMPLETE event occurs; it sets the connection handle.

- `BluevoiceADPCM_DisconnectionComplete_CB` must be called when an EVT_DISCONN_COMPLETE event occurs; it resets internal parameters.
- `BluevoiceADPCM_AttributeModified_CB` must be called when an EVT_BLUE_GATT_ATTRIBUTE_MODIFIED event occurs; when an enable notification is received, it sets the working mode accordingly.

A timeout has been included for BlueVoice profile status management. When the module is no longer streaming or receiving, the profile status switches to `BV_ADPCM_STATUS_READY` as soon as the timer expires. The duration (in ms) of this timeout is defined by the `BV_ADPCM_TIMEOUT_STATUS`. To increase this timer, call the `BluevoiceADPCM_IncTick` every 1 ms through the SysTick_Handler.

2.6.3.2 Working mode setting

The library working mode can be configured as NOT_READY (initial setting), TRANSMITTER, RECEIVER or HALF-DUPLEX.

The service and characteristics for the BlueVoice profile can be created by calling `BluevoiceADPCM_AddService` and `BluevoiceADPCM_AddChar` functions. Both APIs require the UUIDs (chosen by the user) as parameters to return the relevant handlers.

The handle of the service must also be passed to the `BluevoiceADPCM_AddChar` API.

Alternatively, BlueVoice characteristics can be added to a pre-existing service created in your own application by calling `BluevoiceADPCM_AddChar` and passing the handle of that particular service as a parameter.

If both functions return `BV_ADPCM_SUCCESS`, the library is set to TRANSMITTER mode and can stream audio over BLE.

You can also create the characteristics out of the library and pass the relevant handles using a type structure like `BV_ADPCM_ProfileHandle_t` or `BluevoiceADPCM_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 collateral information used to implement a synchronization mechanism:aci_gatt_add_char(ServiceHandle,
  UUID_TYPE_128, CharAudioSyncUUID, 6,
  CHAR_PROP_NOTIFY, ATTR_PERMISSION_NONE,
  GATT_DONT_NOTIFY_EVENTS, 16, 1,
  CharAudioSyncHandle);

If you choose not to create the BlueVoice service, the module is not able to transmit audio. This node can still function as a RECEIVER: however, if the connected module exports the BlueVoice profile; you must enable notifications on the other node by calling the `BluevoiceADPCM_EnableNotification` function and setting the handle of the BlueVoice service and characteristics exported by the transmitter module through the `BluevoiceADPCM_SetRxHandle` API.

If both the TRANSMITTER and RECEIVER procedures are performed, the module acts as both and the working mode is set to HALF-DUPLEX, creating a half-duplex link over BLE.

2.6.3.3 Audio signal injection

The BlueVoiceADPCM library receives audio PCM input samples. The `BluevoiceADPCM_AudioIn` function accepts parameters from a PCM buffer, containing all the acquired audio channels (according to the previous library configuration), and the number of PCM samples (for each channel) given as input. An amount of data equal to 1, 2, 5 or 10 ms is accepted, otherwise `BV_ADPCM_PCM_SAMPLES_ERR` is returned.

The library compresses received PCM input samples; when 10 ms of audio is compressed, the `BluevoiceADPCM_AudioIn` API returns `BV_ADPCM_OUT_BUF_READY`.

2.6.3.4 Compressed audio streaming

For the transmitter part, the BlueVoiceADPCM library gathers compressed data in an internal double buffer. For every 10 ms of audio, the `BluevoiceADPCM_AudioIn` function returns `BV_ADPCM_OUT_BUF_READY` to signal output data can be sent via BLE by calling the `BluevoiceADPCM_SendData` API.

If the audio sampling frequency is set to 8 kHz, two 20-byte packets are sent every 10 ms (according to the connection interval); if the frequency is 16 kHz, four 20-byte packets are sent.

On the receiver side, compressed audio is received after notification through an EVT_BLUE_GATT_NOTIFICATION event and passed to the `BluevoiceADPCM_ParseData` function that decompresses data and returns a PCM buffer. This API is used to parse both audio and collateral information data.

2.7 FP-AUD-BVLINK1 application description

FP-AUD-BVLINK1 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 half-duplex speech communication over BLE using the BlueVoice Profile specification, central and peripheral application projects are included in the package.

As part of a bidirectional communication system, both BlueVoice modules can act as the transmitter or receiver of voice communication, depending on the active channel:

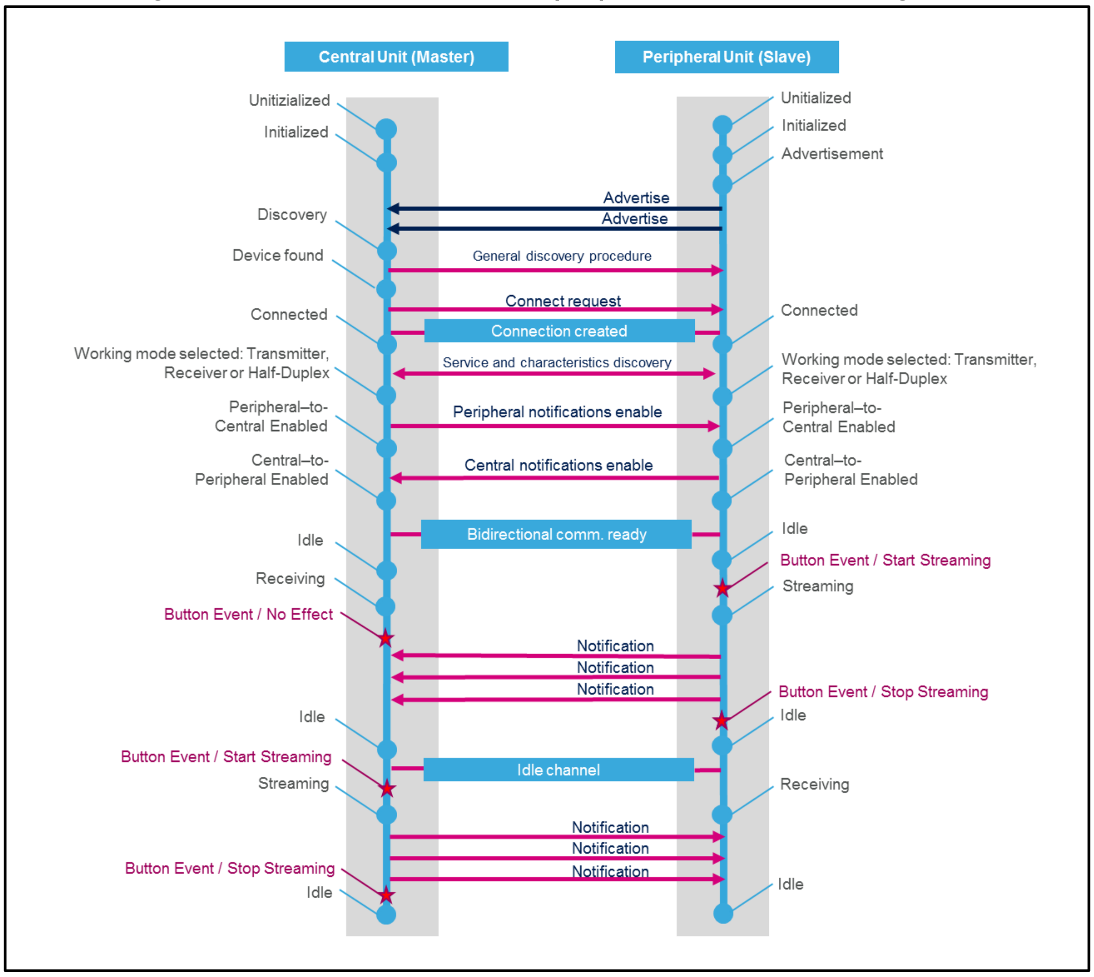
- when a module is streaming, the application handles audio acquisition, data compression and "packeting" of the ADPCM compressed audio to be streamed over BLE, according to the BlueVoice Profile specification.
- when a module is receiving, the application is responsible for the decompression of ADPCM audio data received via BLE and for USB or audio OUT streaming of decoded PCM samples.

After connection, the active communication channel is controlled by the STM32 Nucleo and BlueCoin user button, or by a double tap if a SensorTile is used: *Figure 11: "FP-AUD-BVLINK1: central-peripheral communication diagram"* shows how the communication link is set and how audio streaming is performed. Once connected, if the channel is idle, pressing the user button starts and stops streaming. If the channel is receiving, clicking on the user button has no effect.

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

- initialization: slow blinking
- connection: normal blinking
- streaming: rapid blinking
- reception: steady on (not blinking)

Figure 11: FP-AUD-BVLINK1: central-peripheral communication diagram


Central Unit (Master)
Peripheral Unit (Slave)
Unitizialized
Initialized
Discovery
Device found
Connected
Working mode selected: Transmitter, Receiver or Half-Duplex
Peripheral–to-Central Enabled
Central–to-Peripheral Enabled
Idle
Receiving
Button Event / No Effect
Idle
Button Event / Start Streaming
Streaming
Button Event / Stop Streaming
Idle
Unitialized
Initialized
Advertisement
Advertise
Advertise
General discovery procedure
Connect request
Connection created
Connected
Service and characteristics discovery
Working mode selected: Transmitter, Receiver or Half-Duplex
Peripheral notifications enable
Peripheral–to-Central Enabled
Central notifications enable
Central–to-Peripheral Enabled
Bidirectional comm. ready
Idle
Button Event / Start Streaming
Streaming
Notification
Notification
Notification
Button Event / Stop Streaming
Idle
Idle channel
Receiving
Notification
Notification
Notification
Idle

2.7.1 FP-AUD-BVLINK1 implementation

In this section, implementation of the FP-AUD-BVLINK1 application is briefly described.

Initialization

1. Audio acquisition initialization:
   - PDM to PCM middleware is initialized (not needed if an STM32L4 is used);
   - the required MCU peripherals are set up using the dedicated BSP function.
2. Audio IN USB driver initialization (not needed if a SensorTile or a BlueCoin is used):
   - the related descriptor is configured to configure the single channel at 16 kHz;
   - after the configuration phase, the device streams signals to a host device as a standard single channel USB microphone.
3. BlueNRG-MS component and BLE middleware initialization
4. BlueVoiceADPCM library initialization
5. BlueVoiceADPCM library initialization:
   - CRC peripheral enabled for library unlock;
   - BlueVoiceADPCM library configuration;
   - BlueVoiceADPCM service and characteristics set up.

Running

1. HCI_Process: BLE middleware engine running;

2. Application process
3. Status control: reads the BlueVoice Profile status:
   – Unitialized: BlueVoice Profile is not yet initialized.
   – Initialized: BlueVoice Profile has been initialized successfully.
   – Advertisement/Discovery: depending on central or peripheral role, BlueVoice Profile is in discovery or advertisement mode, respectively.
   – Connected: central and peripheral modules are connected but notification mechanism from the other module is not enabled yet.
   – Central to peripheral enabled: the peripheral device has enabled notifications from the central device.
   – Peripheral to central enabled: the central device has enabled notifications from the peripheral device.
   – Half Duplex Ready: the bidirectional channel is ready and idle.
   – Streaming: the module is streaming audio data to the other module.
   – Receiving: the module is receiving audio data from the other module.
   – BlueVoiceADPCM StateMachine: responsible for internal status update and data processing.

2.7.1.1 Streaming state

During the initialization phase, BSP functions are used to set up DMA for audio acquisition via I²S of the PDM signal of on-board digital MEMS microphones. In streaming mode, the acquisition engine is turned on: each 1 ms, the I²S DMA interrupt handler is responsible for the conversion of high frequency PDM data of one of the two microphones to a 16 kHz PCM audio stream via the PDM to PCM decimation library middleware.

If an STM32L4 is used, microphones can be connected via DFSDM: the PDM signal is acquired and converted to PCM in HW by the DFSDM peripheral itself.

The resulting 16 samples/ms are transmitted to the BlueVoiceADPCM library, which internally implements a double buffering mechanism to collect 10 ms of audio encoded in ADPCM and autonomously sent via BLE, depending on the module status.

2.7.1.2 Receiving state

In receiving state, compressed audio data are received in the form of periodic notifications from the other module. BlueNRG-MS stack functions are used to handle notification events.

GATT Notification callback is called for any new notification received from the other module.

Received data are transmitted to the BlueVoiceADPCM library which decodes the compressed audio and returns the number of audio samples, if any.

Decoded PCM samples are then sent to the USB driver, or to the audio OUT system via the relevant APIs.

3 System setup guide

3.1 Hardware description

The FP-AUD-BVLINK1 application is based on:

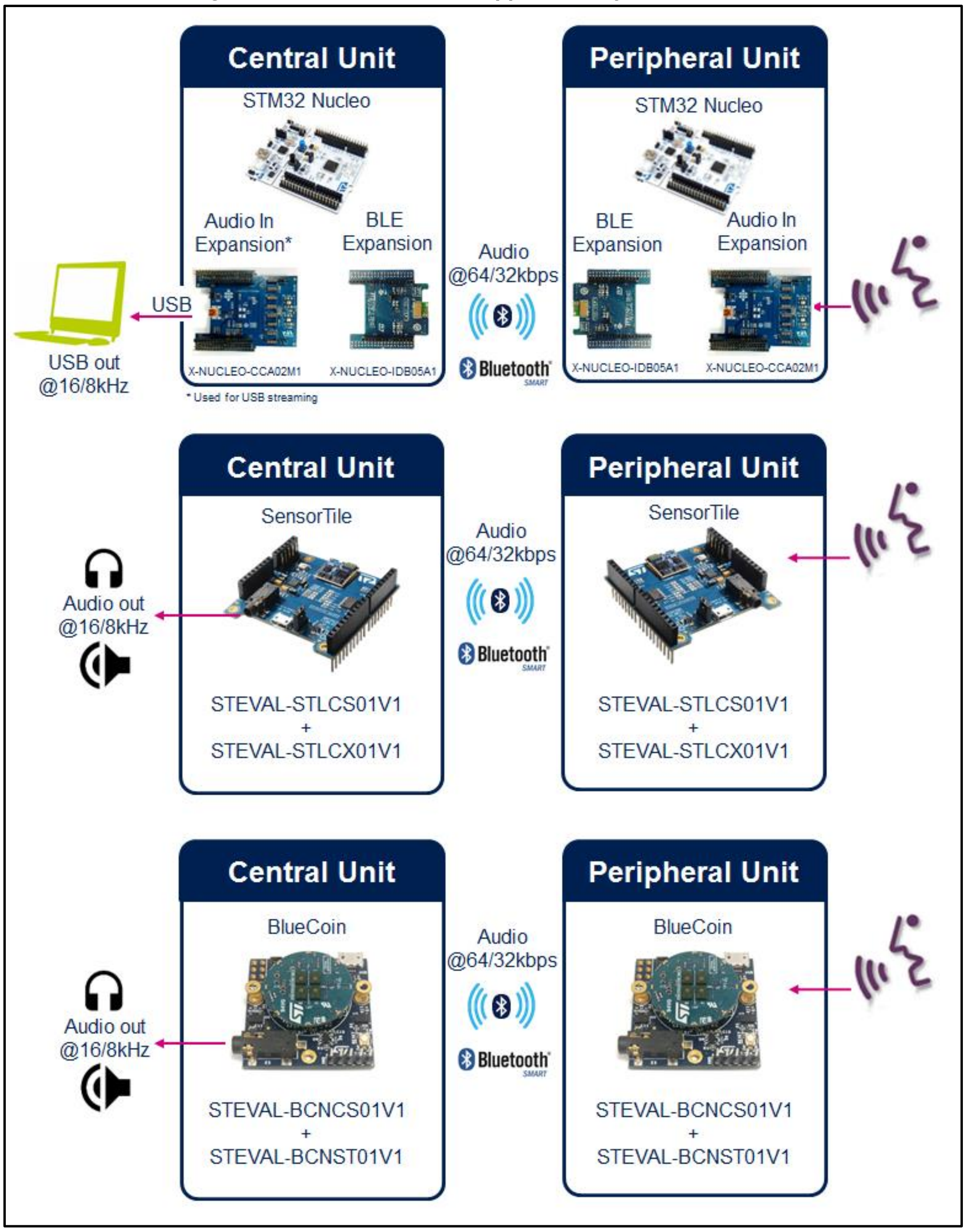
- STM32Nucleo: with ready-to-build projects available for NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L053R8, together with a BlueNRG-MS expansion board (X-NUCLEO-IDB05A1) and a microphone expansion board (X-NUCLEO-CCA02M1);
- SensorTile: STEVAL-STLCS01V1 mounted on the cradle expansion STEVAL-STLCX01V1 which 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.

Figure 12: "FP-AUD-BVLINK1 application system overview" shows the system overview. The peripheral unit acts as the transmitter whereas the central unit acts as the receiver of the audio stream.

The half-duplex demo can be setup by using a Peripheral and a Central node chosen among the different architectures described above. As Central unit an Android or iOS device running ST BlueMS app can be used.

A NUCLEO-L053R8 board can act only as peripheral.

Figure 12: FP-AUD-BVLINK1 application system overview



3.1.1 STM32 Nucleo platform

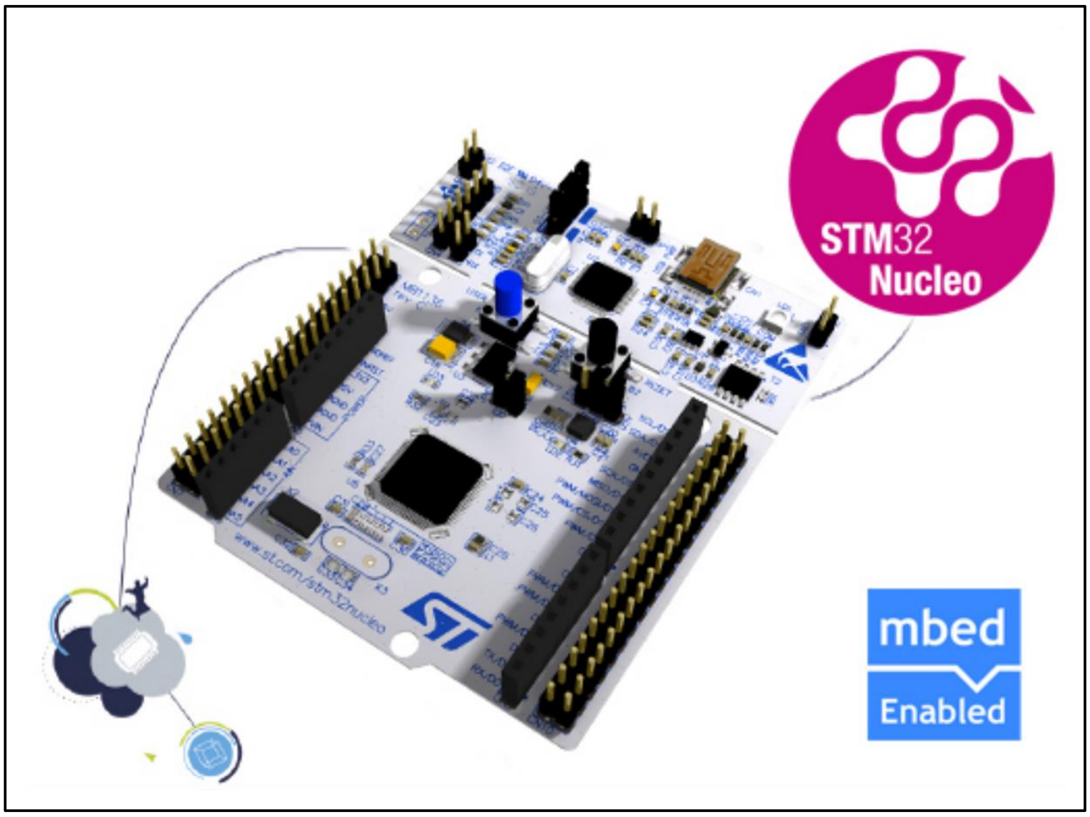
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

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 13: STM32 Nucleo board



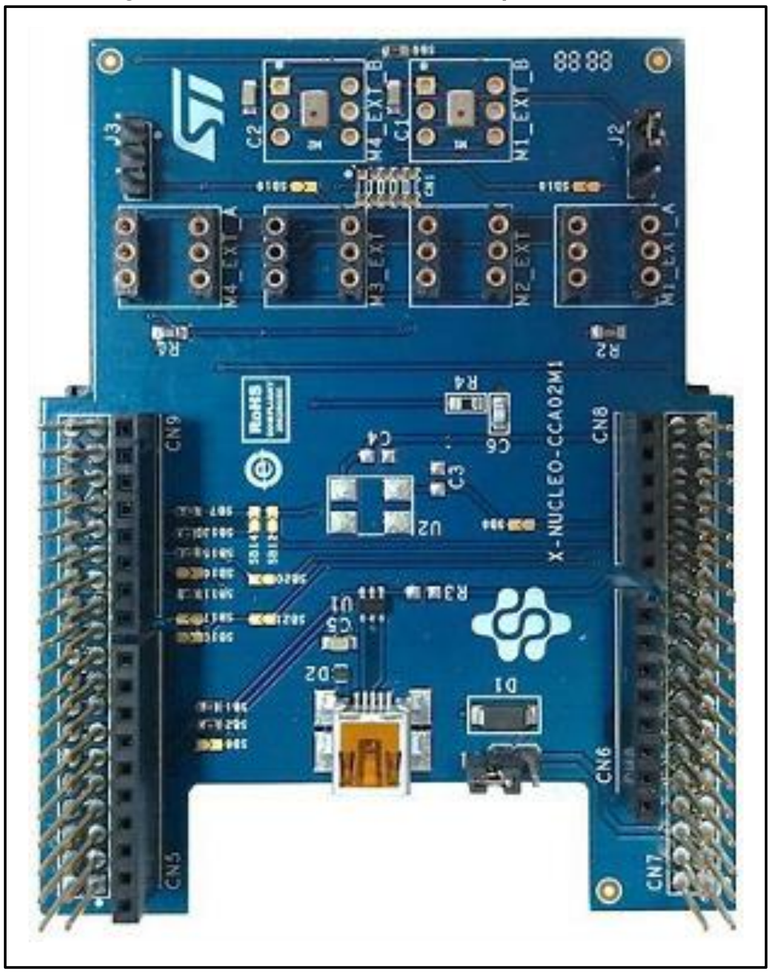
Information regarding the STM32 Nucleo board is available at *www.st.com/stm32nucleo*

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 I^{2}S bus and up to four coupon microphones using I^{2}S 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 14: X-NUCLEO-CCA02M1 expansion board

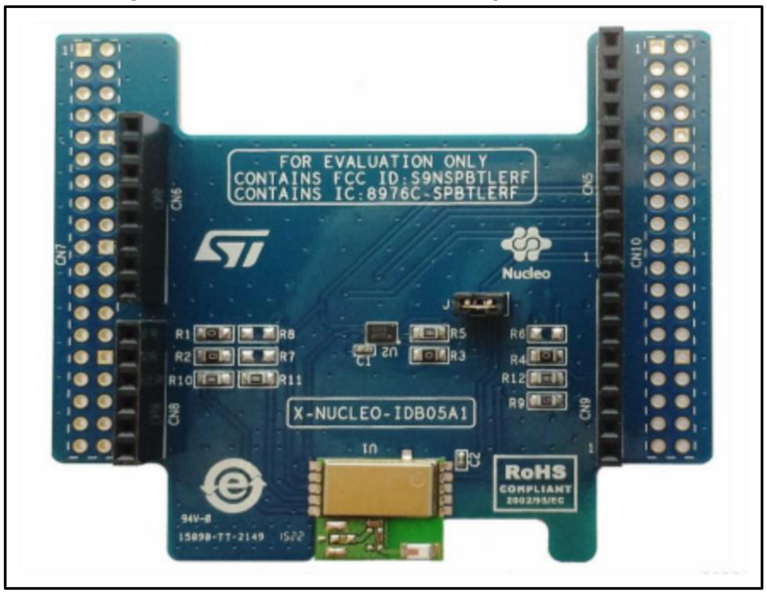


Information regarding the X-NUCLEO-CCA02M1 expansion board is available on www.st.com at *http://www.st.com/x-nucleo*.

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 15: 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 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 App.

The SensorTile is a tiny, square-shaped IoT module that packs powerful processing capabilities leveraging an 80 MHz STM32L476JGY microcontroller and Bluetooth low energy connectivity based on BlueNRG 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.

3.1.4.2 Features

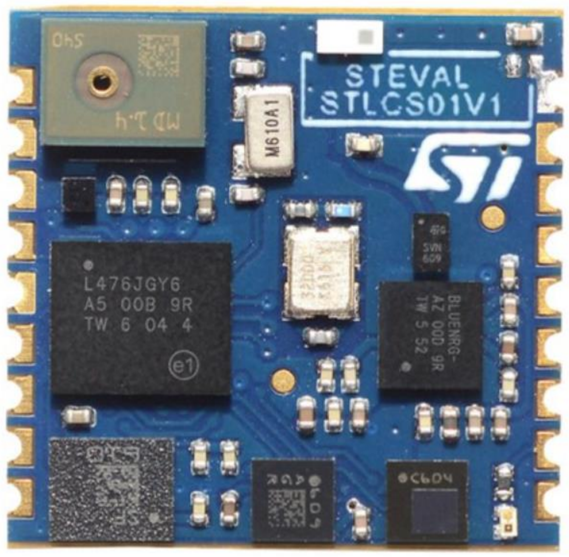
- FCC (ID: S9NSTILE01) certified
- IC (IC: 8976C-STILE01) certified with PMN: STEVAL-STLKT01V1; HVIN: STEVAL-STLCS01V1; HMN: STEVAL-STLCX01V1; FVIN: bluenrg_7_1_e_Mode_2-32MHz-XO32K_4M.img
- Included in the development kit package:
  - SensorTile module (STEVAL-STLCS01V1) with STM32L476, LSM6DSM, LSM303AGR, 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 for housing the SensorTile cradle and the battery
- 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. It includes low level drivers for all the on-board devices
  - BLUEMICROSYSTEM1 and BLUEMICROSYSTEM2: STM32Cube expansion software package, supporting different algorithms tailored to the on-board sensors
  - FP-SNS-ALLMEMS1 and FP-SNS-MOTENV1: STM32 ODE functional packs
  - ST BlueMS: iOS and Android demo Apps
  - BlueST-SDK: iOS and Android Software Development Kit
  - Compatible with STM32 ecosystem through STM32Cube support

3.1.4.3 Boards included in the kit

Figure 16: 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 BLUEMICROSYSTEM1 and BLUEMICROSYSTEM2 software expansion package for STM32Cube and the STM32 ODE functional pack FP-SNS-ALLMEMS1
- 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.

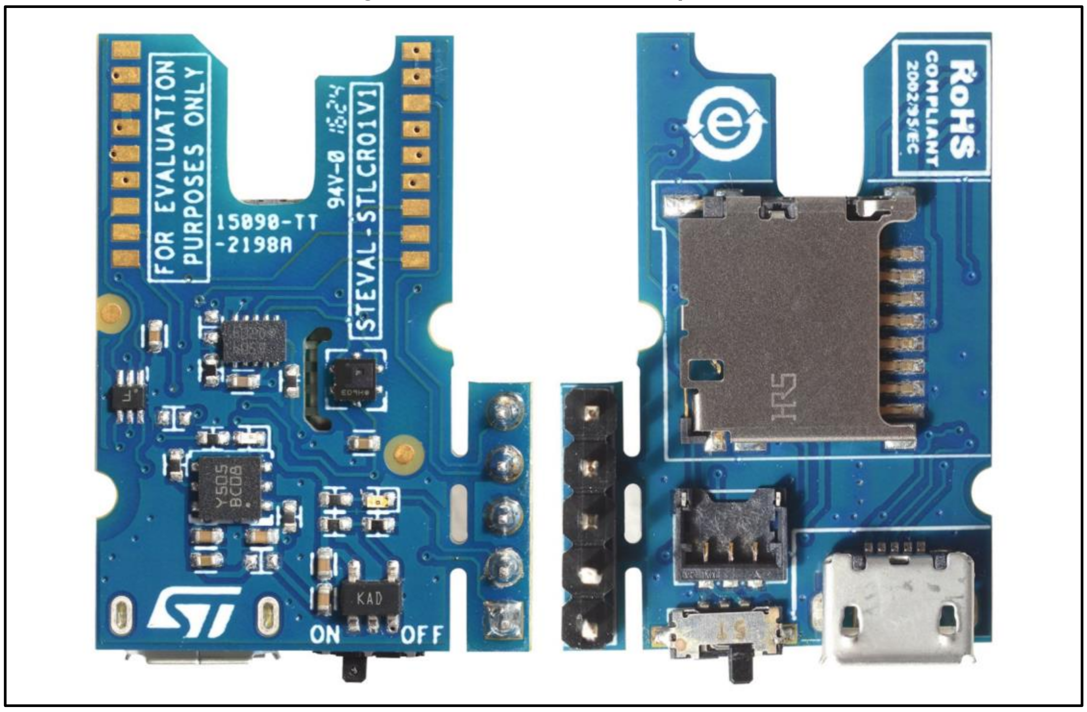
The module comes with pre-loaded BLUEMICROSYSTEM2 software that initializes all the sensors and the Bluetooth low energy radio. The “ST BlueMS” app, available free of charge on Apple Store™ and Google Play™, is the easiest and fastest way to start using the SensorTile board and to experience a real activity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

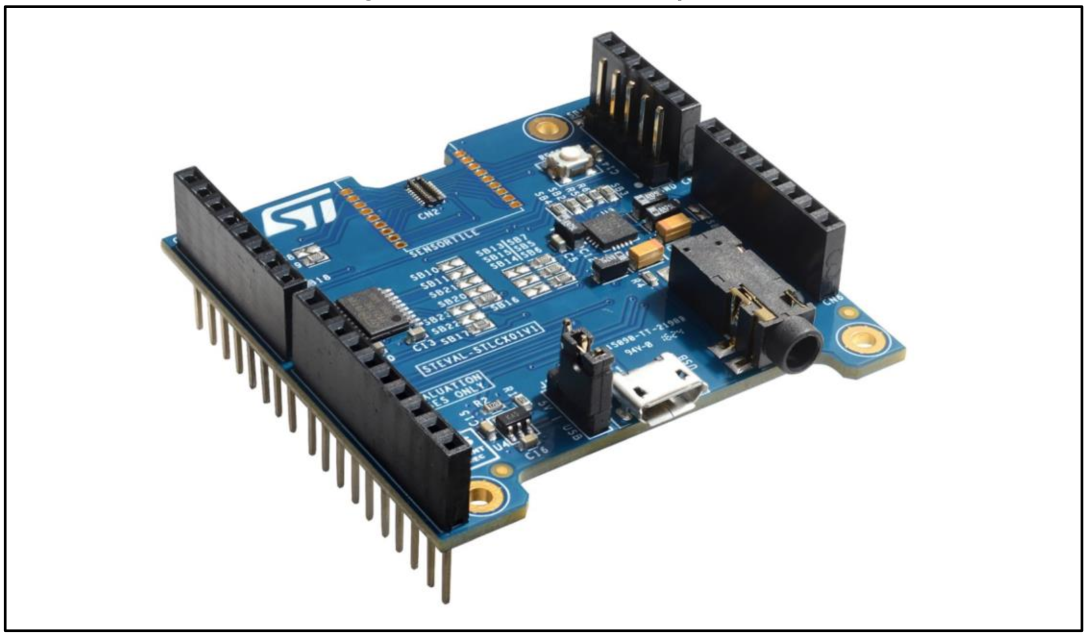
Figure 17: STLCR01V1 board photo



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 18: 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 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.

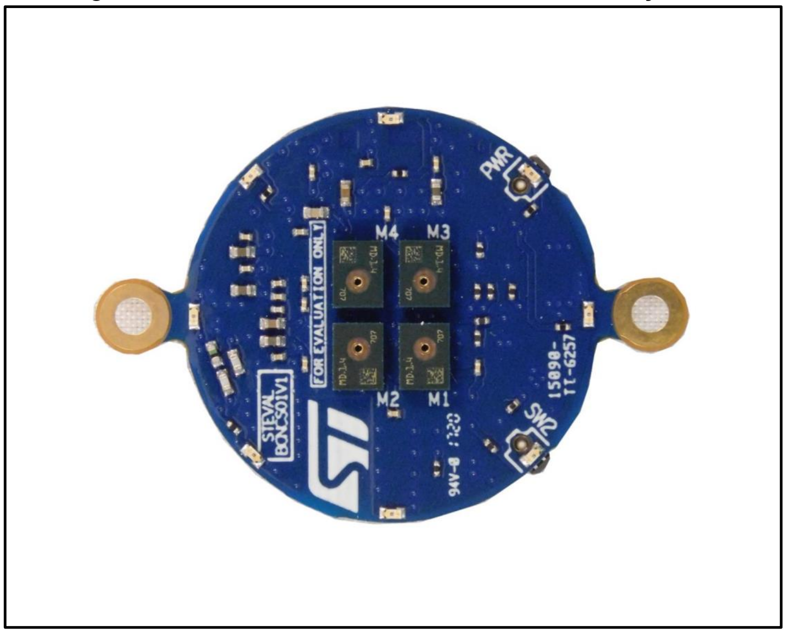
3.1.5.2 Features

- 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 equipped with 2 x VL53L0X time-of-flight, LDK120M LDO, audio DAC, 3.5 mm audio jack, USB connector, battery connector and SWD programming interface
  - BlueCoin Cradle (STEVAL-BCNCR01V1) with SD memory card slot, USB port, ST1S12XX step down DC-DC converter and battery connector
  - 130 mAh Li-Po battery
  - Plastic box for housing the BlueCoin cradle and the battery
  - SWD programming cable
- Software libraries and tools:
  - STSW-BCNKT01 BlueCoin firmware package with raw sensor data streaming support via USB, data logging on SDCard, audio acquisition and audio streaming, time-of-flight example and BLE protocol to interface to a Smartphone App. It includes low level drivers for all the on-board devices
  - 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
  - ST 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 Starter kit contents

Figure 19: STEVAL-BCNCS01V1 - BlueCoin Core System



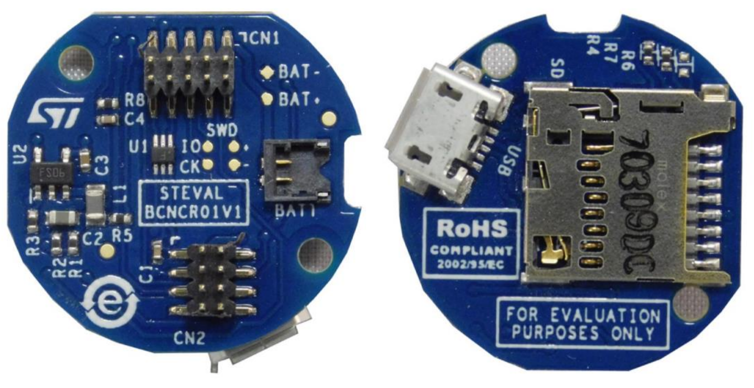
STEVAL-BCNCS01V1 - BlueCoin Core System board features

- 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.

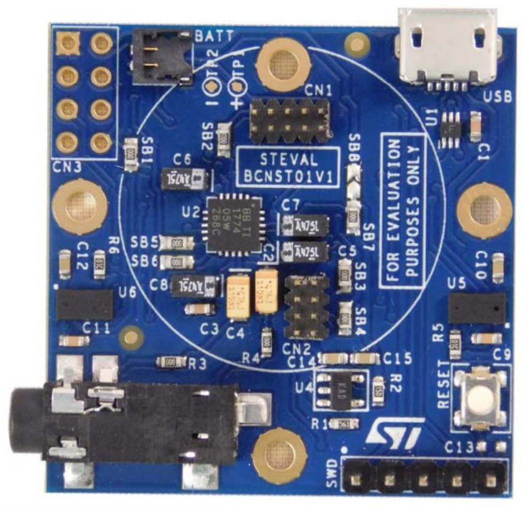
Figure 20: STEVAL-BCNCR01V1 - BlueCoin Cradle board



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 21: STEVAL-BCNST01V1 - CoinStation 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

3.2 Software description

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

- FP-AUD-BVLINK1: an application based on STM32Cube that demonstrates voice communication over BLE. The FP-AUD-BVLINK1 firmware related documentation is available on www.st.com
- One of the following development environments:
  - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
  - Ac6 System Workbench for STM32 toolchain + ST-LINK

3.3 Hardware and software setup

This section describes the hardware and software setup procedures. It also describes the required system setup.

3.3.1 Hardware setup

The following hardware components are needed:

1. For half-duplex communication:

   - two development platforms among STM32Nucleo (F401 or L476), SensorTile or BlueCoin
2. For simplex communication with an Android/iOS device:
   - one development platform among STM32Nucleo, SensorTile or BlueCoin
   - an Android (version 4.4 and above) or iOS (version 8 and above) device running ST BlueMS app
3. If you choose two STM32 Nucleo boards, two digital MEMS microphones expansion boards (order code: X-NUCLEO-CCA02M1) or two BlueNRG-MS Bluetooth low energy expansion boards (order code: X-NUCLEO-IDB05A1) are needed.
4. One USB type A to Mini-B USB cable to power up the Tx module, for STM32Nucleo; one USB type A to Micro-B USB for SensorTile or BlueCoin.
5. One USB type A to Mini-B USB cable to power up the Rx module and for USB streaming, for STM32Nucleo; one USB type A to Micro-B USB for SensorTile or BlueCoin, plus a headset or a speaker.

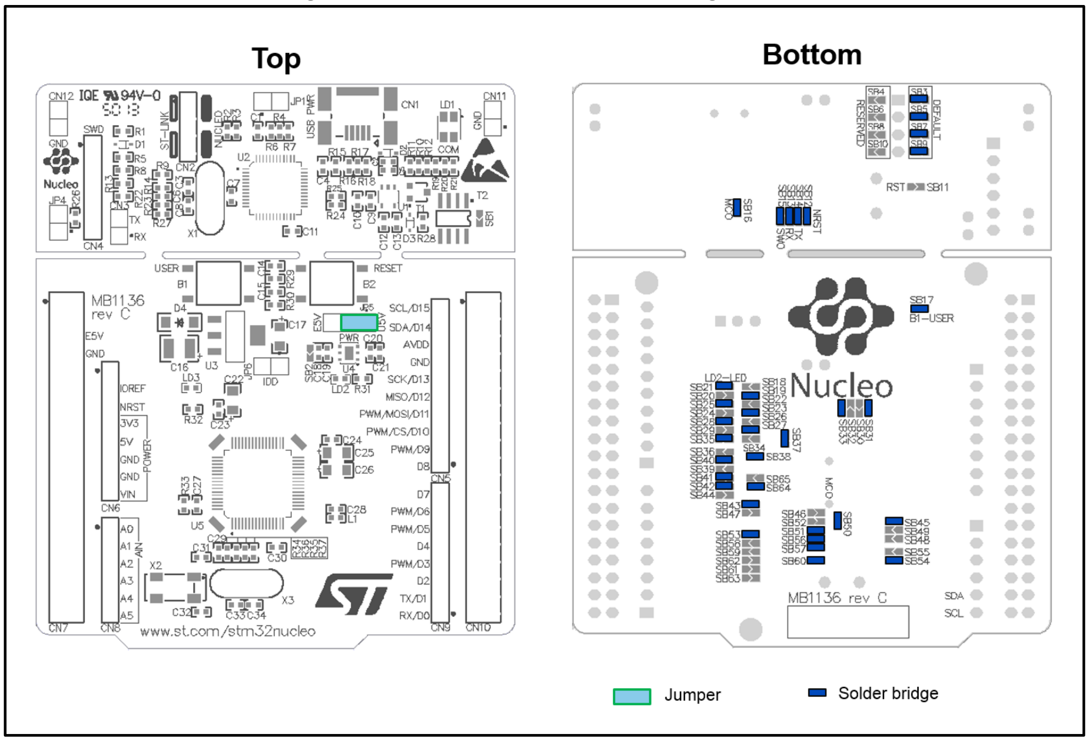
Both the STM32 Nucleo development board and the X-NUCLEO-CCA02M1 expansion board must be correctly configured to run the FP-AUD-BVLINK1 application.

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. This 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.

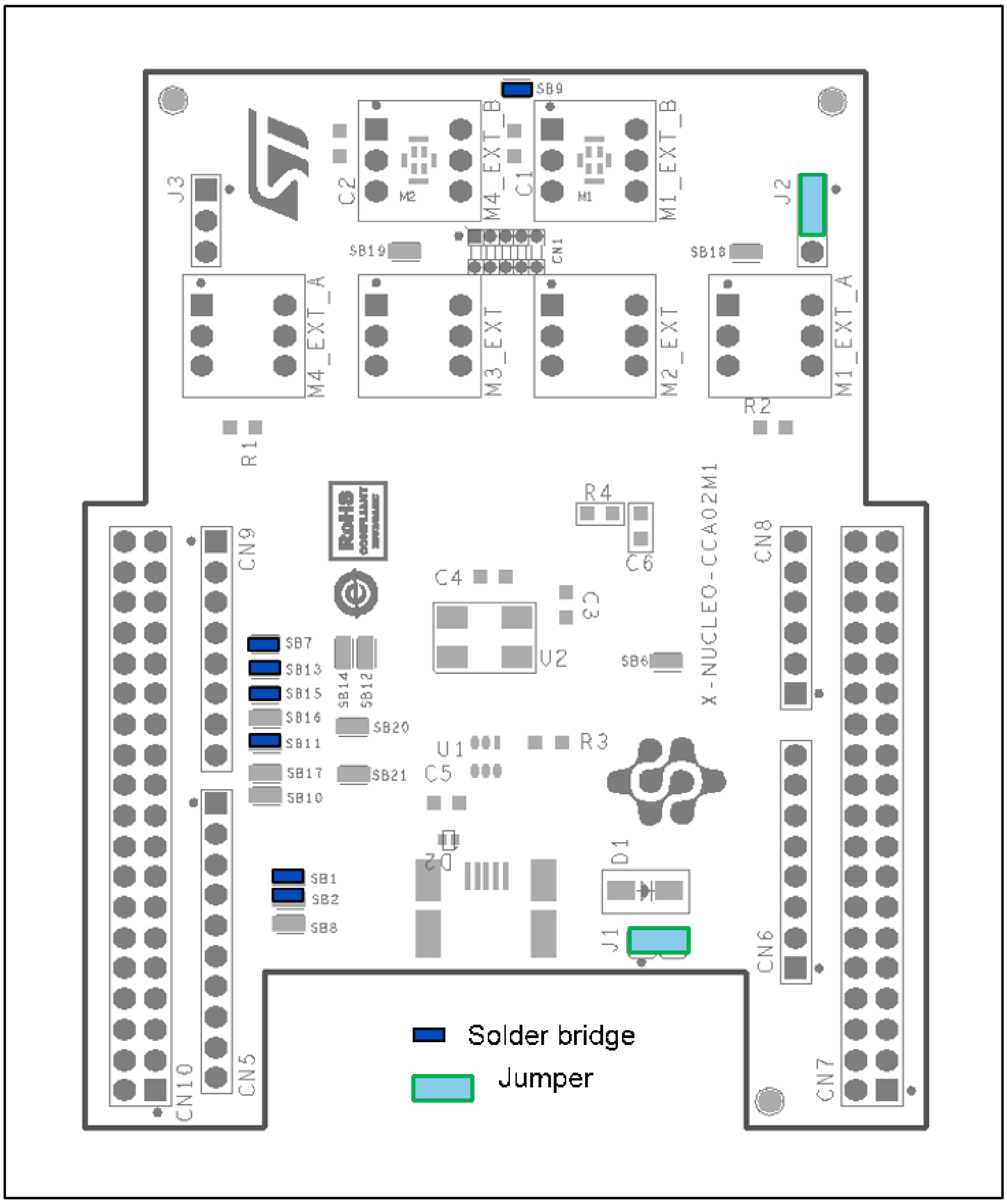
Figure 22: STM32 Nucleo board configuration



3.3.1.2 X-NUCLEO-CCA02M1 configuration

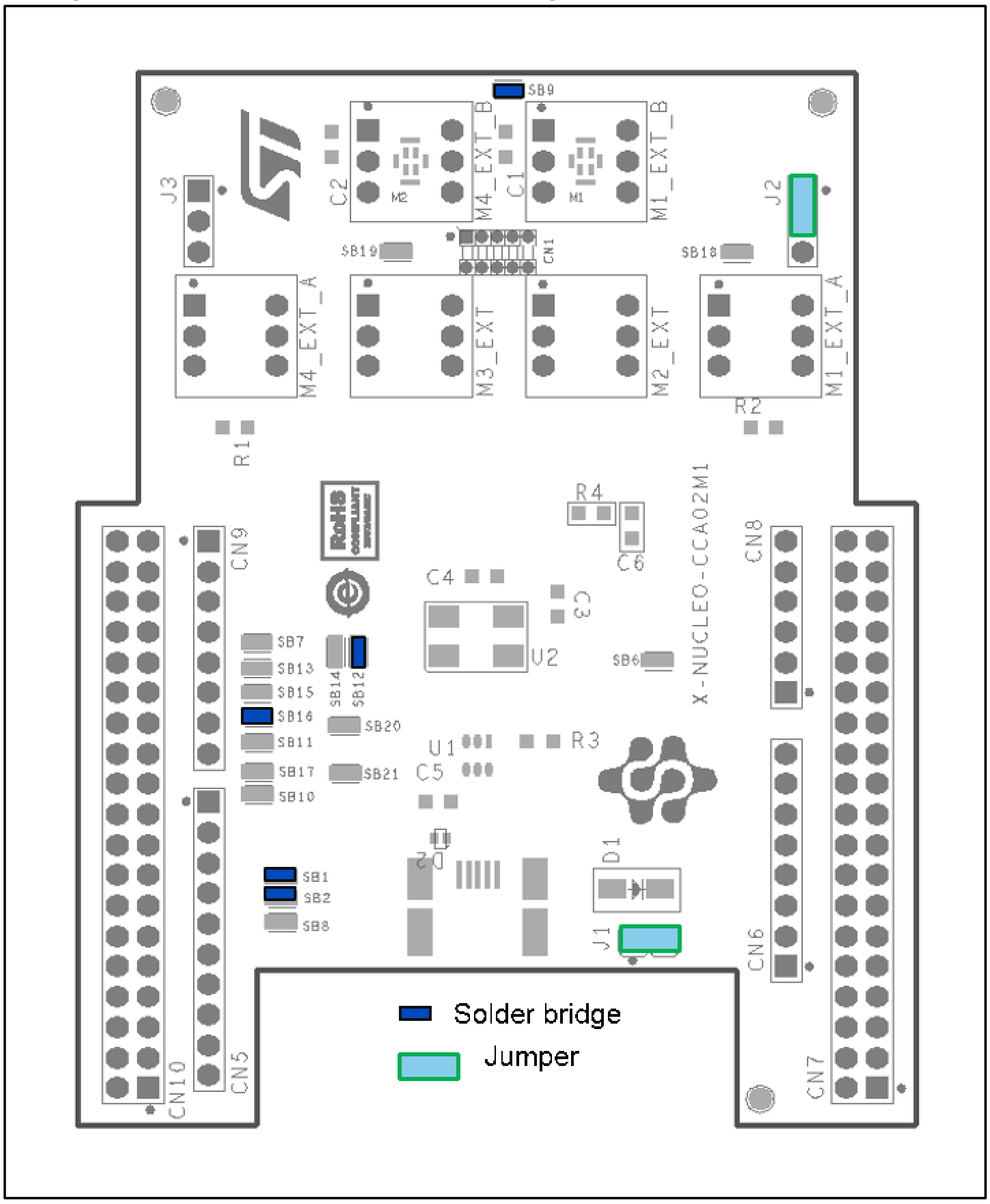
The X-NUCLEO-CCA02M1 board uses different configurations depending on the STM32 Nucleo board connected. If two NUCLEO-F401RE boards are used for half-duplex communication or a NUCLEO-L053R8 board is used for simplex communication with a mobile device, a two-microphone configuration is needed even if only one microphone is actually used by the FP-AUD-BVLINK1 application.

Figure 23: X-NUCLEO-CCA02M1 hardware configuration for NUCLEO-F401RE or NUCLEO-L053R8 boards



If two NUCLEO-L476RG boards are used for half-duplex communication, a different configuration is required; the clock is generated by the DFSDM peripheral and the PDM line of the first and second microphone is routed to the MCU.

Figure 24: X-NUCLEO-CCA02M1 hardware configuration for NUCLEO-L476RG boards



The device has to be recognized as a standard microphone to stream audio via USB; therefore, it needs to be interfaced directly with the STM32 Nucleo USB mini connector (connected to a serial port via ST-LINK) through its own mini USB connector.

Please 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 Half-duplex software setup using STM32Nucleo board

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

3.3.2.1 Development toolchains and compilers

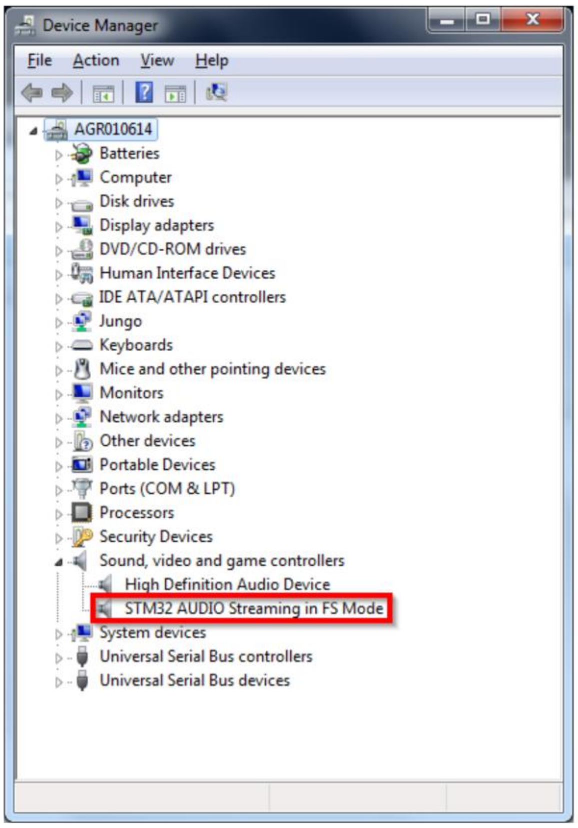
Select one of the integrated development environments supported by the STM32Cube expansion software. Read the system requirements and setup information provided by the selected IDE provider.

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

When an STM32Nucleo 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 Device Manager, as shown below.

Figure 25: Device Manager: STM32Nucleo board microphone recognition

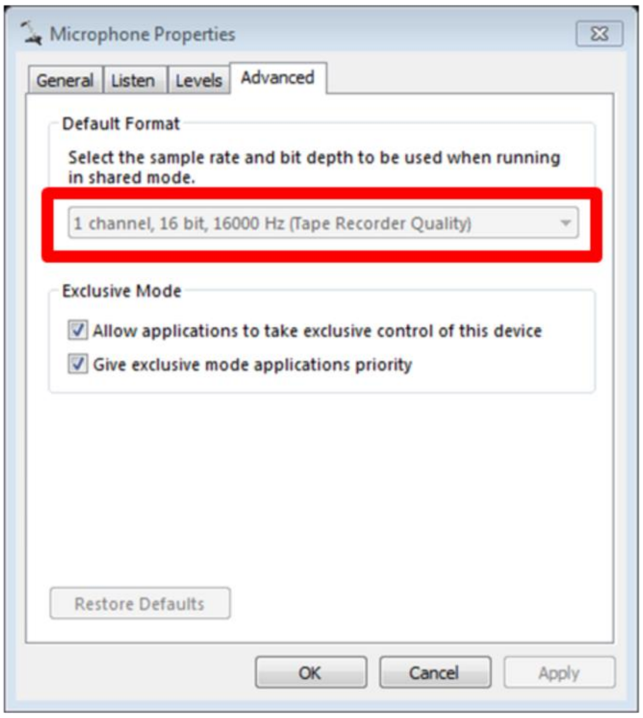


Right-click on the volume icon in the Windows task bar (on the bottom right side of the screen) and choose “recording device”. Select STM32 microphone and click on Properties.

In the Advanced tab, a summary of the current device setup appears , showing sampling frequency and number of channels. The module should be recognized as a "1 channel, 16000 Hz" microphone.

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

Figure 26: Advanced tab: microphone properties



Default Format

Select the sample rate and bit depth to be used when running in shared mode.

1 channel, 16 bit, 16000 Hz (Tape Recorder Quality)

Exclusive Mode

☑ Allow applications to take exclusive control of this device

☑ Give exclusive mode applications priority

3.3.3 FP-AUD-BVLINK1 setup

This section describes how to set up the FP-AUD-BVLINK1 half-duplex application to demonstrate audio transmission over Bluetooth low energy. Two BLE devices interact with each other in order to set up point-to-point wireless communication. One of the modules acts as the central module and the other as a peripheral module.

3.3.3.1 STM32 Nucleo and expansion board setup

The STM32Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. You can find the relevant version of the ST-LINK/V2-1 USB driver, STSW-LINK008 or STSW-LINK009 (according to the Windows™ version used), on www.st.com.

The X-NUCLEO-CCA02M1 microphone expansion board can be easily connected to the STM32Nucleo development board through the ST morpho extension connector. The board can interface with the external STM32Nucleo microcontroller via I²S and USB.

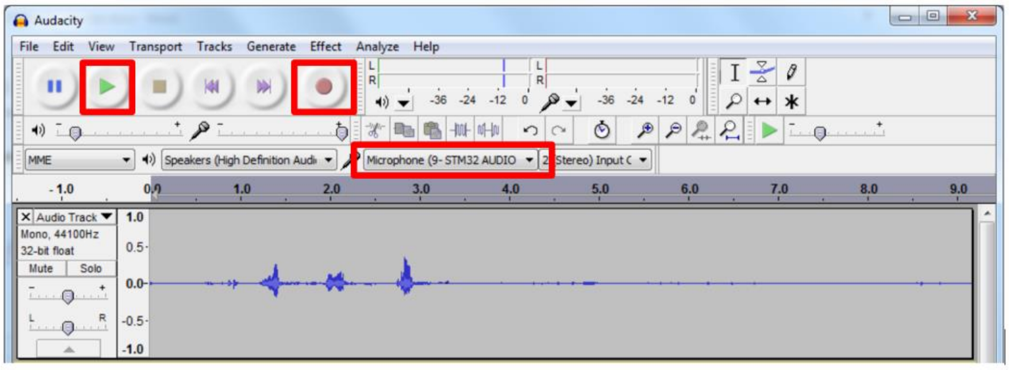
The X-NUCLEO-IDB05A1 BlueNRG-MS expansion board can be easily connected to the X-NUCLEO-CCA02M1 via the Arduino UNO R3 extension connector.

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 27: Audacity for Windows

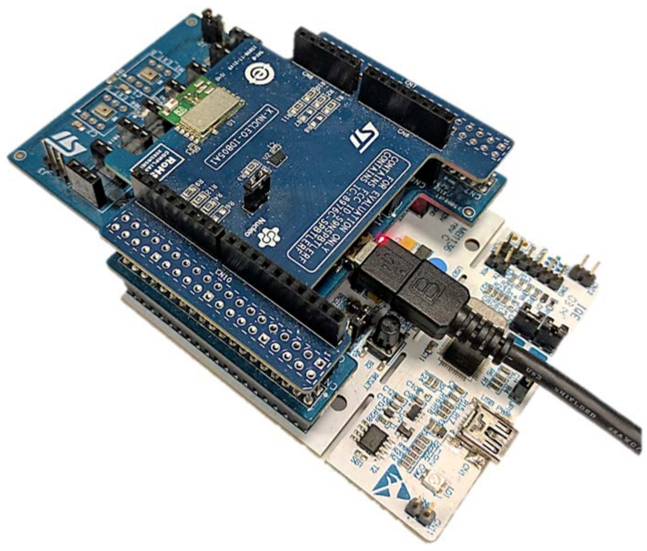

Audacity
File Edit View Transport Tracks Generate Effect Analyze Help
MME
Speakers (High Definition Audi
Microphone (9- STM32 AUDIO
2 (Stereo) Input C
Audio Track
Mono, 44100Hz
32-bit float
Mute Solo

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

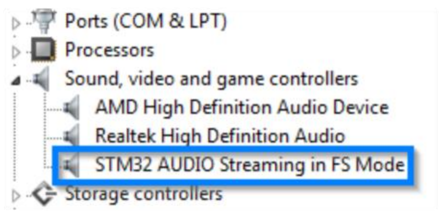
1 Connect the X-NUCLEO-CCA02M1 USB connector of the central unit to the PC.

Figure 28: X-NUCLEO-CCA02M1 USB connection to the PC: module setup



The STM32 Audio Streaming peripheral appears in the Device Manager.

Figure 29: STM32 AUDIO Streaming peripheral

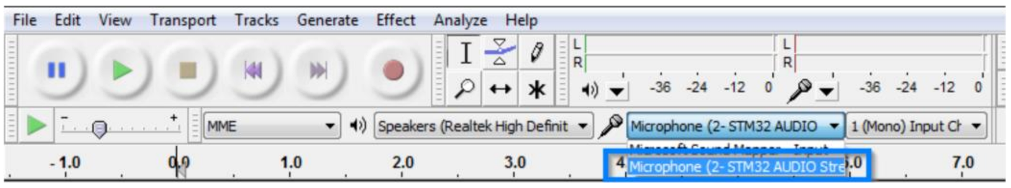

Ports (COM & LPT)
Processors
Sound, video and game controllers
AMD High Definition Audio Device
Realtek High Definition Audio
STM32 AUDIO Streaming in FS Mode
Storage controllers

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 30: Central unit microphone in Audacity

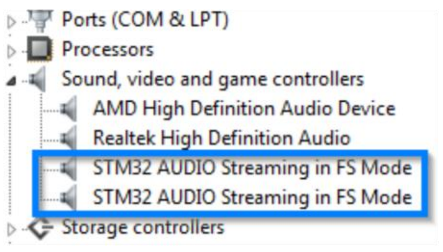


MME
Speakers (Realtek High Definit
Microphone (2- STM32 AUDIO
1 (Mono) Input Ch
Microsoft Sound Mapper - Input
Microphone (2- STM32 AUDIO Stre

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 31: STM32 recognized as Audio Streaming peripheral

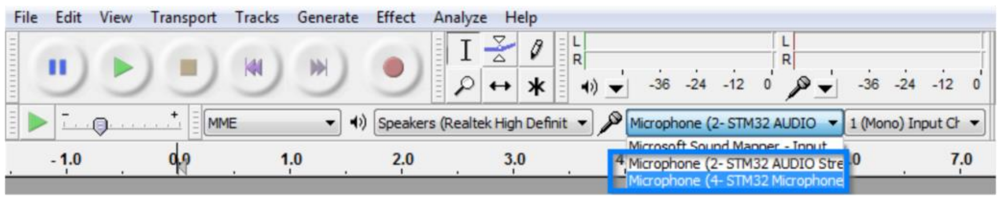

Ports (COM & LPT)
Processors
Sound, video and game controllers
AMD High Definition Audio Device
Realtek High Definition Audio
STM32 AUDIO Streaming in FS Mode
STM32 AUDIO Streaming in FS Mode
Storage controllers

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 32: Peripheral unit microphone in Audacity

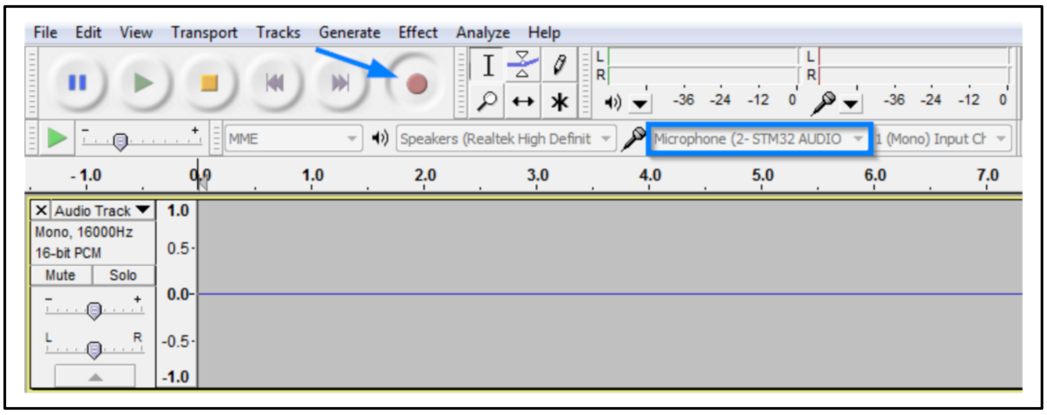


MME
Speakers (Realtek High Definit
Microphone (2- STM32 AUDIO
1 (Mono) Input Ch
Microsoft Sound Mapper - Input
Microphone (2- STM32 AUDIO Stre
Microphone (4- STM32 Microphone

3.3.3.1.3 Peripheral to central recording

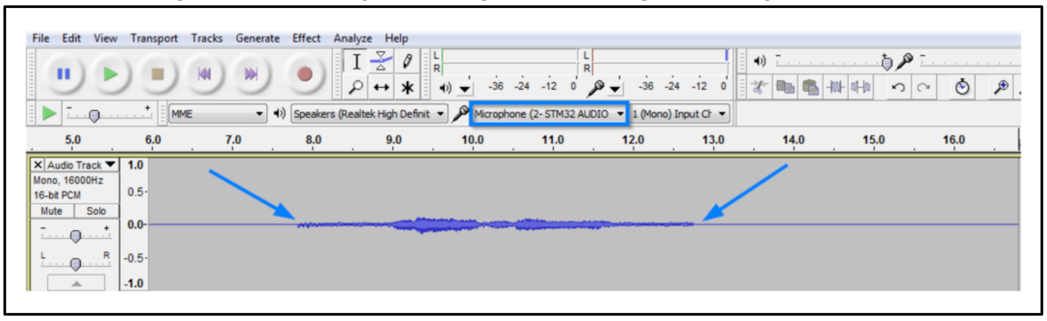
1. Choose the Central unit microphone (microphone 2)
2. Click Record to start silent recording

Figure 33: Audacity silent recording from Central unit USB stream



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

Figure 34: Audacity recording voice coming from Peripheral unit

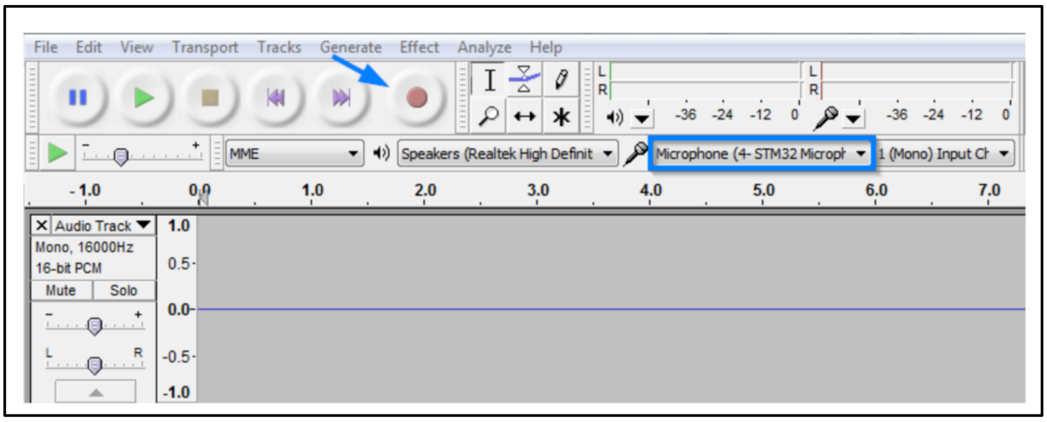


4. Click the button to toggle the streaming status.

3.3.3.1.4 Central to peripheral recording

1. Choose the Peripheral unit microphone (microphone 4).
2. Click Record to start silent recording.

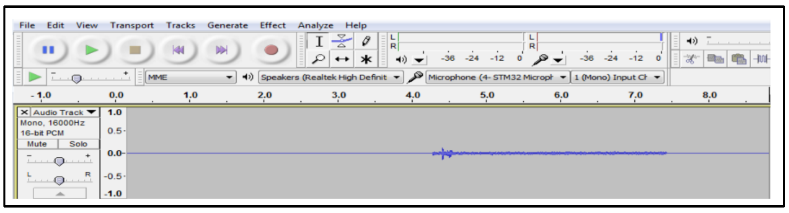
Figure 35: Audacity silent recording from Peripheral unit USB stream



3 Click central unit user button

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

Figure 36: Audacity recording voice coming from Central unit



4 Click the button to toggle the streaming status.

3.3.3.2 SensorTile system setup

The FP-AUD-BVLINK1 supports the STEVAL-STLKT01V1. To create a half-duplex communication, two SensorTile kits are necessary (each kit 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.

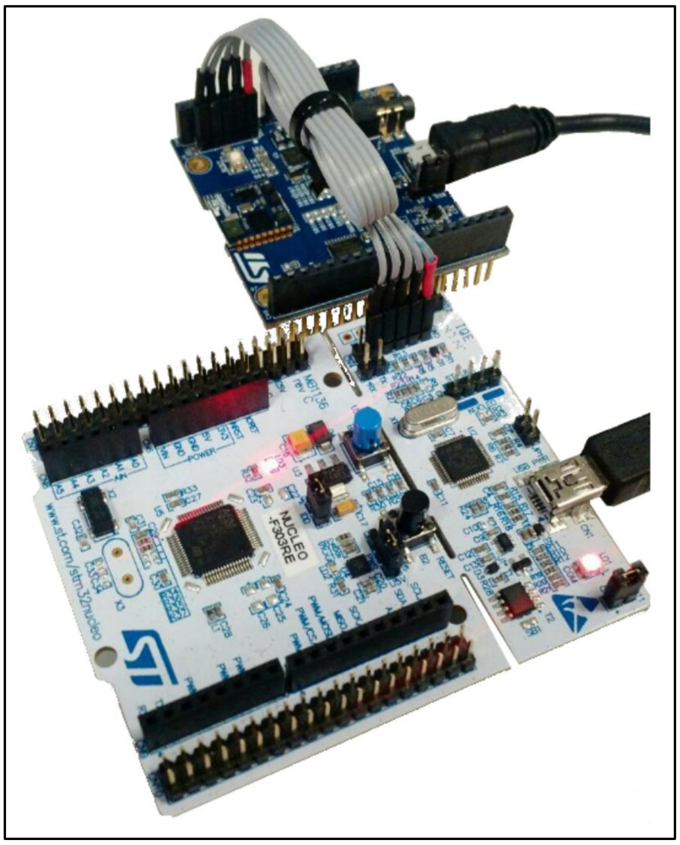
The SensorTile core system (STEVAL-STLCS01V1) must be plugged to the cradle expansion (STEVAL-STLCX01V1) through the dedicated connector.

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 37: SensorTile-STM32Nucleo: SWD connections with a 5-pin flat cable

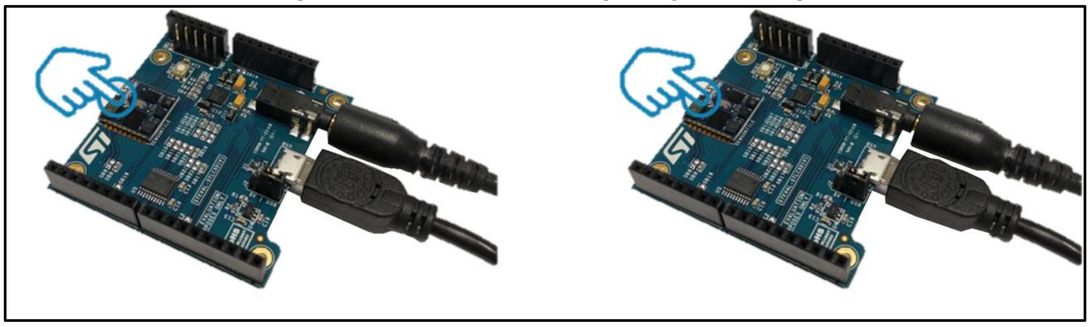


It is possible to reproduce the streamed audio with a headset or a speaker connected to the 3.5 mm phone jack.

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.

Only one module can stream at a time.

Figure 38: SensorTile half-duplex system setup



3.3.3.3 BlueCoin system setup

The FP-AUD-BVLINK1 also supports the STEVAL-BCNKT01V1. To create a half-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

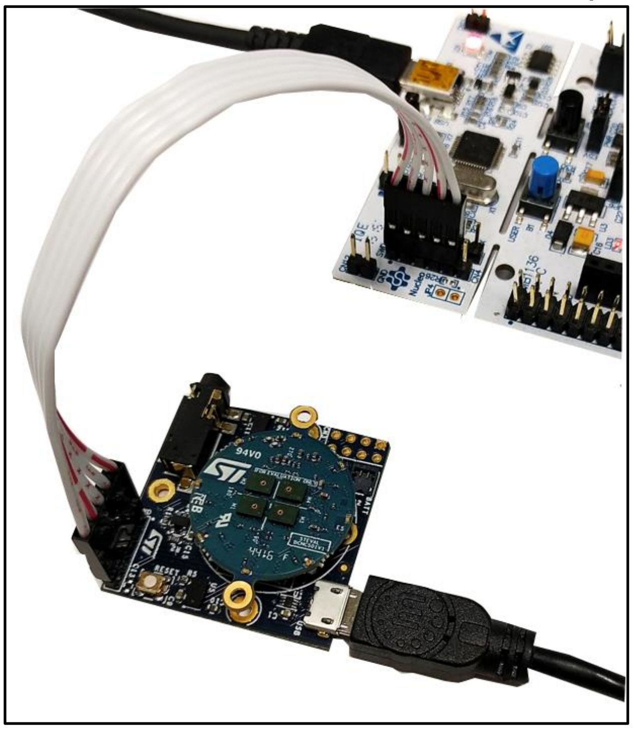
Two modules are needed in order to create a BLE link: one peripheral and one central.

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 39: BlueCoin-STM32Nucleo: SWD connections with 5-pin flat cable

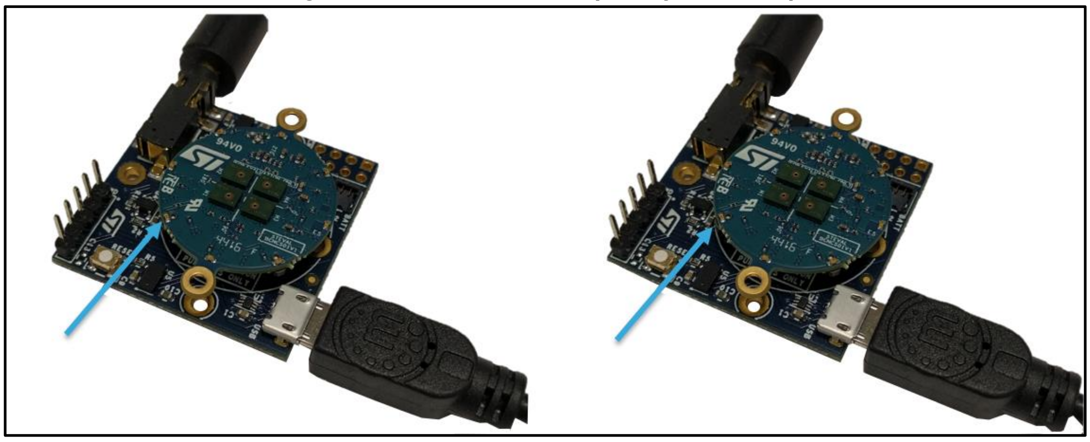


It is possible to reproduce the streamed audio with a headset or a speaker connected to the 3.5 mm phone jack.

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.

Only one module can stream at a time.

Figure 40: BlueCoin half-duplex system setup



4 FP-AUD-BVLINK1 Android™/iOS™ demo setup

The BlueVoice profile is compatible with the ST BlueMS app (Version 3.0.0 or higher) available on Android/iOS stores.

A single direction audio stream can be generated from the STM32Nucleo to Android/iOS devices. In this scenario, the ST module acts as the Peripheral node, whereas the smartphone/tablet is the Central node.

You can use the FP-AUD-BVLINK1 Peripheral module to stream the audio acquired at 8 kHz and compressed via ADPCM to the mobile device.

4.1 Hardware setup

The setup requires an ST module that acts as Peripheral node. It can be composed of an STM32Nucleo stack, a SensorTile plugged to a cradle board or a BlueCoin connected to a cradle board or a Coin Station.

4.2 Software setup

4.2.1 Peripheral firmware

The ST BlueMS app accepts audio input acquired at 8 kHz and compressed via ADPCM.

To configure the Peripheral module accordingly, open the BlueVoiceLink Peripheral firmware with one of the available toolchains and modify the source code as follows:

- `bluevoice_application_peripheral.h` change the sampling frequency define:
  - `#define AUDIO_IN_SAMPLING_FREQUENCY (uint16_t) (SAMPLING_FREQ_8000)`
- `bluevoice_app_per_xxxx.c`: change the sampling_frequency parameter of the `BV_ADPCM_Config` structure (`BV_APP_PER_ProfileInit` function):
  - `BV_ADPCM_Config.sampling_frequency = FR_8000;`

The modified firmware must be recompiled and flashed on the Peripheral module.

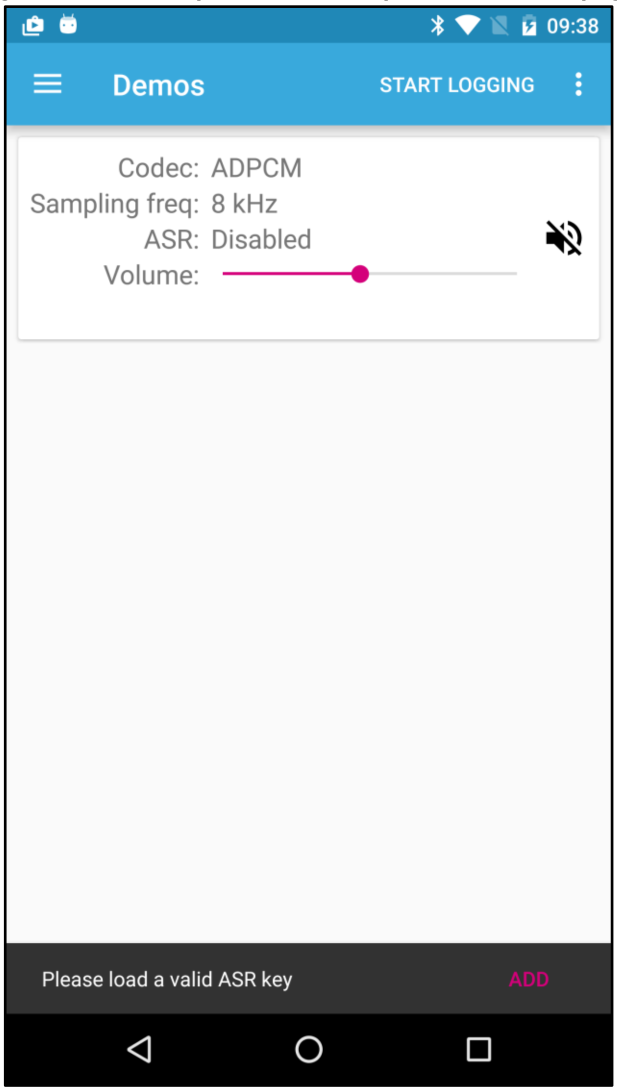
4.2.2 ST BlueMS app

FP-AUD-BVLINK1 is compatible with the ST BlueMS app v3.0.0 or higher, available at the Android and iOS stores. For further information on the BlueMS app, refer to the FP-SNS-ALLMEMS1 User Manual on www.st.com.

If the BlueVoiceADPCM voice over BLE library is enabled, the following page is available with the following functions:

- Play back the audio stream received from the ST device.
- Web-based Google ASR service.
- Web-based chinese ASR: iFlyTek MSC service.

Figure 41: BlueMS (Android version) BlueVoice start page



Codec: ADPCM
Sampling freq: 8 kHz
ASR: Disabled
Volume:

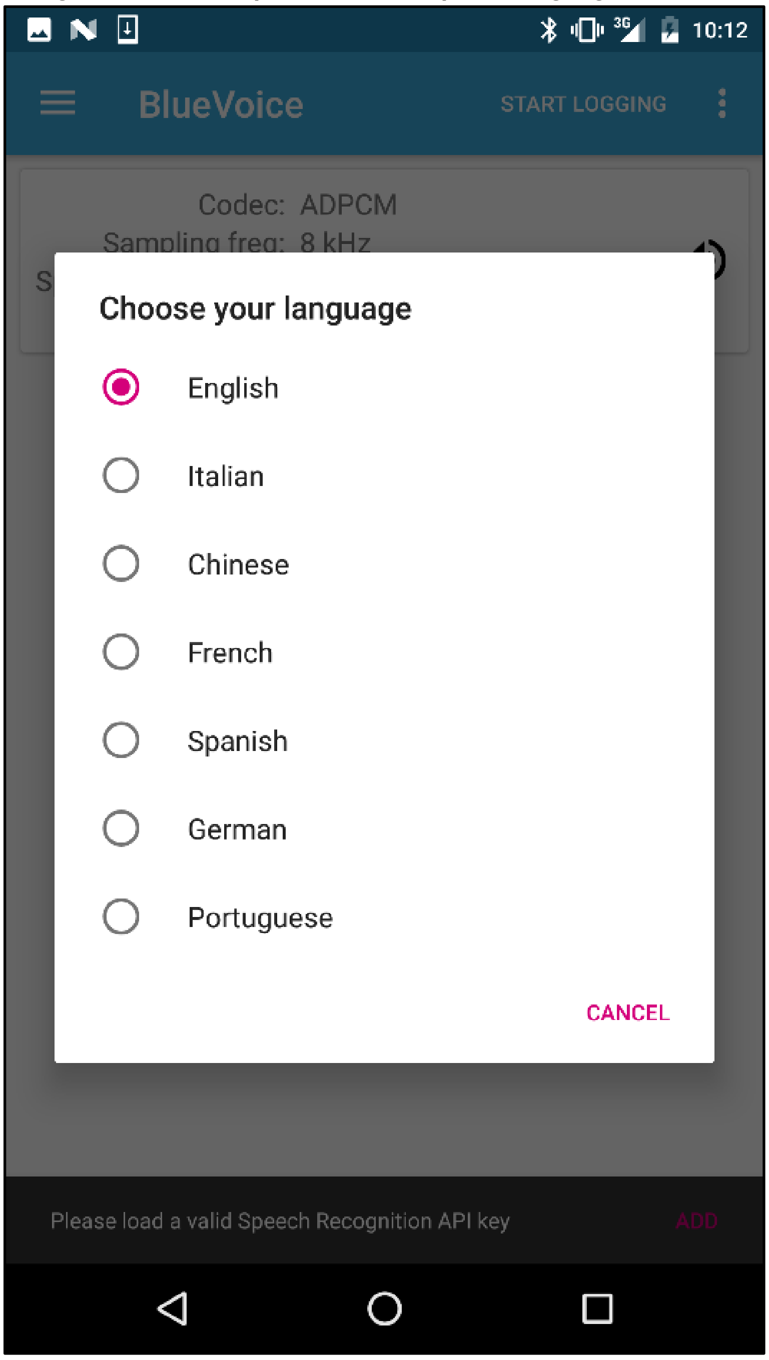
Please load a valid ASR key

The audio playback begins as soon as the streaming from the peripheral node starts. The volume can be adjusted using the slider or muted by clicking on the speaker icon.

4.2.2.1 ASR language selection

Opening the ASR language menu, in the demo main menu, the application displays a popup window that allows the ASR language selection. According to the language selected, a specific ASR service will be configured.

Figure 42: BlueMS (Android version) ASR language selection



Codec: ADPCM

Sampling freq: 8 kHz

Choose your language

[✓ English]

Italian

Chinese

French

Spanish

German

Portuguese

Please load a valid Speech Recognition API key

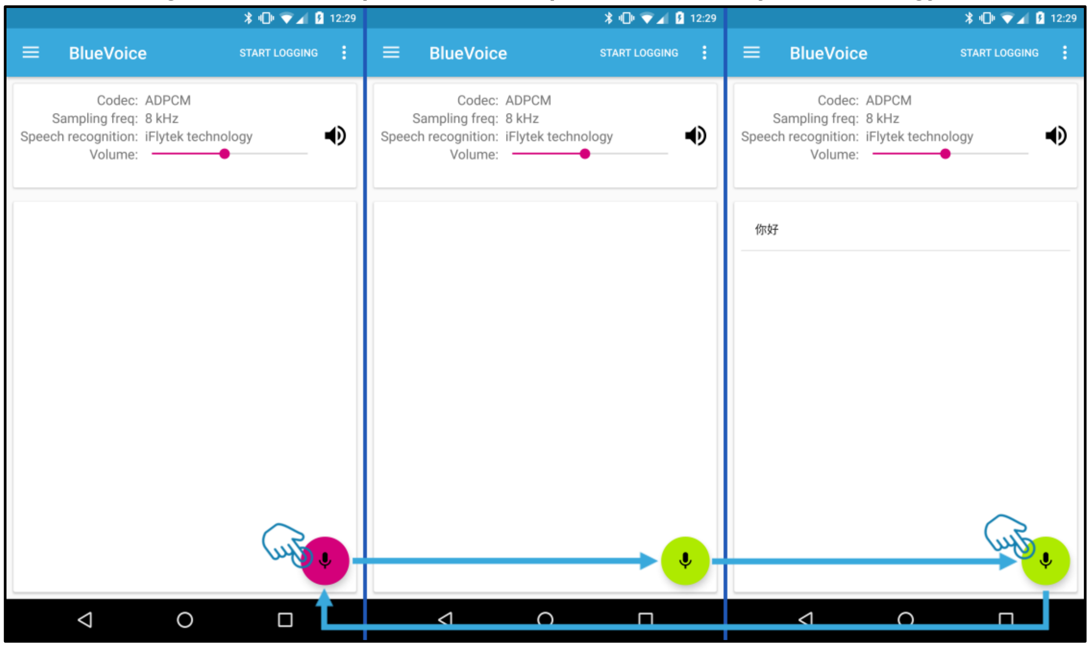
4.2.2.1.1 Chinese ASR: iFlyTek MSC service

When Chinese is selected, the ASR service provided by iFlyTek is enabled.

Pushing the button on the bottom right hand of the screen, it becomes green and the speech-to-text service starts.

The recognition is continuous and every sentence is recorded as shown below.

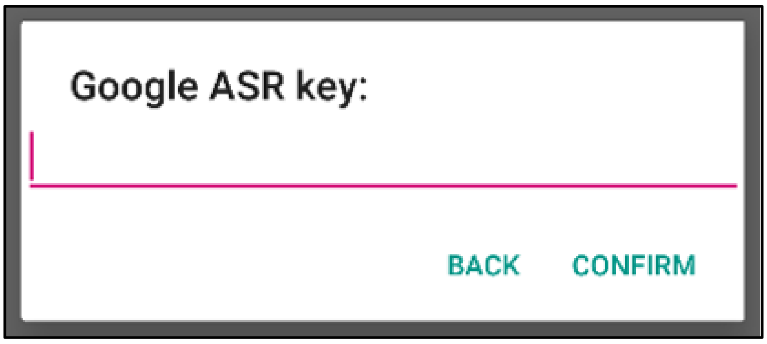
Figure 43: BlueMS (Android version) Chinese ASR, iFlytek technology



4.2.2.1.2 Alternative languages: Google Speech API

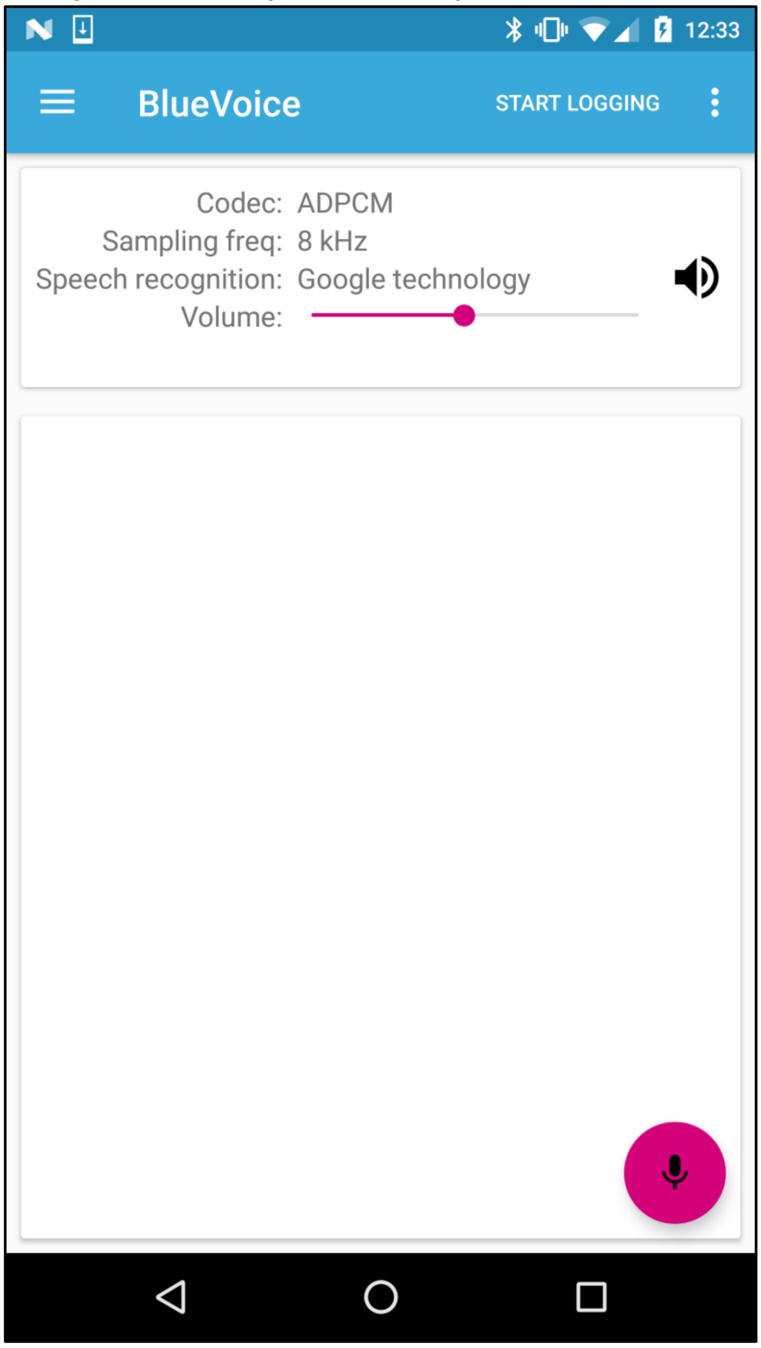
The ADD button allows the insertion of the key to enable the ASR feature: a popup window prompts the insertion of a valid API key, followed by the ASR service activation key.

Figure 44: BlueMS (Android version) popup API key window



Once the key is correctly inserted, the start screen changes.

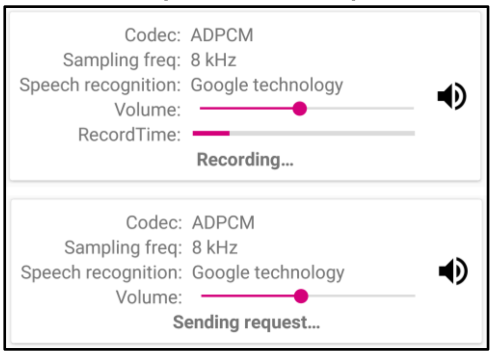
Figure 45: BlueMS (Android version) ASR service enabled



Codec: ADPCM
Sampling freq: 8 kHz
Speech recognition: Google technology
Volume:

Hold the recording button to register your voice for subsequent recognition. While the button is pressed, a bar progressively indicates the elapsed recording time. When you release the button a “Sending request…” message appears.

Figure 46: BlueMS (Android version) voice recording

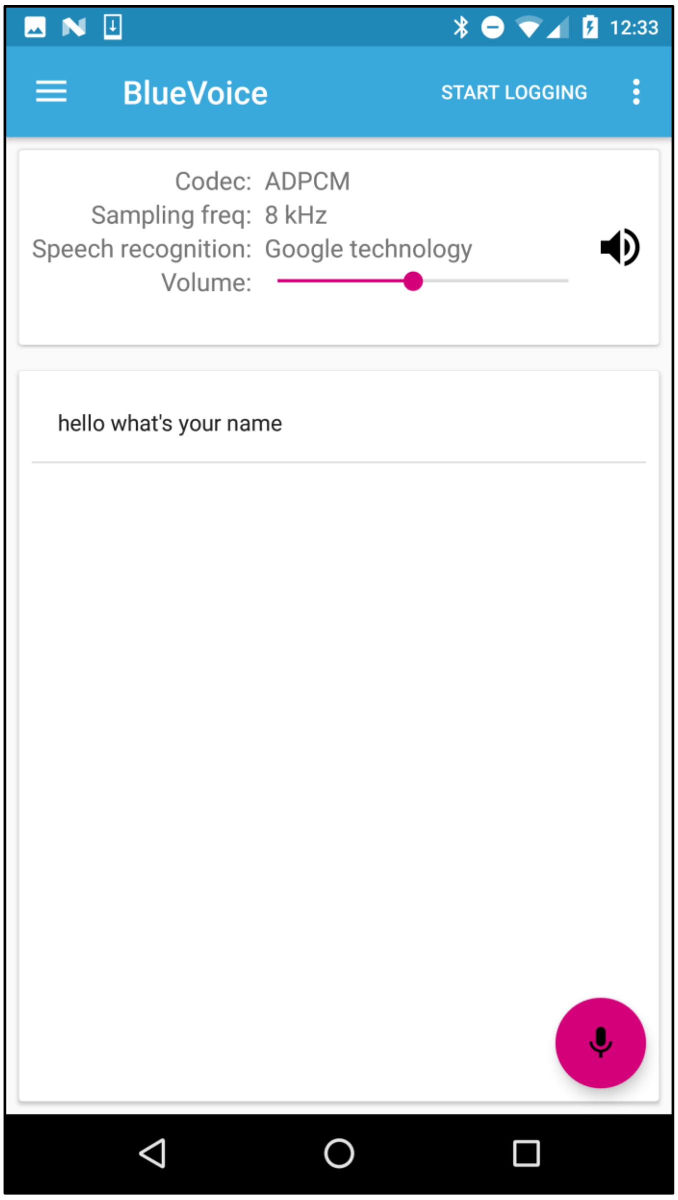


Codec: ADPCM
Sampling freq: 8 kHz
Speech recognition: Google technology
Volume:
RecordTime:
Recording...

Codec: ADPCM
Sampling freq: 8 kHz
Speech recognition: Google technology
Volume:
Sending request...

Eventually, the speech recognized by the ASR service appears below the volume bar.

Figure 47: BlueMS (Android version) recognized voice text



12:33
BlueVoice
START LOGGING

Codec: ADPCM
Sampling freq: 8 kHz
Speech recognition: Google technology
Volume:

hello what's your name

If the recording cannot be recognized, a “Token not recognized’ message appears instead of the text

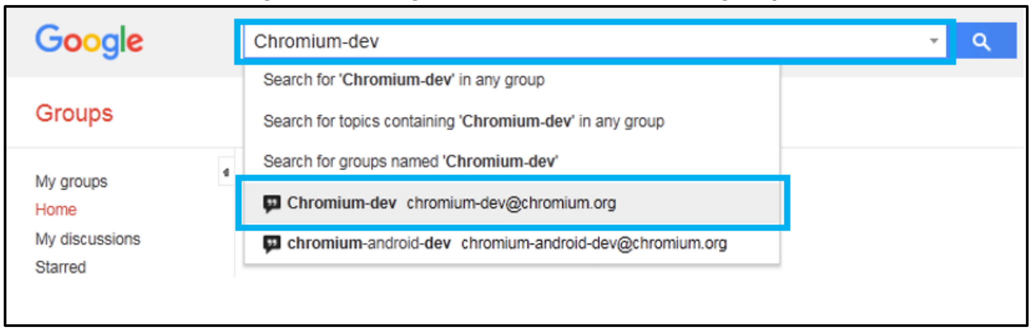
4.2.2.2 Google speech ASR Key generation

The Google Speech APIs require a key to access the web-based service. You need a Google account to complete the procedure and access the service.

To generate a key:

1. Login with your own Google account.
2. Subscribe to Chromium-dev at *https://groups.google.com/a/chromium.org/forum/?fromgroups#!forum/chromium-dev*.
3. Write “Chromium-dev” in the search box, and select the appropriate group.

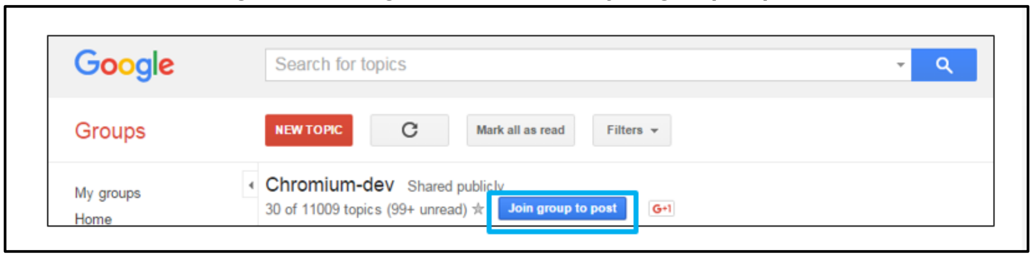
Figure 48: Google Chromium-dev: search group



Search for 'Chromium-dev' in any group
Search for topics containing 'Chromium-dev' in any group
Search for groups named 'Chromium-dev'
Chromium-dev chromium-dev@chromium.org
chromium-android-dev chromium-android-dev@chromium.org

4. Click on “Join group to post” button

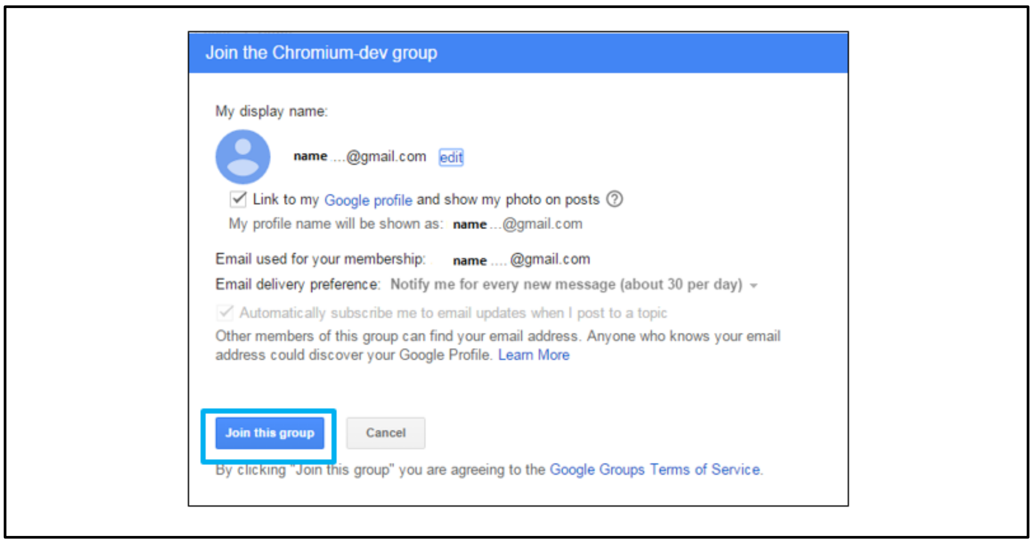
Figure 49: Google Chromium-dev: join group to post



Chromium-dev Shared publicly
30 of 11009 topics (99+ unread)

5 Click on “Join this group” button to join the Chromium-dev group.

Figure 50: Google Chromium-dev: join the group



Join the Chromium-dev group

My display name:

name ...@gmail.com edit

☑ Link to my Google profile and show my photo on posts

My profile name will be shown as: name ...@gmail.com

Email used for your membership: name .... @gmail.com

Email delivery preference: Notify me for every new message (about 30 per day)

☑ Automatically subscribe me to email updates when I post to a topic

Other members of this group can find your email address. Anyone who knows your email address could discover your Google Profile. Learn More

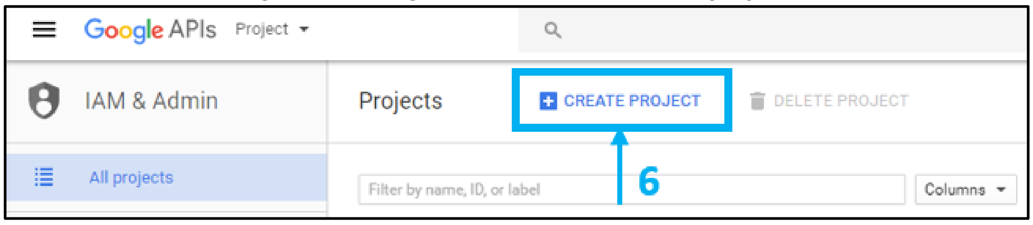
Join this group Cancel

By clicking "Join this group" you are agreeing to the Google Groups Terms of Service.

6 Go to *https://console.developers.google.com/project*

7 Click on “Create a project…”

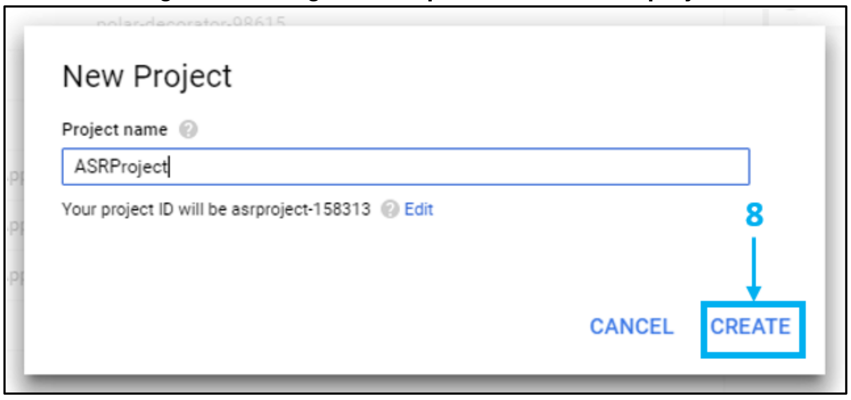
Figure 51: Google Chromium-dev: create project



8 Choose the Project name.

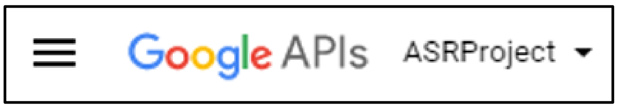
9 Click on “Create” button.

Figure 52: Google Developers Console: new project



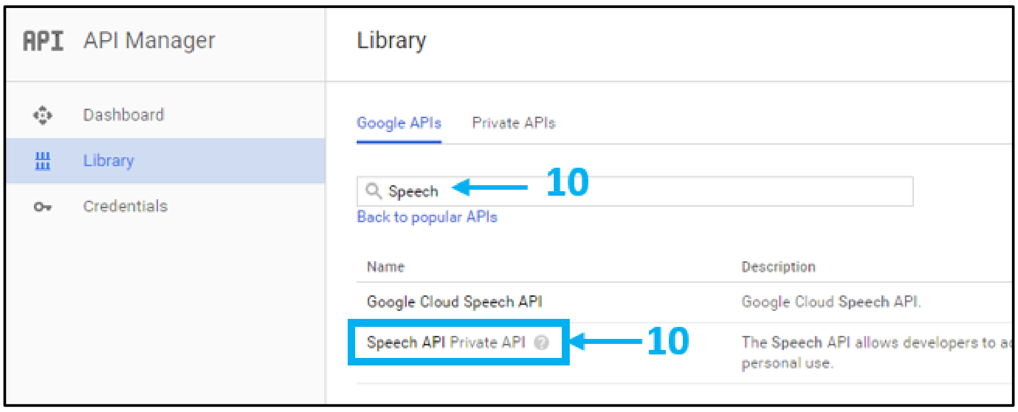
10 Make sure you have selected the newly created project.

Figure 53: Google Developers Console: ASRProject



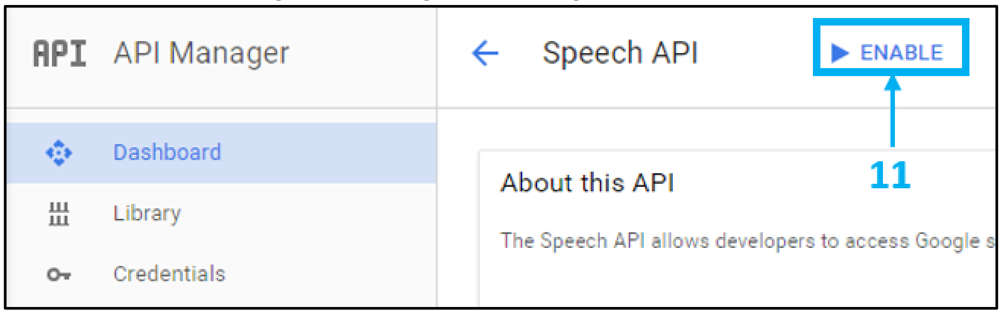
11 Write "Speech API" in the search box, and select correct result.

Figure 54: Google Developers Console: select API



Name	Description
Google Cloud Speech API	Google Cloud Speech API.
Speech API Private API	The Speech API allows developers to a personal use.

12 Enable the Speech API clicking on the blue button.

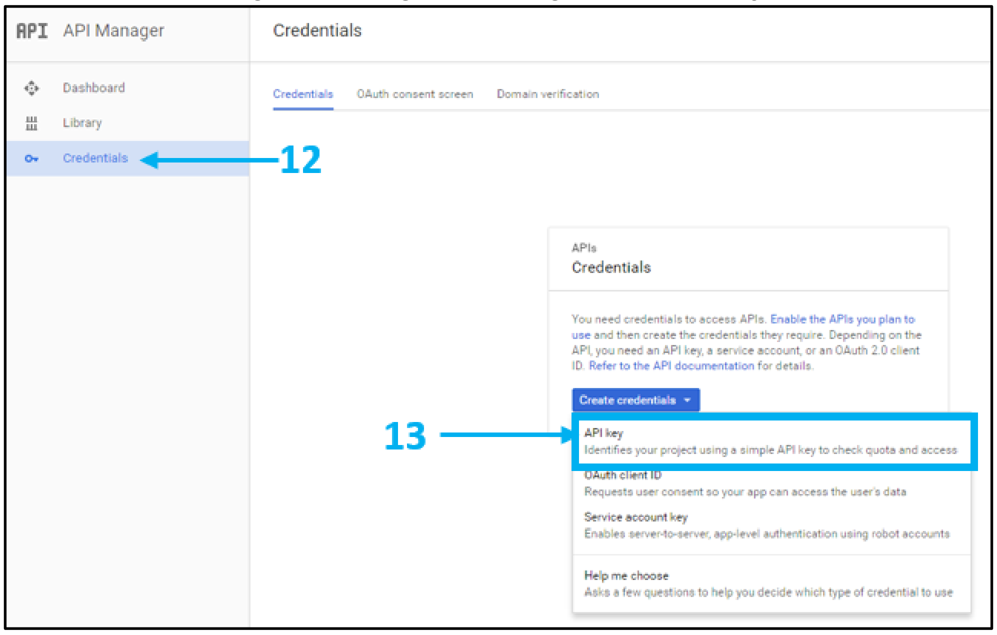


The Speech API allows developers to access Google s

Figure 55: Google API Manager: enable API

13 Move from the “Dashboard” tab to “Credentials” tab.

14 Open the “Create credentials” menu and select “API key”.



You need credentials to access APIs. Enable the APIs you plan to use and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. Refer to the API documentation for details.

API key
Identifies your project using a simple API key to check quota and access

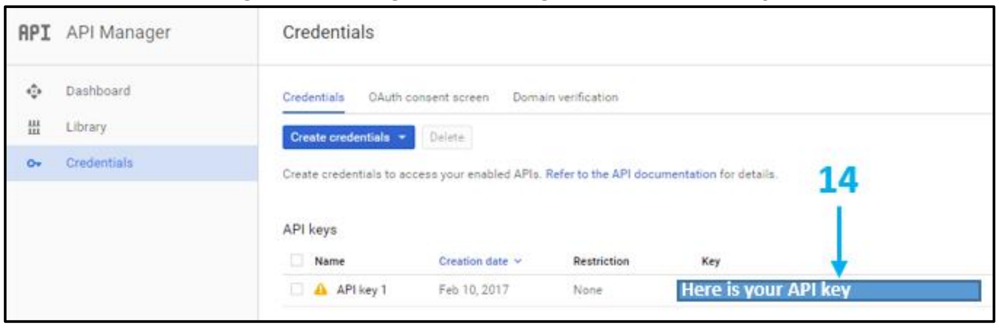
OAuth client ID
Requests user consent so your app can access the user's data

Service account key
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

Figure 56: Google API Manager: create API key

15 Your API key is created. Click on Close to return to the Credentials section. Here you can see your API Key.



Create credentials to access your enabled APIs. Refer to the API documentation for details.

API keys

Name	Creation date	Restriction	Key
API key 1	Feb 10, 2017	None	Here is your API key

Figure 57: Google API Manager: Android API key

5 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. MP34DT01-M MEMS audio sensor omnidirectional digital microphone, datasheet on *www.st.com*
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.

6 Revision history

Table 4: Document revision history

Date	Version	Changes
12-Jun-2017	1	Initial release.