UM1898
User manual

Getting started with the BLUEVOICELINK1 Bluetooth LE and digital MEMS microphones software expansion for STM32Cube

Introduction

BLUEVOICELINK1 is an expansion software package for STM32Cube. The software runs on the STM32 and includes drivers and middleware for Bluetooth Low Energy (BlueNRG and BlueNRG-MS) and MP34DT01-M 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-IDB04A1 or X-NUCLEO-IDB05A1 plus X-NUCLEO-CCA02M1, when connected to a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L053R8 board.

Information about 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 BLUEVOICELINK1 software description

2.1 Overview

The key features of the BLUEVOICELINK1 package are:

- OSXBLUEVOICE, half-duplex voice-over-Bluetooth low energy communication profile (under OPEN.AUDIO license)
- Very low power Bluetooth low energy single-mode network processor, compliant with Bluetooth specification core 4.0 (BlueNRG) and 4.1 (BlueNRG-MS)
- Complete middleware to build applications using digital MEMS microphones (MP34DT01-M)
- 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-IDB04A1 or X-NUCLEO-IDB05A1 plus X-NUCLEO-CCA02M1 connected to a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L053R8 board
- Compatibility with ST BlueMS app (v3.0.0 or higher), available for Android™ and iOS™

The proposed software is based on the STM32CubeHAL, a hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for BlueNRG/BlueNRG-MS and MP34DT01-M MEMS microphone expansion boards, 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 (OSXBLUEVOICE library, under OPEN.AUDIO license).

The BlueVoice 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. OSXBLUEVOICE 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/BlueNRG-MS is a very low power Bluetooth Low Energy (BLE) single-mode network processor, compliant with Bluetooth specification core 4.0 (BlueNRG) and 4.1 (BlueNRG-MS).

The drivers abstract low-level details of the hardware and allow the middleware components and applications to access the devices in a hardware-independent fashion. The software implements low power optimizations to allow system power consumption of just a few microamperes.

The package includes a sample application that developers can use to start experimenting with the code. The sample application enables acquisition, compression and transmission over Bluetooth Low Energy of voice data from the module that is acting as transmitter to the one that is acting as 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

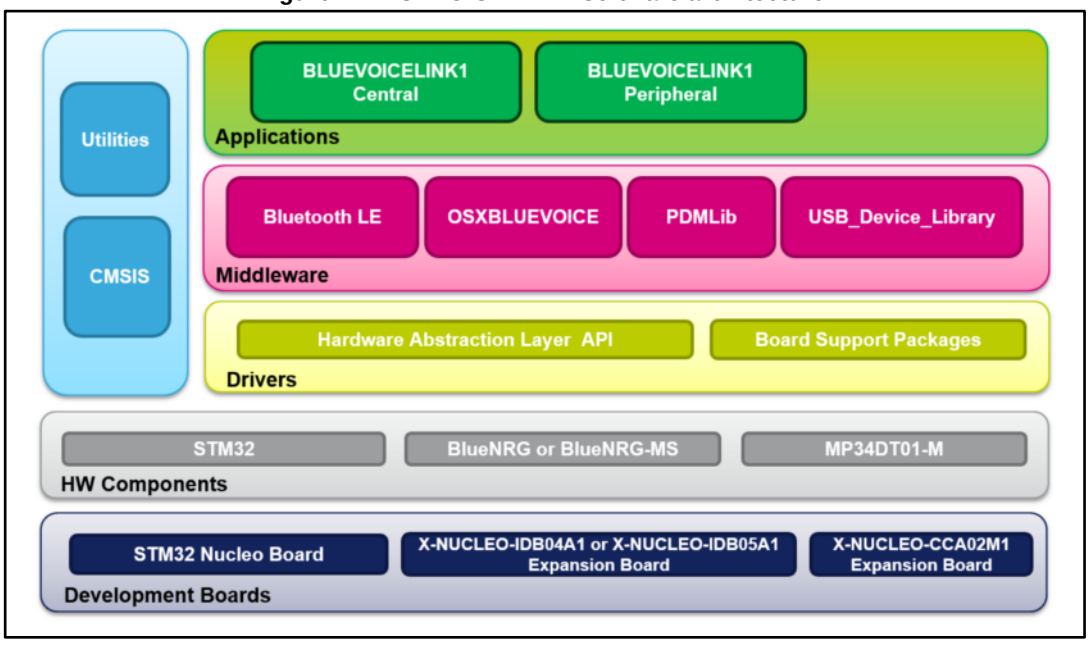
The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller.

The package provides a board support package (BSP) for the Bluetooth low energy (X-NUCLEO-IDB04A1/X-NUCLEO-IDB05A1) and the digital MEMS microphone (X-NUCLEO-CCA02M1) expansion boards, together with a set of middleware components for audio acquisition, voice transmission over Bluetooth low energy and USB audio in class implementation.

The application works with the following software layers:

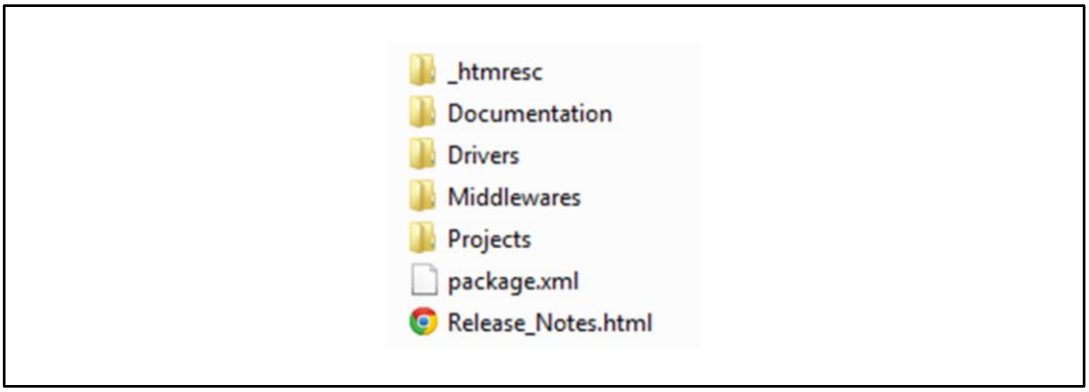
- **STM32Cube HAL layer**: provides a simple, generic and multi-instance of APIs (application programming interfaces) to interact with the upper application, library and stack layers. These generic and extension APIs are built on a common architecture and allow layers built on top of them like middleware to implement functions without requiring specific microcontroller unit (MCU) hardware information. This structure improves library code reusability and guarantees easy portability to other 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 the LED and user button; it also helps in identifying the specific board version.
  - for the Bluetooth low energy expansion board, it provides a set of APIs for initialization and communication with the BlueNRG component
  - for microphone acquisition boards, it provides APIs for audio acquisition and processing.

Figure 1: BLUEVOICELINK1 software architecture



2.3 Folder structure

Figure 2: BLUEVOICELINK1 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 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 OSXBLUEVOICE library.
- **Projects**: contains central and peripheral applications to demonstrate voice transmission over BLE. The projects are supplied for the NUCLEO-F401RE 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

This section describes the design of the application with respect to the Bluetooth low energy architecture.

2.5.1.1 Generic access profile (GAP)

Bluetooth Low Energy 4.0 communications can be either broadcast or connection-based. The BlueVoice application deploys a connection-based communication paradigm insofar as it provides a permanent point-to-point link between two devices. Data sent through a BLE connection is organized through an additional protocol layer, the Generic Attribute Profile (GATT).

The Bluetooth Specification v 4.1 [*1*] specifies the following device roles:

- A **Central** role supporting multiple connections and initiating connections with peripheral devices. These devices require a controller that supports the master role with more complex functions.

- A **Peripheral** role for devices supporting a single connection and less complexity. These devices only require a controller that supports the slave role. They use the frequency of the central controller to data with it.

Figure 3: BlueVoice Profile master-slave GAP role assignment

Central and peripheral role assignment is consistent with the asymmetric design concept of BLE, where the device with a lower energy source is given less to do: a slave cannot initiate complex procedures, whereas a master manages communication timing, adaptive frequency hopping, sets encryption, and so on. A portable device provided with a coin-size battery is usually suitable for a slave device.

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 from "transmitter" or "receiver" functionality.

2.5.1.2 Generic attribute profile (GATT)

The Bluetooth SIG GATT specification provides standard profiles to ensure interoperability between devices from different vendors that implement features like Proximity Profile, Glucose Profile and Health Thermometer Profile. The Bluetooth specification also lets you add custom profiles for new features.

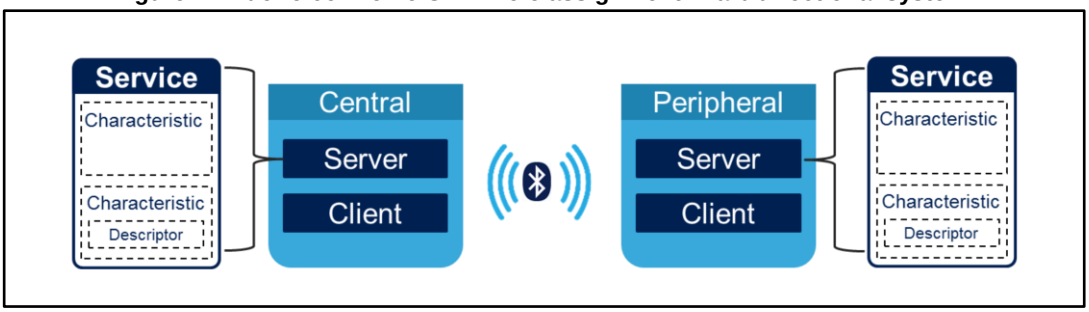
GATT defines client and server roles for interacting devices that are independent of the GAP master/central and slave/peripheral roles:

- **Client** performs service discovery regarding the presence and nature of server attributes; tt 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 sends 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

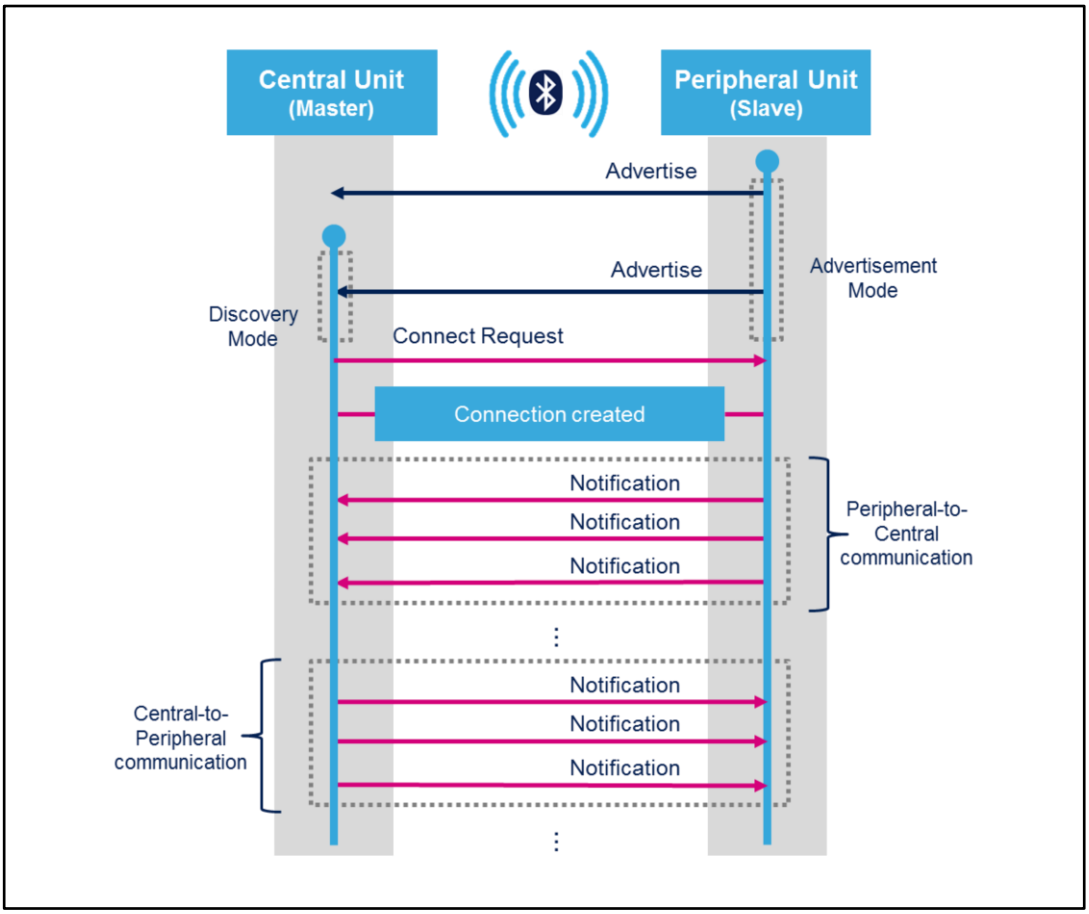


In both directions, streaming audio data transmission is based on periodic server-to-client notifications which do not require a request or response from the receiving device. Server-initiated updates are sent as asynchronous notification packets which include the handle of a characteristic value attribute 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. Following connection, notifications carrying audio data are periodically sent from server to client, according to the selected peripheral-to-central or central-to-peripheral direction.

Figure 5: BLE connection creation



2.5.1.4 BlueVoice service

The Attribute Protocol (ATT) is used by GATT as the transport protocol for exchanging data between 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 (this section is taken from [*4*], which includes an in-depth description of the heart rate service):

- **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's 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.

They are structured thus:

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 OSXBLUEVOICE 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 is compressed via ADPCM (adaptive differential pulse code modulation) to fit in the available data rate while minimizing radio transmission time and power consumption. On 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.

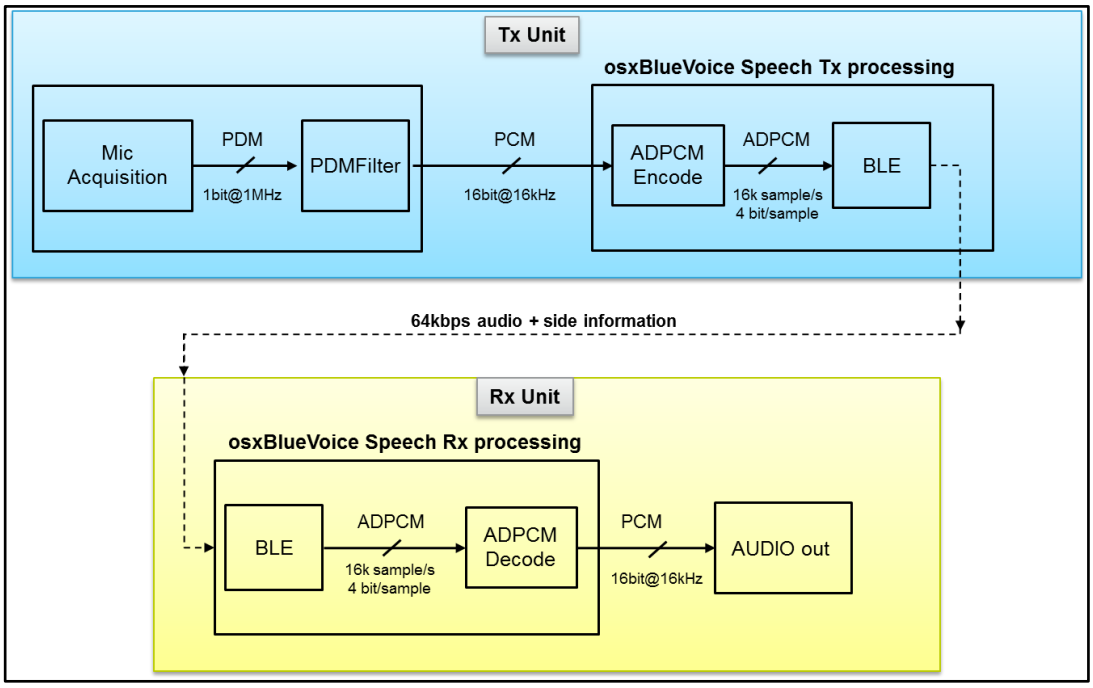
Figure 6: "BLUEVOICELINK1 chain1 – peripheral to central communication" shows the speech processing chain in a complete communication system with Tx and Rx.

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. On a half-duplex implementation, the same processing chain is implemented and the same bandwidth is used for communication in the opposite direction.

Figure 6: BLUEVOICELINK1 chain1 – peripheral to central communication



2.5.2.1 PDM filter

The digital MEMS microphone is designed to output 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's and full negative waveforms are all 0's.

Pulse code modulation (PCM) is chosen as an intermediate step between PDM and compressed audio data that will be actually sent over BLE. In fact, PCM is a representation method widely adopted in many commercial systems. To convert the PDM stream to PCM data, decimation filters are tipically used: in this application the conversion is performed exploiting a conversion library named PDMLib. It consists of two parts, 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*].

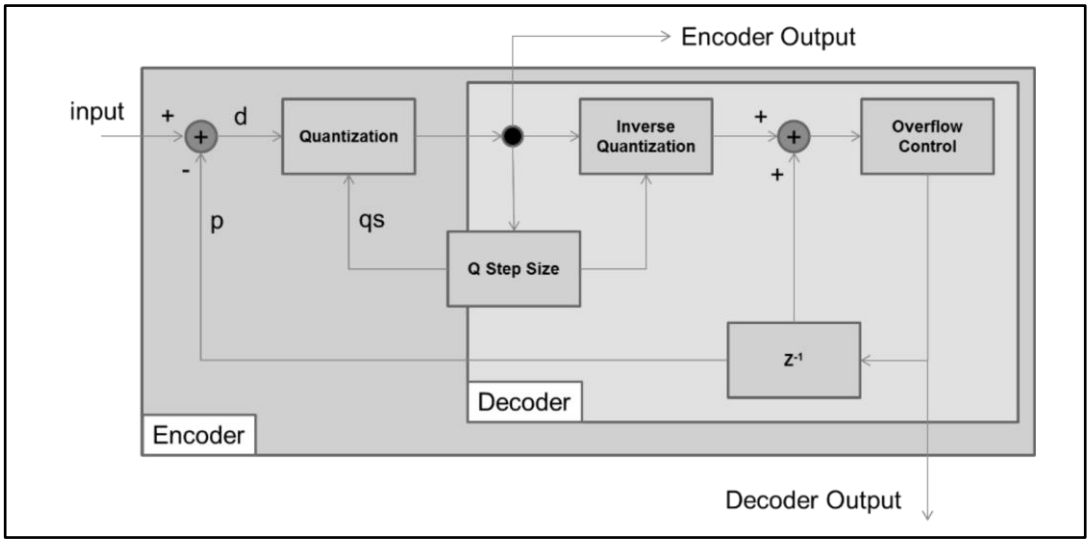
2.5.2.2 ADPCM compression

The ITU-T G.726 adaptive differential pulse code modulation (ADPCM) standard is applied to save bandwidth. This audio algorithm for lossy 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 OSXBLUEVOICE application compresses digital voice signals encoded as:

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

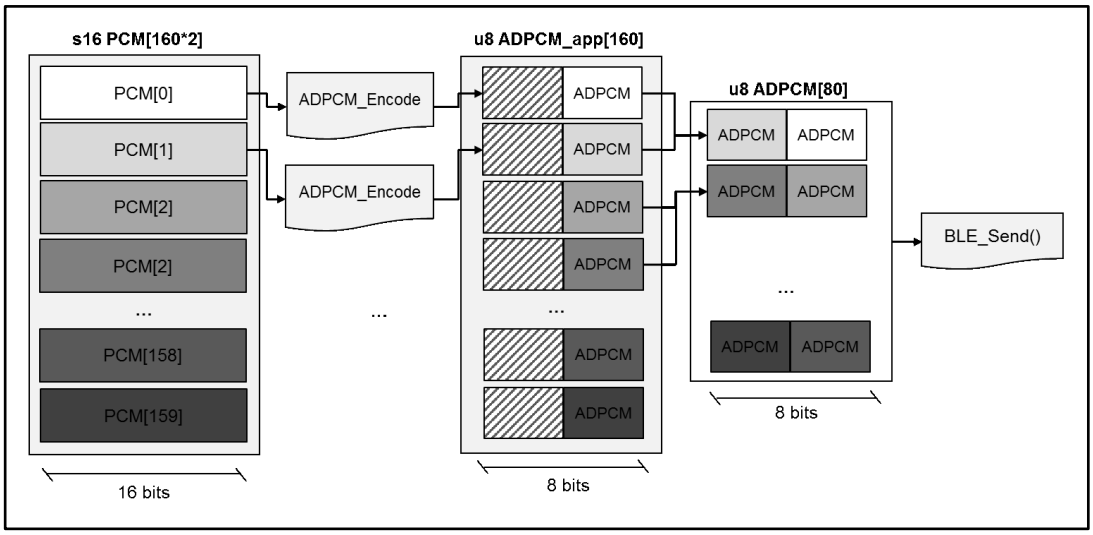
Figure 7: ADPCM encode-decode schema



OSXBLUEVOICE 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 fashion.

2.5.3 Packetization

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 BLUEVOICELINK1 application implements:

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

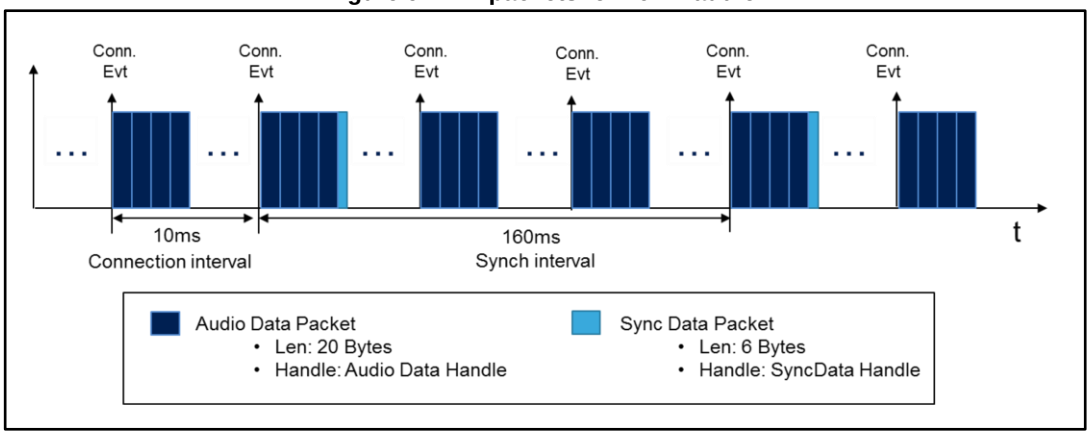
With a 16 kHz audio sampling, the following connection intervals are set for Android™ and iOS™ compliance:

- conn_min_int=10 ms
- conn_max_int=21.25 ms

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

2.6.1 Overview

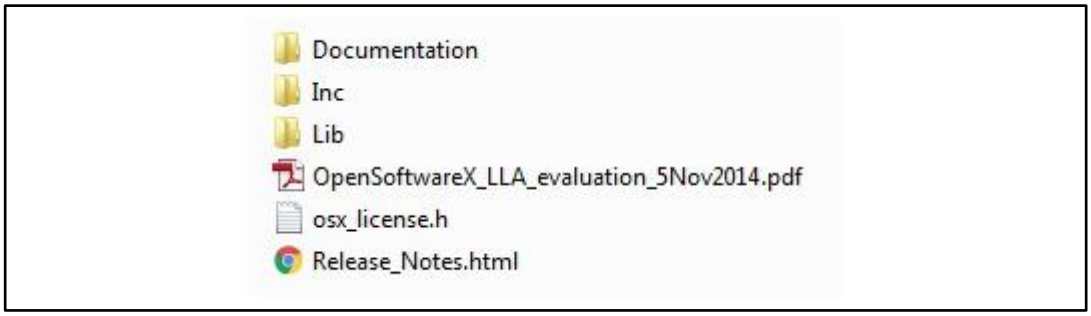
OsxBlueVoice implements a vendor-specific profile that expands on the standard Bluetooth low energy profiles already supported by BlueNRG. The OSXBLUEVOICE engine is provided as a node-locked library which allows derivative firmware images to run on a specific STM32 device only. Licensing activation codes must to be requested from ST and included in the project (and will become part of the build process) prior to attempting its usage. The resulting firmware binary image will therefore be node-locked.

The library is implemented as middleware ready to be integrated in projects based on STM32Cube and the Bluetooth low energy BlueNRG network module. Specifically, OSXBLUEVOICE 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_OSX_BlueVoice_Library in the Middlewares folder is shown below.

Figure 10: Package 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 OSXBLUEVOICE library

- **Lib**: contains the OSXBLUEVOICE library binary code compiled for the three different tool-chains
- **OpenSoftwareX_LLA**: limited license agreement document
- **osx_license.h**: the license number received from a license request must be pasted into this file

2.6.3 Using the OSXBLUEVOICE library

A typical flow of OSXBLUEVOICE library operation can be seen by analyzing the BLUEVOICELINK1 sample firmware.

2.6.3.1 Initialization and configuration

First call the `osx_BlueVoice_Initialize` API to check if the license number copied in the osx_license.h file is correct. If the return value is 0, the library is unlocked and ready to use.

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

```
OSX_BLUEVOICE_Config_t OSX_BLUEVOICE_Config;
OSX_BLUEVOICE_Config.sampling_frequency = FR_16000;
OSX_BLUEVOICE_Config.channel_in = 1;
OSX_BLUEVOICE_Config.channel_tot = 2;
osx_BlueVoice_SetConfig(&OSX_BLUEVOICE_Config);
```

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

The OSXBLUEVOICE can be reset by recalling `osx_BlueVoice_SetConfig`.

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

- `osx_BlueVoice_ConnectionComplete_CB` must be called when an EVT_LE_CONN_COMPLETE event occurs; it sets the connection handle.
- `osx_BlueVoice_DisconnectionComplete_CB` must be called when an EVT_DISCONN_COMPLETE event occurs; it resets internal parameters.
- `osx_BlueVoice_AttributeModified_CB` must be called when an EVT_BLUE_GATT_ATTRIBUTE_MODIFIED event occurs; needed when an enable notification is received, it sets the working mode accordingly.

A timeout has been included for correct management of the BlueVoice profile status. When the module is no longer streaming or receiving, the profile status switches to OSX_BLUEVOICE_STATUS_READY as soon as the timer expires. The duration (in ms) of this timeout is defined by the OSX_BLUEVOICE_TIMEOUT_STATUS. To increase this timer, call the `osx_BlueVoice_IncTick` every 1 ms from 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 the `osx_BlueVoice_AddService` and `osx_BlueVoice_AddChar` functions. Both the APIs require the UUIDs (chosen by the user) as parameters, and they return the relevant

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

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

If both the functions return OSX_BV_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 structure of the type `OSX_BLUEVOICE_ProfileHandle_t` to the `osx_BlueVoice_SetTxHandle` API. In the latter case, the following characteristics must be created:

```
aci gatt add char(ServiceHandle,
              UUID_TYPE_128, CharAudioUUID, 20,
              CHAR_PROP_NOTIFY, ATTR_PERMISSION_NONE,
              GATT_DONT_NOTIFY_EVENTS, 16, 1,
              CharAudioHandle);

aci gatt add char(ServiceHandle,
              UUID_TYPE_128, CharAudioSyncUUID, 6,
              CHAR_PROP_NOTIFY, ATTR_PERMISSION_NONE,
              GATT_DONT_NOTIFY_EVENTS, 16, 1,
              CharAudioSyncHandle);
```

The first relates to the compressed audio data; the second sends collateral information used to implement a synchronization mechanism.

You can choose to not create the BlueVoice Service and the module will not be 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 `osx_BlueVoice_EnableNotification` function and setting the handle of the BlueVoice service and characteristics exported by the transmitter module through the `osx_BlueVoice_SetRxHandle` API.

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

2.6.3.3 Audio signal injection

The OSXBLUEVOICE library receives audio PCM input samples. The `osx_BlueVoice_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 an OSX_BV_PCM_SAMPLES_ERR is returned.

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

2.6.3.4 Compressed audio streaming

Regarding the transmitter part, the OSXBLUEVOICE library gathers compressed data in an internal double buffer. For every 10 ms of audio, the `osx_BlueVoice_AudioIn` function returns OSX_BV_OUT_BUF_READY to signal that output data can be send via BLE by calling the `osx_BlueVoice_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 from a notification through a EVT_BLUE_GATT_NOTIFICATION event and passed to the `osx_BlueVoice_ParseData` function that decompresses the data and returns a PCM buffer. This API is used to parse both audio and collateral information data.

2.7 BLUEVOICELINK1 application description

The BLUEVOICELINK1 central and peripheral applications using the X-NUCLEO-CCA02M1 and X-NUCLEO-IDB04A1 or X-NUCLEO-IDB05A1 expansion boards with the NUCLEO-F401RE or NUCLEO-L476RG board are provided in the “Projects” directory. Ready-to-build projects are available for multiple IDEs.

In order to demonstrate 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:

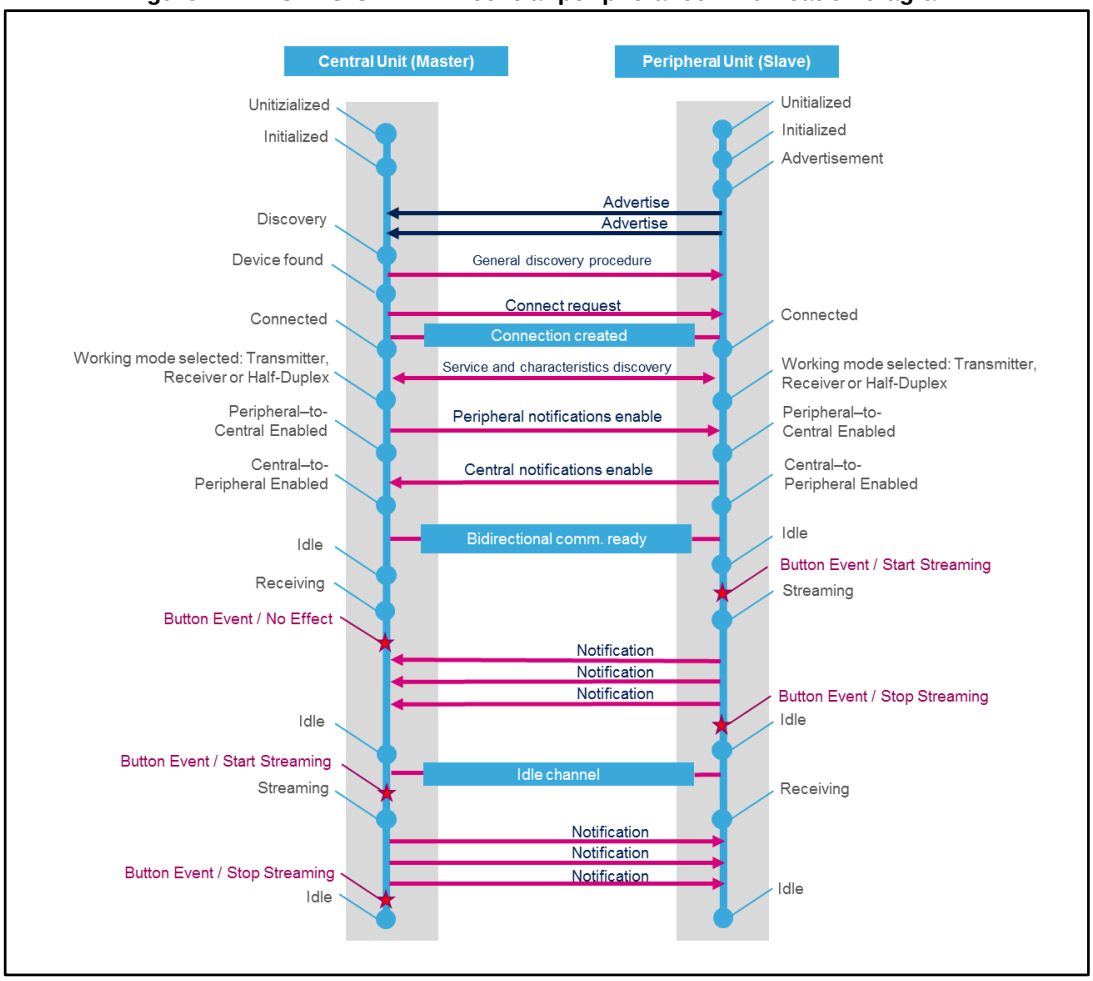
- when a module is streaming, the application handles audio acquisition, data compression and packetization of the ADPCM compressed audio to be streamed over BLE, according to the BlueVoice Profile specification.
- when a module is receiving, the application is responsible for the decompression of ADPCM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

Following connection, the active communication channel is controlled by the STM32 Nucleo user button on the two modules: *Figure 11: "BLUEVOICELINK1: 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 STM32 Nucleo 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, LED2 on the STM32 Nucleo board signals the following:

- initialized: slow blinking
- connected: normal blinking
- streaming: rapid blinking
- receiving: steady on (not blinking)

Figure 11: BLUEVOICELINK1: central-peripheral communication diagram



2.7.1 BLUEVOICELINK1 implementation

In this section, implementation of the BLUEVOICELINK1 application is briefly described:

- Initialization
  - Audio acquisition initialization
    - PDM to PCM middleware is initialized
    - The required MCU peripherals are set up making use of the dedicated BSP function.
  - Audio IN USB driver initialization
    - The related descriptor is configured in order to fit the single channel at 16 kHz configuration.
    - After the configuration phase, the device streams signals to a host device as a standard single channel USB microphone
  - BlueNRG component and BLE middleware initialization
  - OSXBLUEVOICE library initialization
    - BLE role specification as central or peripheral.
    - The central and peripheral address must be passed to the library in order to build a point-to-point connection.
- Running
  - HCI_Process: running of the BLE middleware engine.

- Application process
  - Status control: read 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 notifications mechanism from the other module is not enabled yet.
    - Central to peripheral enabled: Peripheral device has enabled notifications from central device.
    - Peripheral to central enabled: Central device has enabled notifications from 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.
  - OSXBLUEVOICE StateMachine: responsible for internal status update and data processing.

2.7.1.1 Streaming state

During the initialization phase, BSP functions of X-NUCLEO-CCA02M1 are used to set up DMA for audio acquisition via I^2S of the PDM signal of the two on-board digital MEMS microphones with a sample frequency of 1.024 MHz. In streaming mode, the acquisition engine is turned on: each 1 ms, the I^2S 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. The resulting 16 samples/ms are passed to the OSXBLUEVOICE library, which internally implements a double buffering mechanism to collect 10 ms of audio that will then be encoded in ADPCM and sent via BLE autonomously, depending on the status of the module.

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 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 passed to the OSXBLUEVOICE 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 via the relevant APIs.

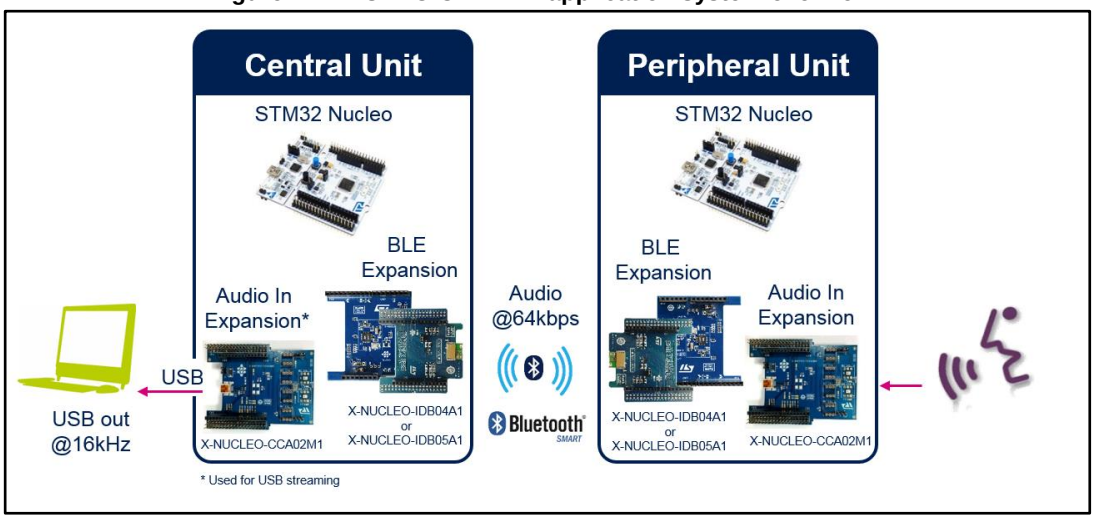
3 System setup guide

3.1 Hardware description

The BLUEVOICELINK1 application is based on the STM32 Nucleo platform, the X-NUCLEO-IDB04A1/X-NUCLEO-IDB05A1 expansion board and the X-NUCLEO-CCA02M1 microphone acquisition expansion board. *Figure 12: "BLUEVOICELINK1 application system overview"* shows the system overview. The peripheral unit acts as transmitter while the central unit acts as receiver of the audio stream.

The half-duplex demo can be setup using a NUCLEO-F401RE or NUCLEO-L476RG development board for both the central and the peripheral modules. A NUCLEO-L053R8 board can be used as the peripheral for the BlueMS Android/iOS app.

Figure 12: BLUEVOICELINK1 application system overview

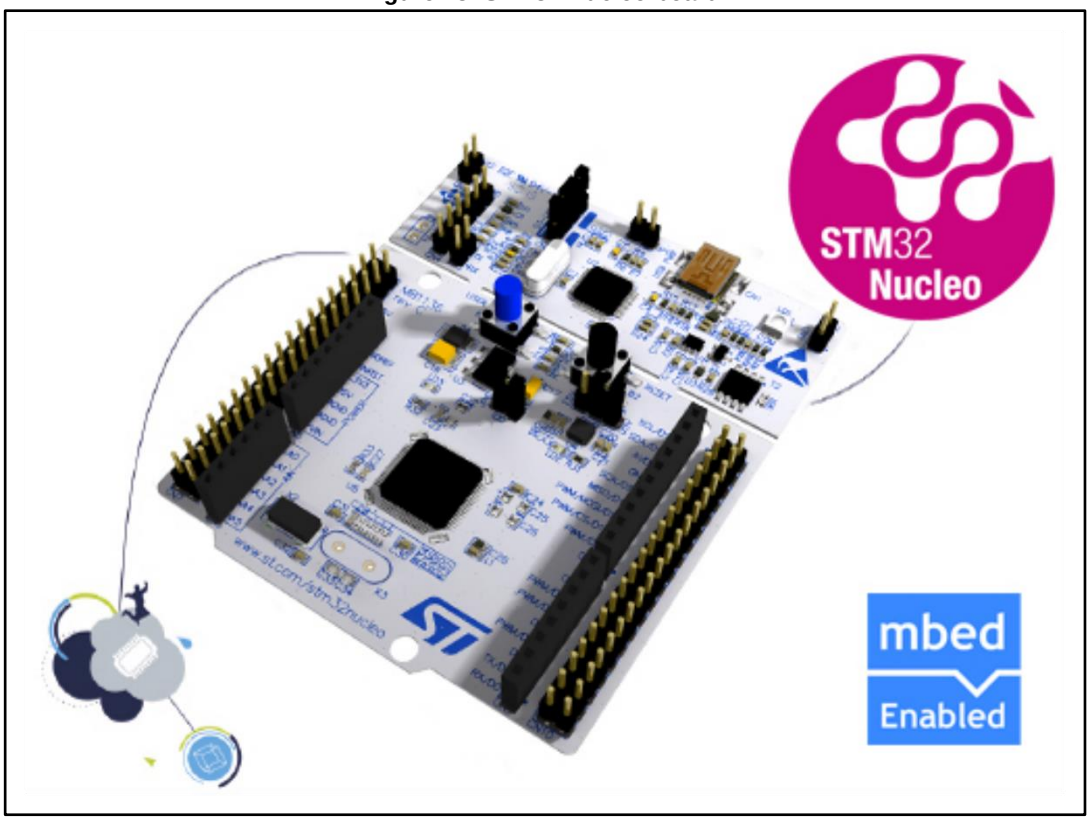


3.1.1 STM32 Nucleo platform

The STM32 Nucleo boards provide an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller lines. The Arduino™ connectivity support and ST morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from. The STM32 Nucleo board does not require any separate probe 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.

Information regarding the STM32 Nucleo board is available on www.st.com at http://www.st.com/stm32nucleo

Figure 13: STM32 Nucleo board



3.1.2 X-NUCLEO-CCA02M1 expansion board

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

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

Figure 14: X-NUCLEO-CCA02M1 expansion board

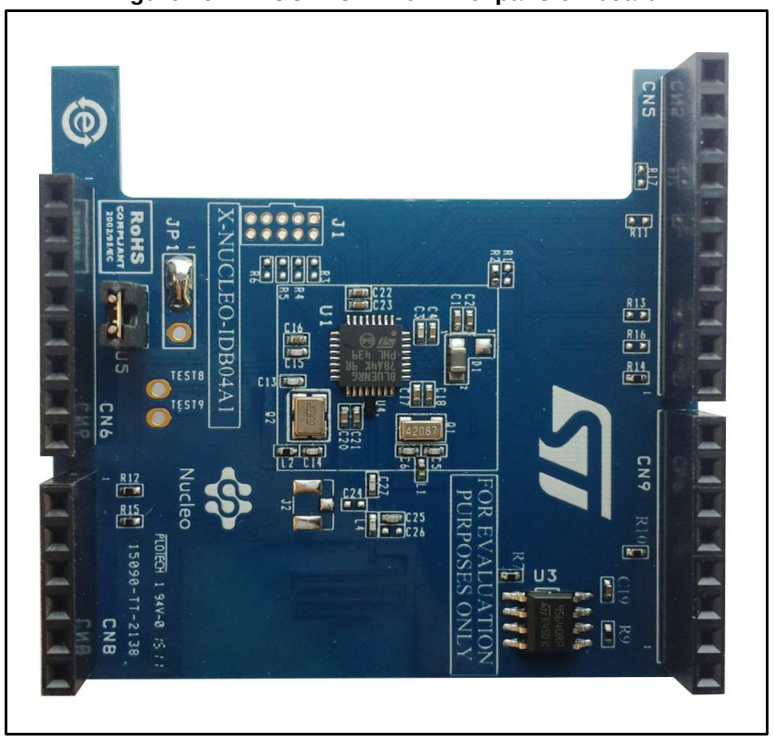
Information regarding the X-NUCLEO-CCA02M1 expansion board is available on www.st.com at *http://www.st.com/x-nucleo*.

3.1.3 -NUCLEO-IDB04A1 or X-NUCLEO-IDB05A1 expansion board

3.1.3.1 X-NUCLEO-IDB04A1 expansion board

The X-NUCLEO-IDB04A1 is a Bluetooth BlueNRG expansion board usable with the STM32 Nucleo system. The BlueNRG is a very low power Bluetooth low energy (BLE) single-mode network processor, compliant with Bluetooth specifications core 4.0.

Figure 15: X-NUCLEO-IDB04A1 expansion board

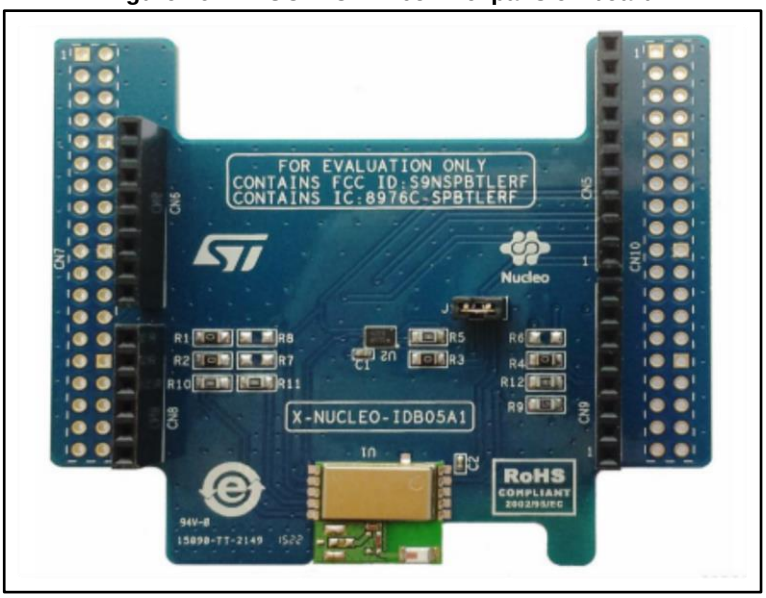


Information regarding the X-NUCLEO-IDB04A1 expansion board is available on www.st.com at *http://www.st.com/x-nucleo*.

3.1.3.2 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 16: 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*

3.2 Software description

The following software components are required to set up a suitable development environment for creating applications for the STM32 Nucleo equipped with the BlueNRG/BlueNRG-MS and digital MEMS microphone expansion boards:

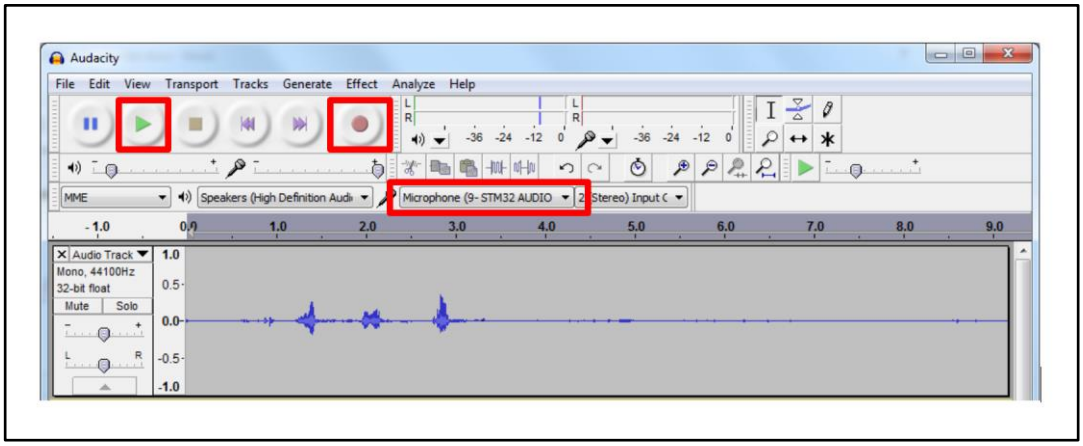
- BLUEVOICELINK1: an application based on STM32Cube that demonstrates voice communication over BLE. The BLUEVOICELINK1 firmware and related documentation is available on www.st.com
- One of the following development environments:
  - IAR Embedded Workbench for ARM® (EWARM) toolchain v7.70.0 + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM) toolchain v5.17.0 + ST-LINK
  - Ac6 System Workbench for STM32 toolchain v1.9.0 + ST-LINK

3.3 PC audio recording utility example: Audacity

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

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

Figure 17: Audacity for Windows



3.4 Hardware and software setup

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

3.4.1 Hardware setup

The following hardware components are needed:

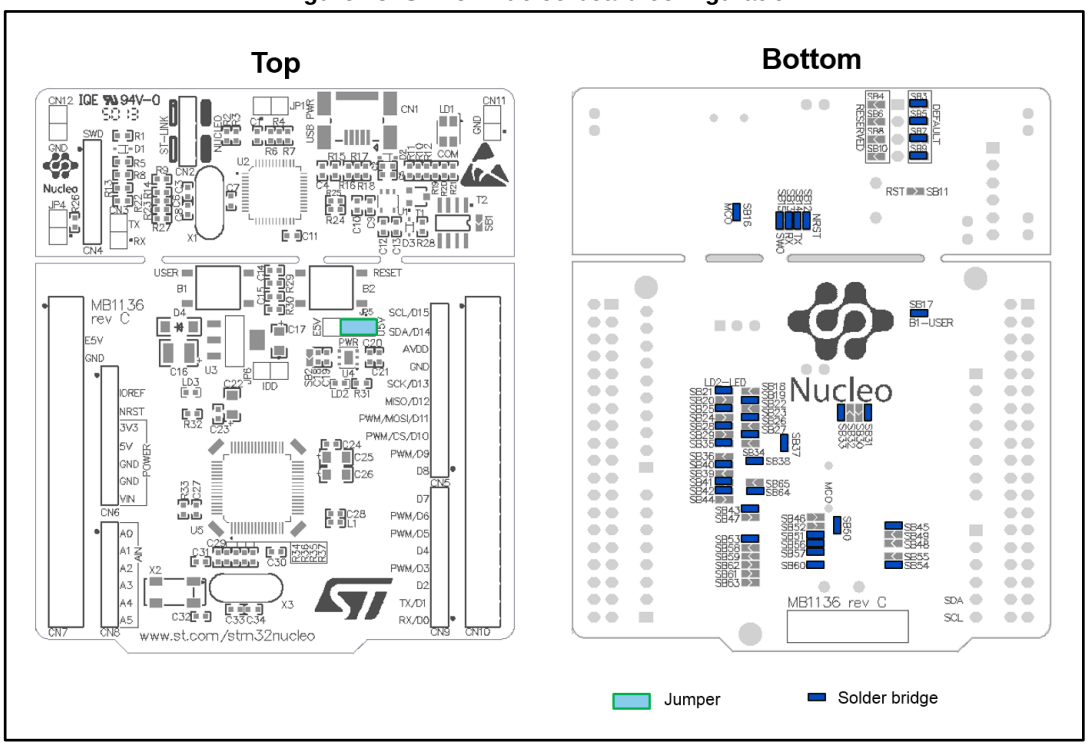
1. For half-duplex communication:
   - two STM32 Nucleo development platforms (order code: NUCLEO-F401RE or NUCLEO-L476RG)
2. For simplex communication to an Android/iOS device:
   - one STM32 Nucleo development platform (order code: NUCLEO-L053R8)
   - an Android (version 4.4 and above) or iOS (version 8 and above) device
3. Two digital MEMS microphones expansion board (order code: X-NUCLEO-CCA02M1).
4. Bluetooth low energy expansion boards:
   - two BlueNRG Bluetooth low energy expansion boards (order code: X-NUCLEO-IDB04A1), updated to BlueNRG FW version 6.4 or later or
   - two BlueNRG-MS Bluetooth low energy expansion boards (order code: X-NUCLEO-IDB05A1), updated to BlueNRG FW version 6.4 or later
5. One USB type A to Mini-B USB cable to power up the Tx module.
6. One USB type A to Mini-B USB cable to power up the Rx module and for USB streaming.

Both the STM32 Nucleo development board and the X-NUCLEO-CCA02M1 expansion board must be configured correctly to run the BLUEVOICELINK1 application.

3.4.1.1 STM32 Nucleo development board

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 PF0/PD0/PH0-OSC_IN of the STM32 microcontroller. Configure the STM32 Nucleo board as shown below.

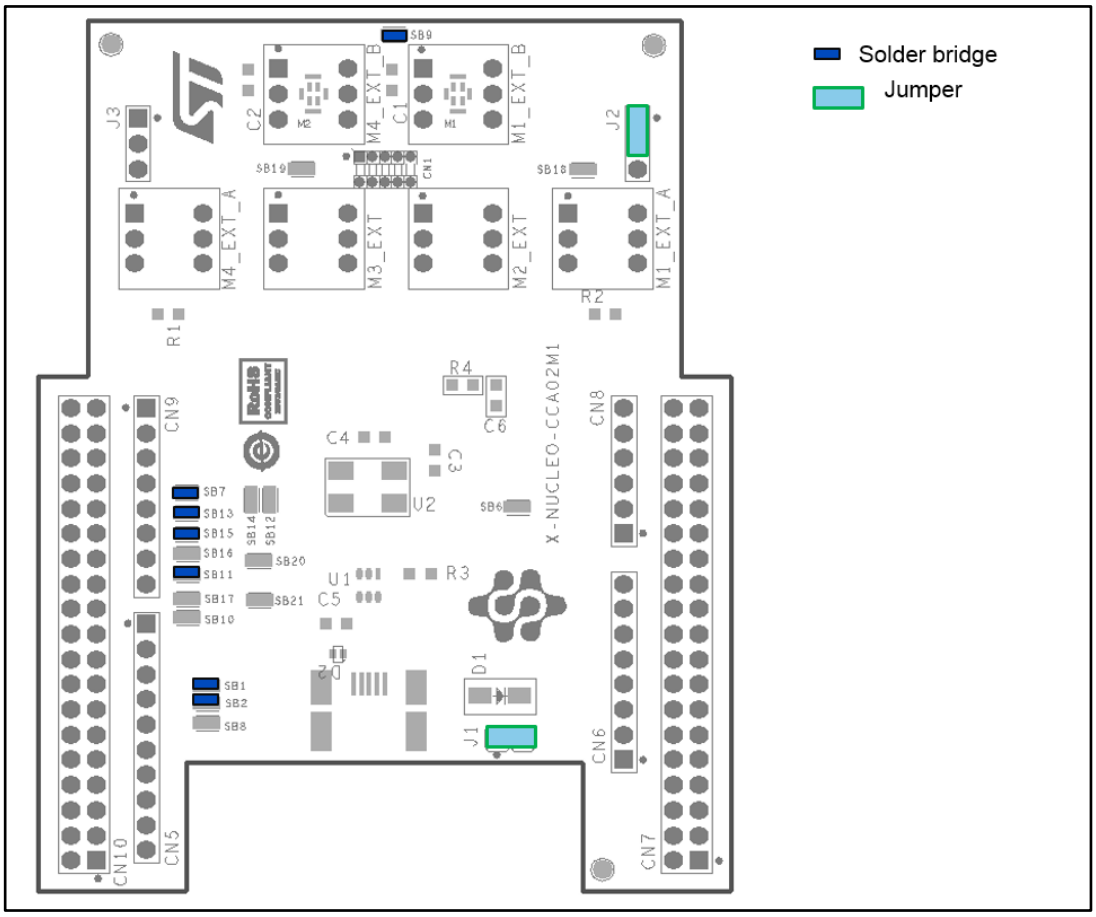
Figure 18: STM32 Nucleo board configuration



3.4.1.2 X-NUCLEO-CCA02M1

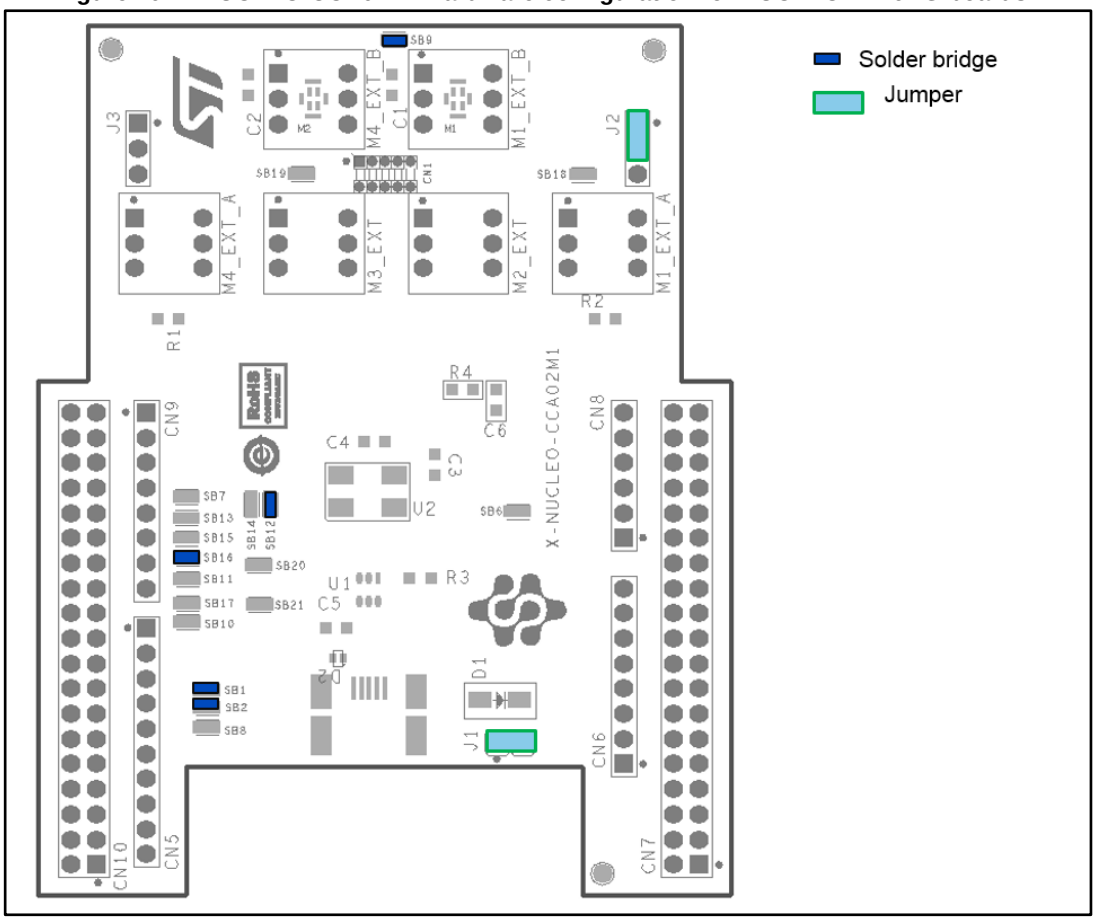
The X-NUCLEO-CCA02M1 board uses different configurations depending on the STM32 Nucleo board connected. If two NUCLEO-F401RE boards are used for half-duplex communication or a NUCLEO-L053R8 board is used for simplex communication with a mobile device, a two-microphone configuration is needed even if only one microphone is actually used by the BLUEVOICELINK1 application.

Figure 19: 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 20: X-NUCLEO-CCA02M1 hardware configuration for NUCLEO-L476RG boards



In order for the device to be recognized as a standard microphone and therefore to stream audio via USB, it needs to be interfaced directly with the USB peripheral of the STM32 provided on the STM32 Nucleo board. Since the STM32 Nucleo USB mini connector is actually connected to a serial port via ST-LINK, the X-NUCLEO-CCA02M1 expansion board embeds a mini USB connector that provides a direct interface to it. This connector must therefore be used in order to stream audio via the STM32 MCU.

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

3.4.2 Half-duplex software setup

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

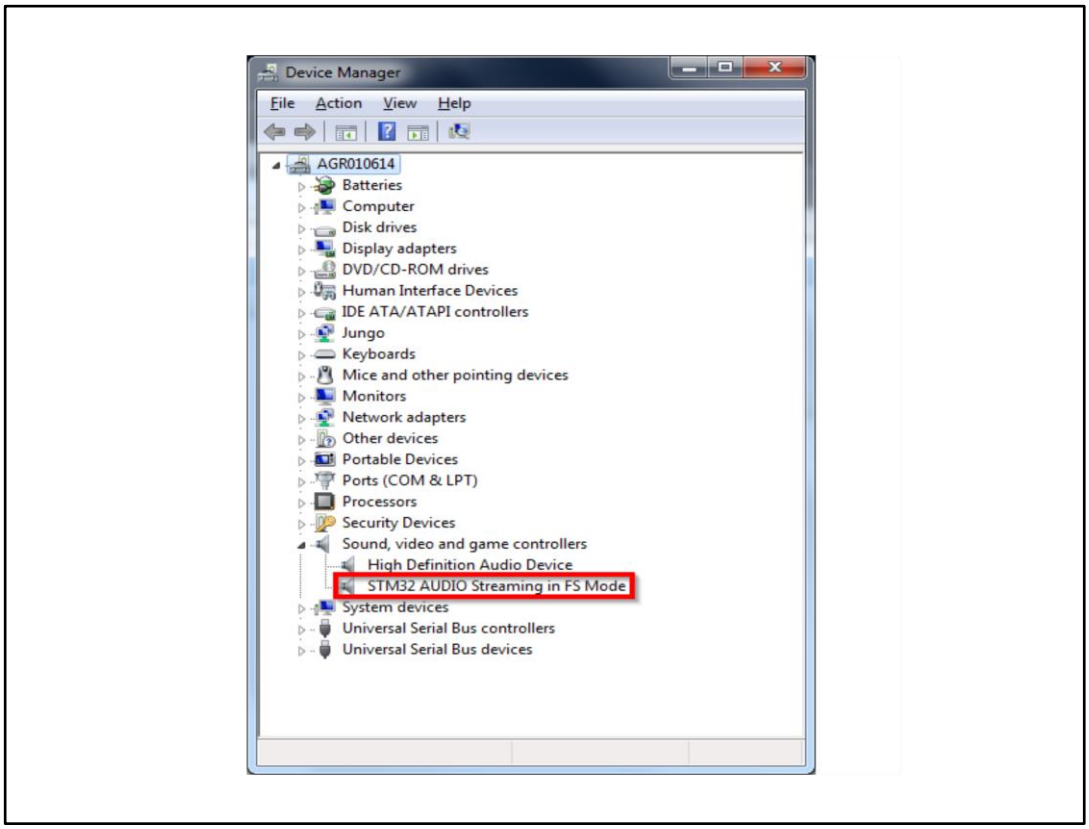
3.4.2.1 Development toolchains and compilers

Select one of the Integrated Development Environments supported by the STM32Cube expansion software. Please read the system requirements and setup information provided by the selected IDE provider.

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

Both central and peripheral applications include an audio input USB driver that allows the device to be recognized as a standard USB microphone. After firmware download to 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 depicted in below.

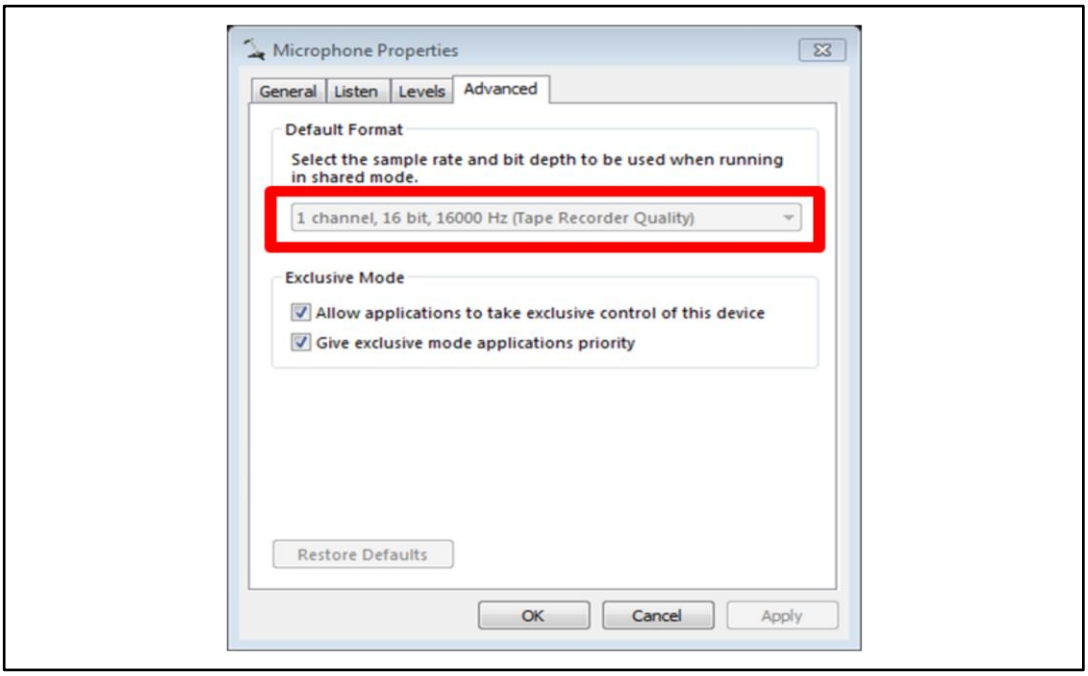
Figure 21: STM32 microphone in Device Manager



Right-click on the volume icon in the Windows task bar (on the bottom right of the screen) and choose “recording device”. Select STM32 microphone and click on Properties. In the Advanced tab, there is a summary of the current device setup in terms of sampling frequency and number of channels. You should see the module recognized as a "1 channel, 16000 Hz" microphone.

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

Figure 22: Microphone properties – Advanced tab



3.4.3 System setup guide

This section describes how to set up different hardware parts before writing and executing an application on the STM32 Nucleo board with the BlueNRG and digital MEMS microphone expansion boards. The following section describes the demo setup in a Windows 7 environment.

3.4.3.1 STM32 Nucleo and expansion board setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. The developer can download the relevant version of the ST-LINK/V2-1 USB driver by searching for STSW-LINK008 or STSW-LINK009 (according to your version of Windows™) on www.st.com.

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

The X-NUCLEO-IDB04A1 BlueNRG expansion board connects easily to the X-NUCLEO-CCA02M1 Arduino UNO R3 extension connector.

3.4.3.2 BLUEVOICELINK1 application setup

This section describes how to set up the BLUEVOICELINK1 half-duplex application to demonstrate audio transmission over Bluetooth low energy. In the BLUEVOICELINK1 application, 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.

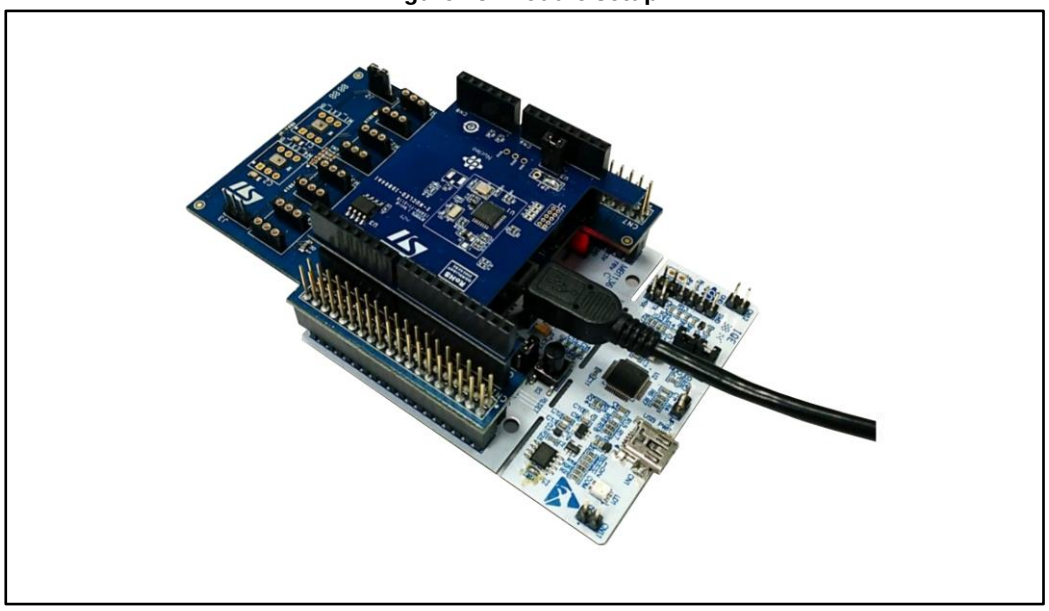
In the following description, a single PC is used to demonstrate the half-duplex functionality, but the same setup can be used for two PCs.

3.4.3.2.1 Module setup

The following steps must be performed:

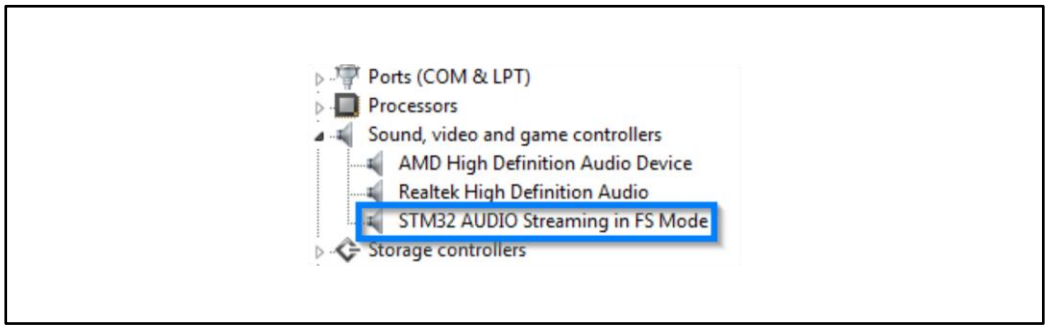
1 Connect the X-NUCLEO-CCA02M1 USB connector of the central unit to the PC.

Figure 23: Module setup



2 The STM32 Audio Streaming peripheral appears in device manager.

Figure 24: STM32 AUDIO Streaming peripheral

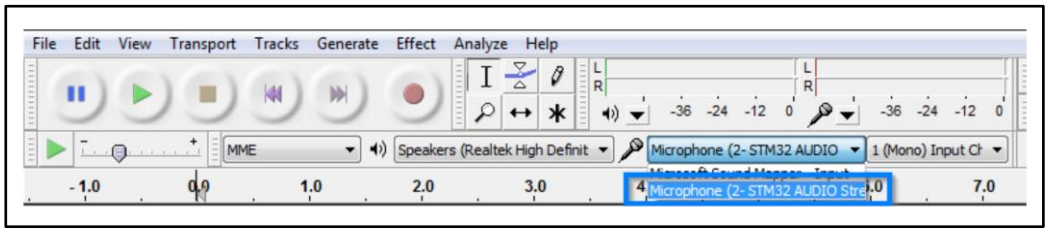


3 Open Audacity

4 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.

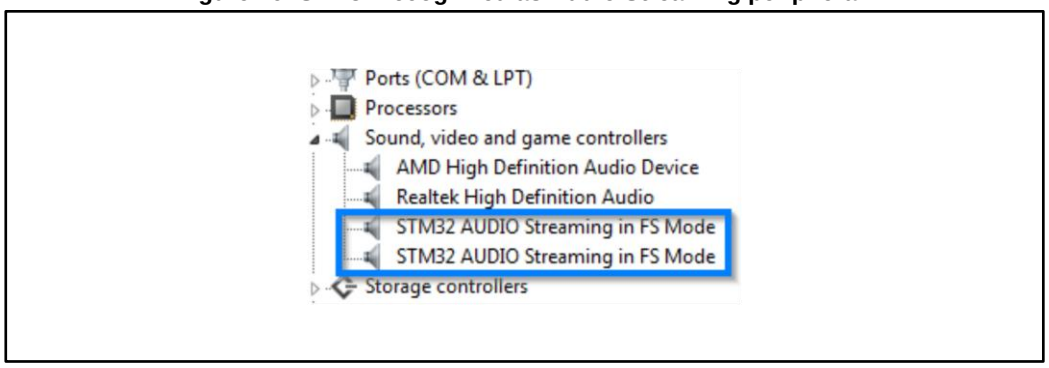
Figure 25: Central unit microphone in Audacity



5 Connect the X-NUCLEO-CCA02M1 USB connector of the peripheral unit to a PC (see *Figure 23: "Module setup"*).

6 A second STM32 AUDIO Streaming peripheral appears in Device Manager.

Figure 26: STM32 recognized as Audio Streaming peripheral

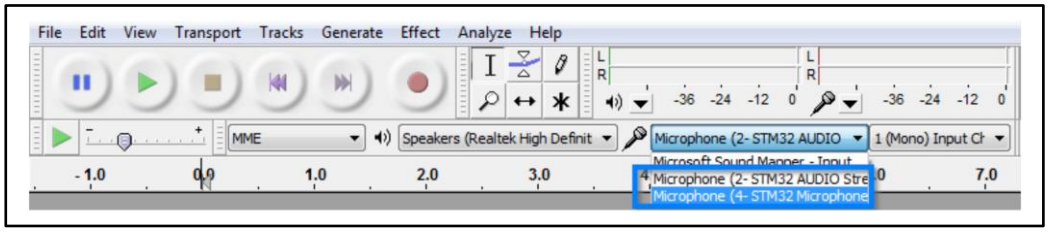


7 In Audacity, click on Transport>Rescan Audio Devices

8 Two STM32 microphones should appear in the Input selector

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

Figure 27: Peripheral unit microphone in Audacity

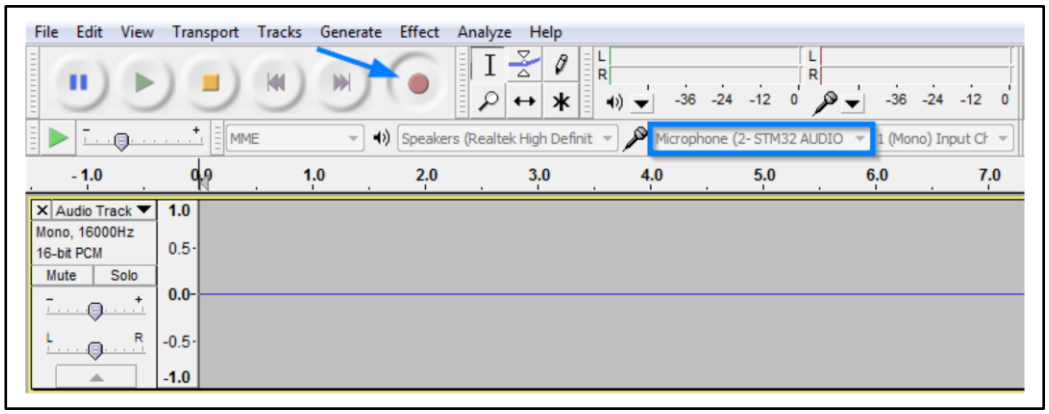


3.4.3.2.2 Peripheral to central recording

1 Choose the Central unit microphone (microphone 2)

2 Click Record to start silent recording.

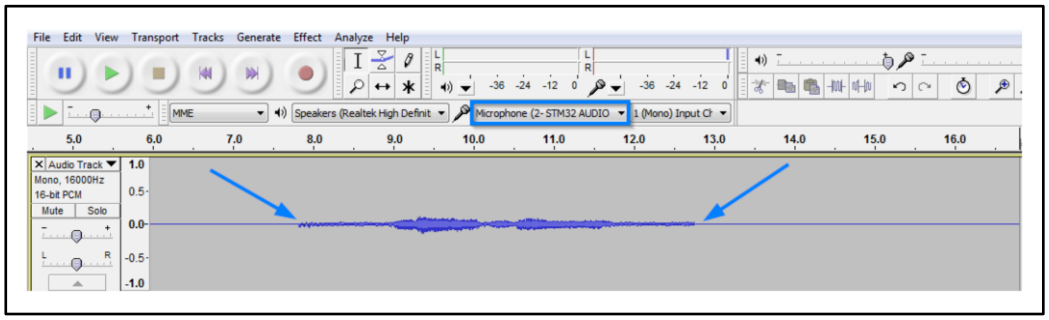
Figure 28: Audacity recording silence from central unit USB stream



3 Click Peripheral unit user button

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

Figure 29: Audacity recording voice coming from peripheral unit



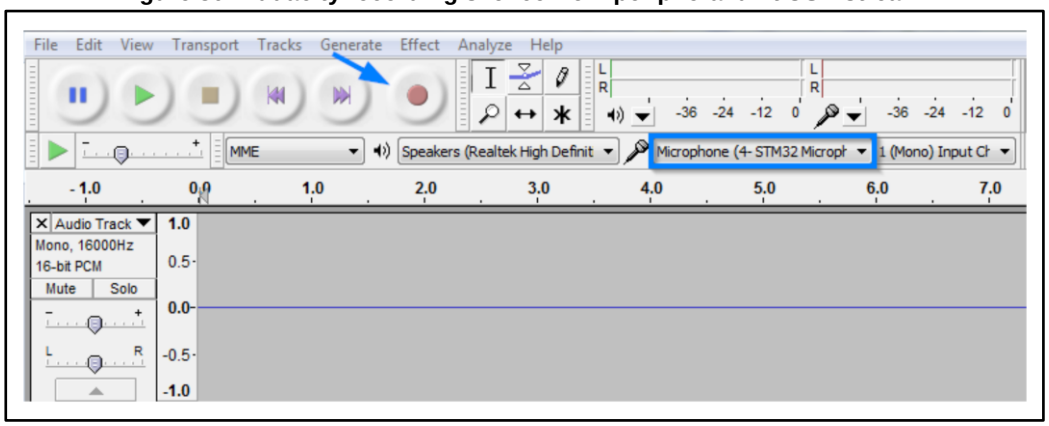
4 Clicking the button toggles the streaming status.

3.4.3.2.3 Central to peripheral recording

1 Choose the Peripheral unit microphone (microphone 4).

2 Click Record to start silent recording.

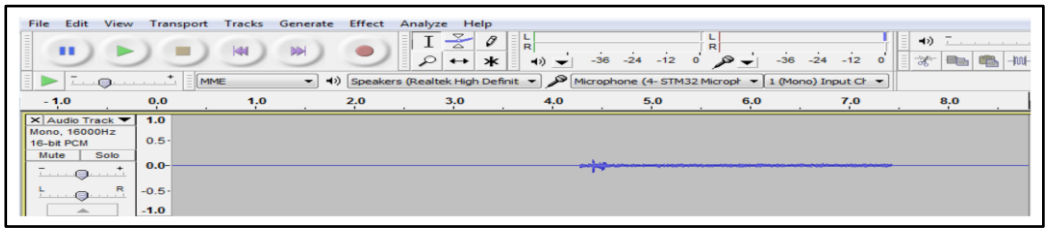
Figure 30: Audacity recording silence from peripheral unit USB stream



3 Click central unit user button

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

Figure 31: Audacity recording voice coming from Central unit



4 Clicking the button toggles the streaming status.

4 BLUEVOICELINK1 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 STM32 to Android/iOS. In this scenario the ST module acts as a Peripheral node, while the smartphone/tablet is the Central node. You can use the Peripheral module of the BLUEVOICELINK1 demo to stream audio acquired at 8 kHz and compressed via ADPCM to the mobile device.

4.1 Hardware setup

Referring to *Section 3.4.1: "Hardware setup"*, the peripheral module can be used in to set up the Android/iOS demo.

The setup requires an X-NUCLEO-IDB04A1 or X-NUCLEO-IDB05A1 expansion board plugged on an X-NUCLEO-CCA02M1 expansion board, plugged on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L053R8 development board.

Move jumper JP5 to “U5V” position and power the system through the mini USB port on the Nucleo board.

4.2 Software setup

4.2.1 Peripheral firmware

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

In order to configure the peripheral module accordingly, open the BlueVoiceLink Peripheral firmware with one of the available toolchains and modify the source code thus:

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

The modified firmware must be recompiled and flashed on the peripheral module. The same OSXBLUEVOICE license can be used for both the 8 kHz and 16 kHz version.

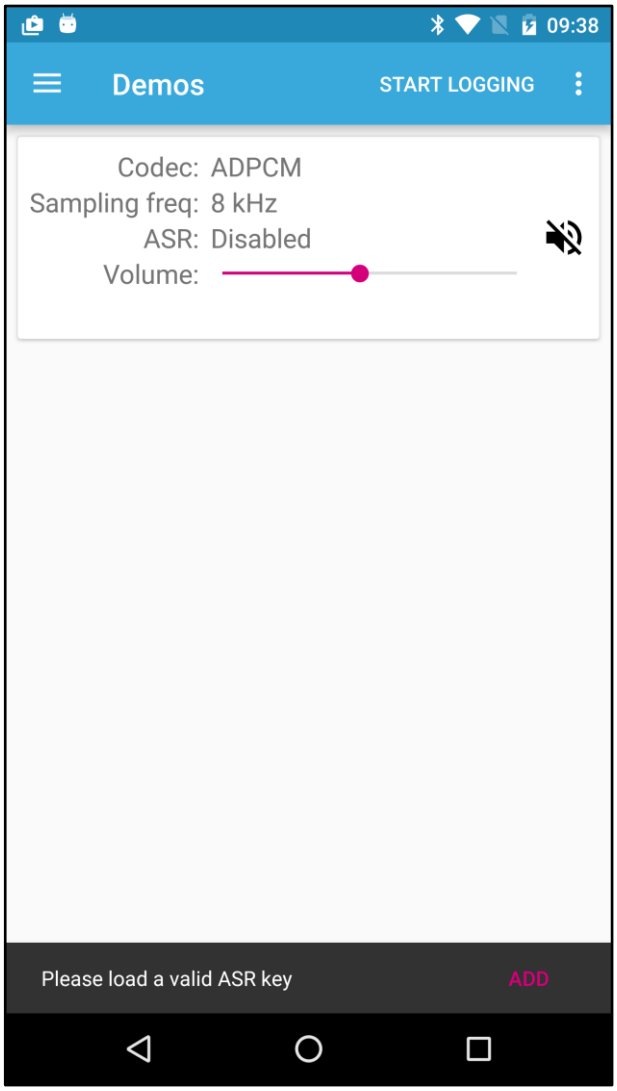
4.2.2 ST BlueMS app

BLUEVOICELINK1 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 BlueMicrosystem2 User Manual on www.st.com.

If the OSXBLUEVOICE voice over BLE library (BlueVoice 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 (only English recognition).

Figure 32: BlueMS (Android version) OSXBLUEVOICE start page

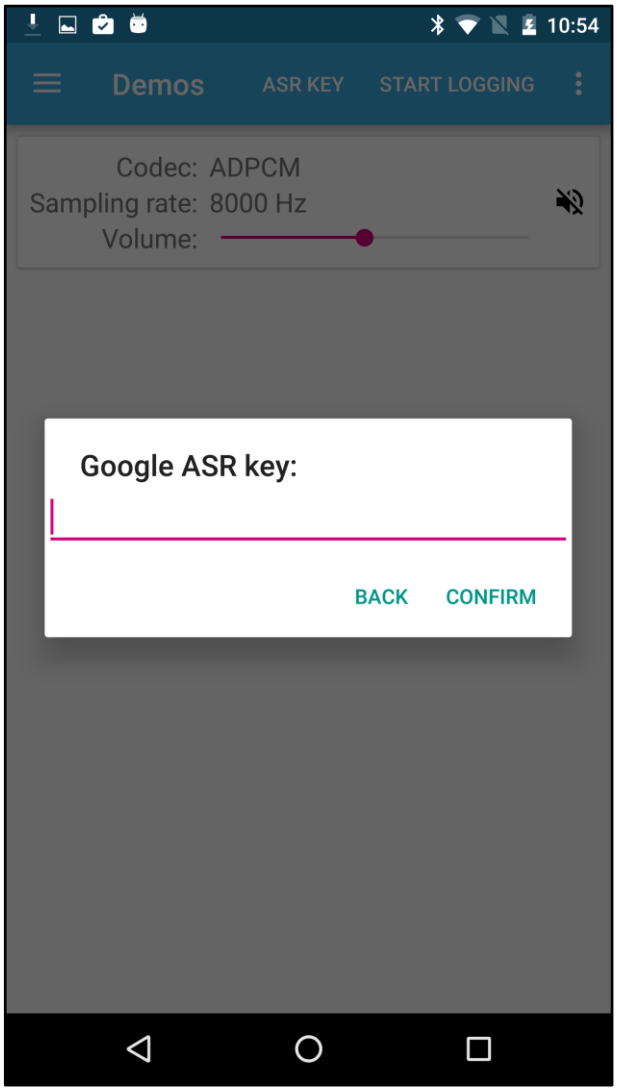


The audio playback begins as soon as the blue button on the Nucleo is pressed (the green led starts blinking faster). The volume can be adjusted using the slider or muted by clicking on the speaker icon.

The ADD button at the bottom of this page allows the insertion of the key (see *Section 4.2.2.1: "Google speech ASR Key generation"*) to enable the ASR feature.

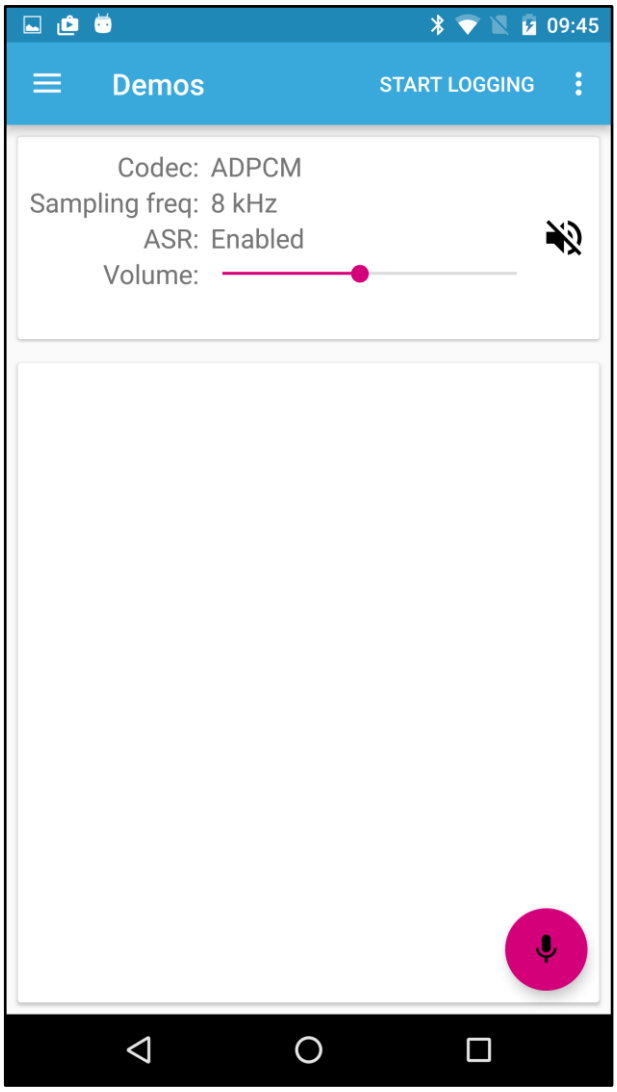
After pressing this button, a popup window prompts the insertion of a valid API key, followed by the ASR service activation key.

Figure 33: BlueMS (Android version) OSXBLUEVOICE popup API key window



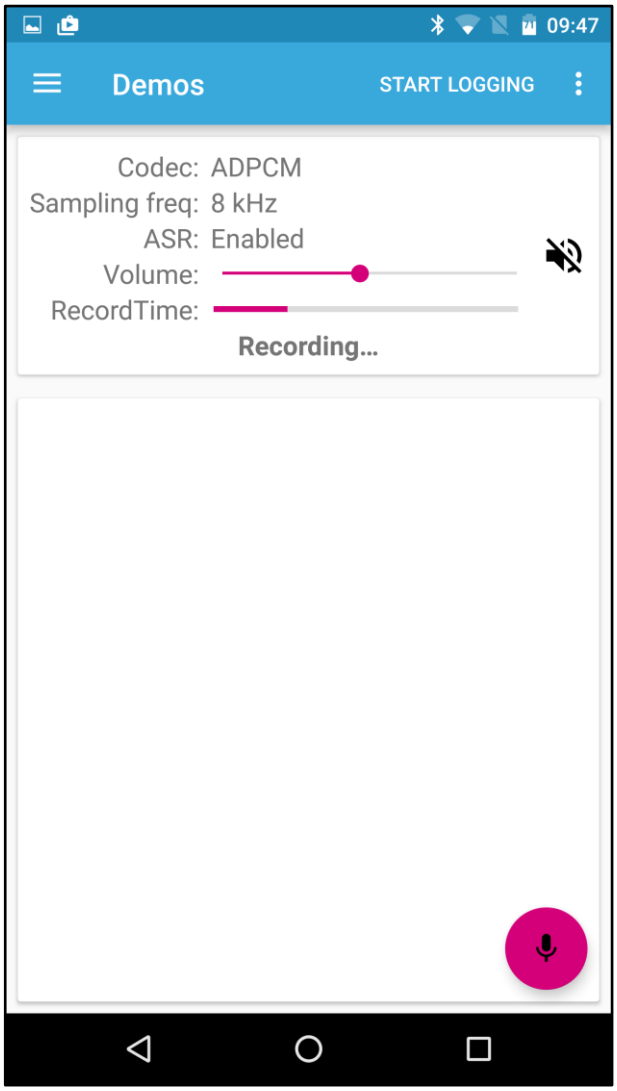
Once the key is correctly inserted, the start screen changes as shown below.

Figure 34: BlueMS (Android version) OSXBLUEVOICE ASR service enabled



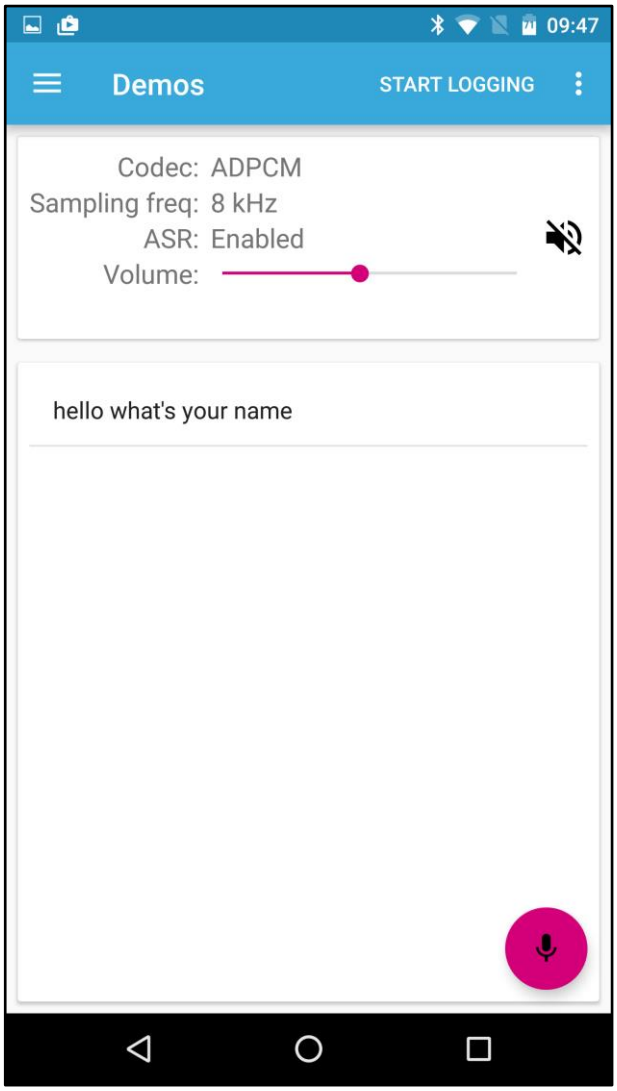
Hold the recording button to register your voice for subsequent recognition. While the button is pressed, a bar progressively indicates the elapsed recording time.

Figure 35: BlueMS (Android version) OSXBLUEVOICE voice recording



When you release the button a “Sending request…” message appears. Eventually, the speech recognized by the ASR service appears below the volume bar.

Figure 36: BlueMS (Android version) OSXBLUEVOICE recognised voice text



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

4.2.2.1 Google speech ASR Key generation

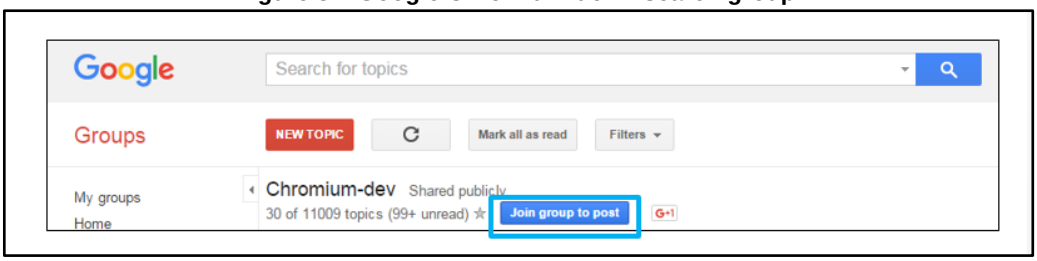
The Google Speech APIs require a key to access the web-based service. You need a Google account to complete the procedure and to access the service.

To generate a key:

1 Login with your own Gmail account.

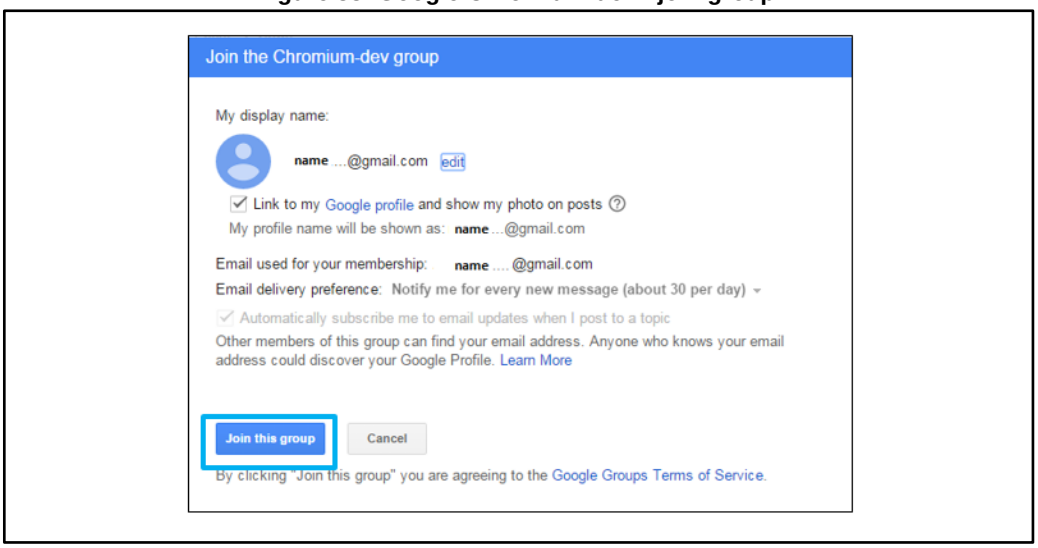
2 Subscribe to Chromium-dev at *https://groups.google.com/a/chromium.org/forum/?fromgroups#!forum/chromium-dev*. Write “Chromium-dev” in the search box, and select the appropriate group.

Figure 37: Google Chromium-dev – search group



3 Click on “Join group to post” button

Figure 38: Google Chromium-dev - join group



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

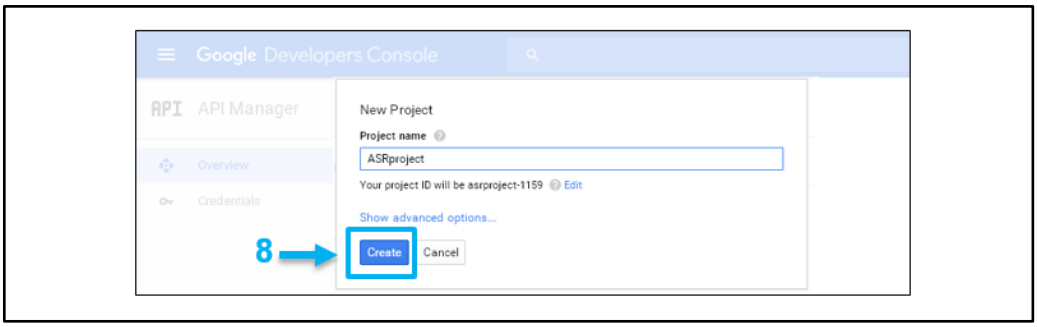
Figure 39: Google Chromium-dev - join group confirmation

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

6 Open the "Select a project” menu in the upper toolbar.

7 Click on “Create a project…”

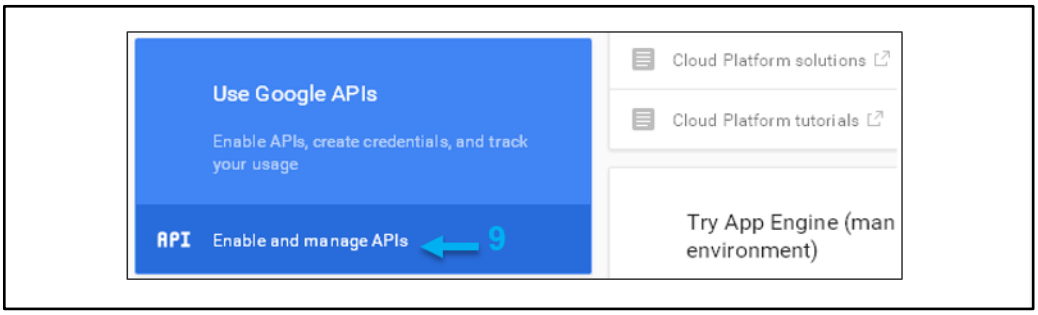
Figure 40: Google Developers Console – new project



8 Choose the Project name.

9 Click on “Create” button.

Figure 41: Google Developers Console - create project



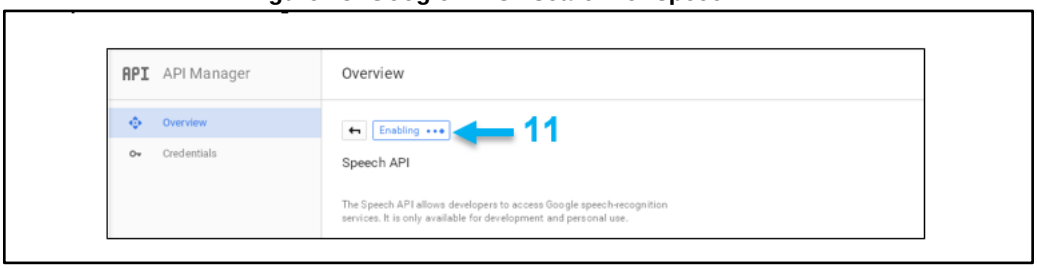
10 Open the project you’ve created.

11 In the dashboard, click on “Enable and manage APIs”.

Figure 42: Google APIs – enable APIs

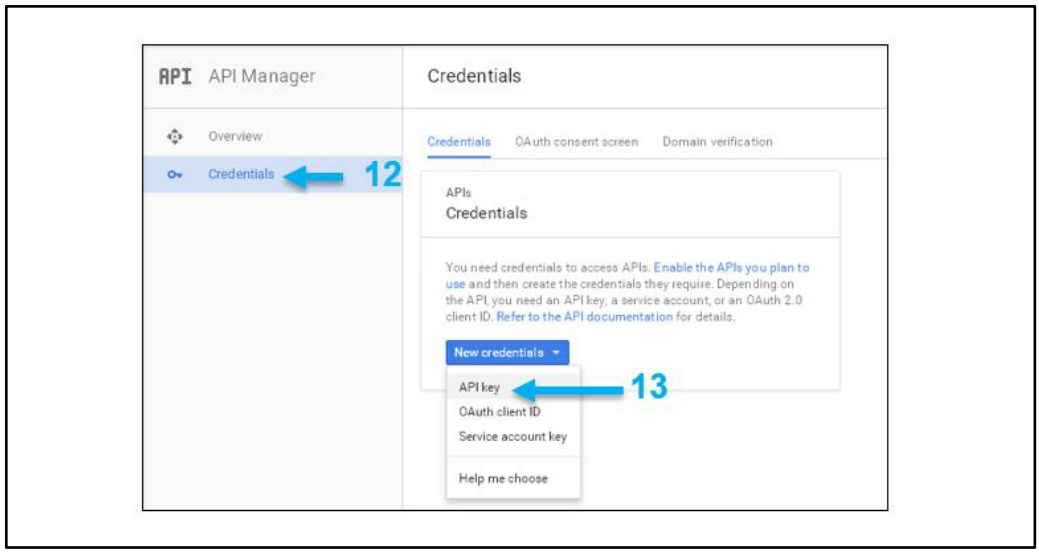
12 Write “Speech API” in the search box, and select the correct result.

Figure 43: Google APIs - search for speech API



13 Enable the Speech API clicking on the blue button.

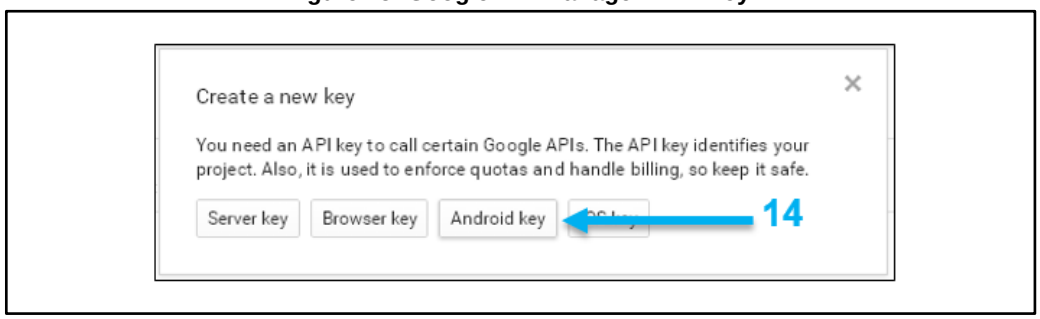
Figure 44: Google APIs - enable speech API



14 Move from the “Overview” tab to “Credentials” tab.

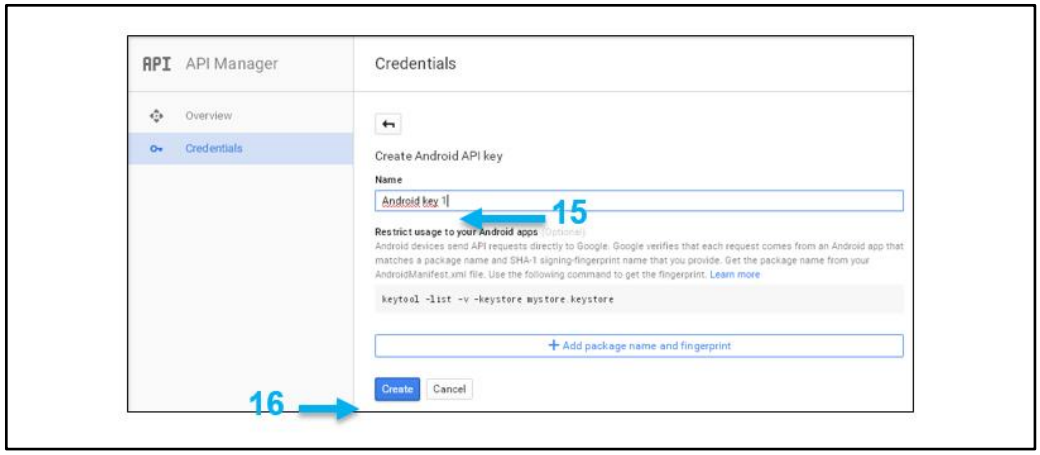
15 Open the “New credentials” menu and select “API key”.

Figure 45: Google API Manager - API key



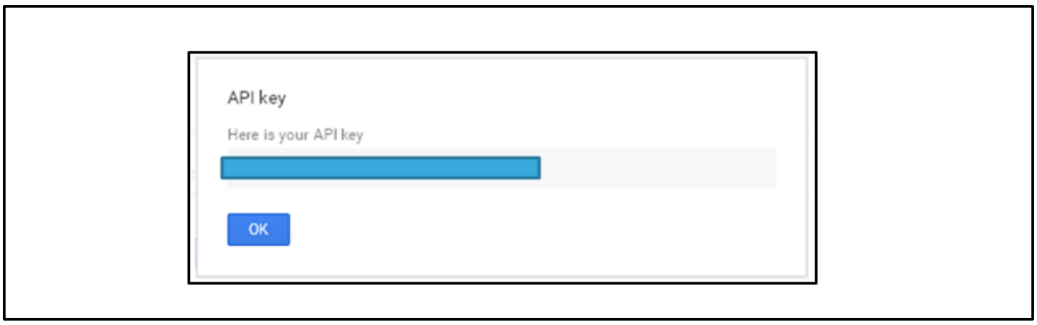
16 Select the Android key button from the pop-up window.

Figure 46: Google API Manager - Android API key



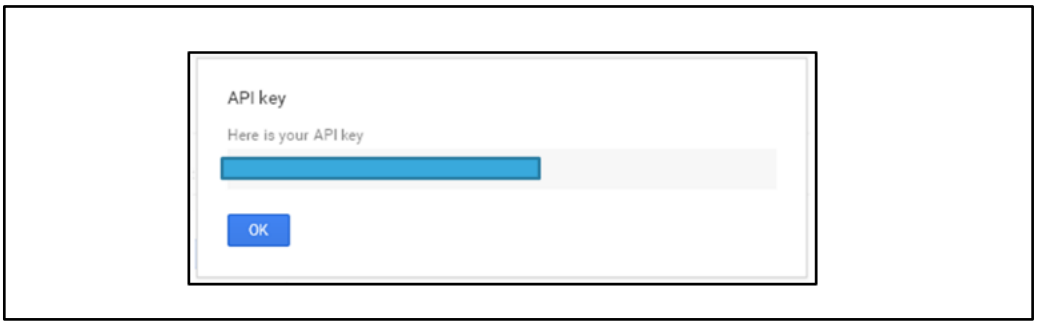
17 Write a valid name for the Android API key.

Figure 47: Figure 16: Google API Manager - Android API key name



18 Click "Create” button and a pop-up window will appear with your key.

Figure 48: Google API Manager - Android API key confirmation



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
3-Aug-2015	1	Initial release.
09-Sep-2016	2	Throughout document: - added NUCLEO-L476RG and NUCLEO-L053R8 development board compatibility information – added X-NUCLEO-IDB05A1 expansion board compatibility information - text and formatting changes Updated *Section "Introduction"* and *Section 2.1: "Overview"* Updated *Figure 1: "BLUEVOICELINK1 software architecture"* Updated *Section 2.5.1.4: "BlueVoice service"* Moved and updated *Section 2.7: "BLUEVOICELINK1 application description"* (was Section: 2.6) Moved and updated *Section 3.3: "PC audio recording utility example: Audacity"* (was Section: 2.7) Added *Section 2.6: "OSXBLUEVOICE library software description"* Updated *Section 3.1: "Hardware description"* Added *Section 3.1.3.2: "X-NUCLEO-IDB05A1 expansion board"* Updated *Section 3.4: " Hardware and software setup "* Added *Section 4: "BLUEVOICELINK1 Android™/iOS™ demo setup"*

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.