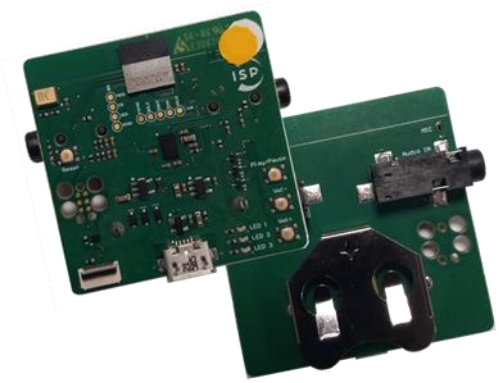
# ISP2080
## Application note AN221201

# Use of ISP2080
### BLE 5.2 Audio Demo Board



## Scope

This application note describes how to set up a BLE 5.2 audio demonstration using the ISP2080 board.

## Contents

## Document Revision History

| Revision | Date | Ref | Change Description |
|---|---|---|---|
| R0 | 09/12/2022 | jf pg | Initial version |

# 1. Introduction

## 1.1. LE Audio

This demonstration shows the use of the new Audio feature introduced in BLE 5.2 (called LE Audio). LE audio is an improvement over the existing "Classic Audio" in Bluetooth classic.

LE Audio provides (non-exhaustive list) :

- Improved sound quality and reduced digital data rate thanks to its new LC3 codec.
- Synchronized audio stream for True Wireless Stereo
- Multiple broadcast streams

## 1.2. Prerequisistes

### 1.2.1. Hardware requirements

- ISP2080 board
- Programming interface – either:
  - ISP Interface board
  or
  - JLINK Base probe
  https://www.segger.com/products/debug-probes/j-link/models/j-link-base
  - 6-pin adapter for the JLink Base probe
  https://www.segger.com/products/debug-probes/j-link/accessories/adapters/6-pin-needle-adapter

### 1.2.2. Software requirements

- nRF Connect SDK v2.2.0
  https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.2.0
- nRF Connect Desktop with Toolchain Manager
  https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-desktop
- nRF Command Line Tools
  https://www.nordicsemi.com/Products/Development-tools/nrf-command-line-tools

## 2. Board Hardware Description

### 2.1. Description

The ISP2080 is a 40 x 40 mm² board that contains:

- ISP2053 BLE module
- A CS47L63 Audio DSP from Cirrus Logic
- Micro USB connector
- CR2032 Battery holder
- 2x 3.5mm Jack connector for audio IN & OUT
- JTAG footprint
- FPC connector
- 4 x buttons
- 3 x LEDs

The ISP2053-AX module is based on nRF5340 Nordic Semiconductor wireless SoC with two Arm® Cortex®-M33 processors and advanced feature set, the ISP2053 module is ideally suited for LE Audio, professional lighting, advanced wearables, industrial, medical, and other complex IoT applications.

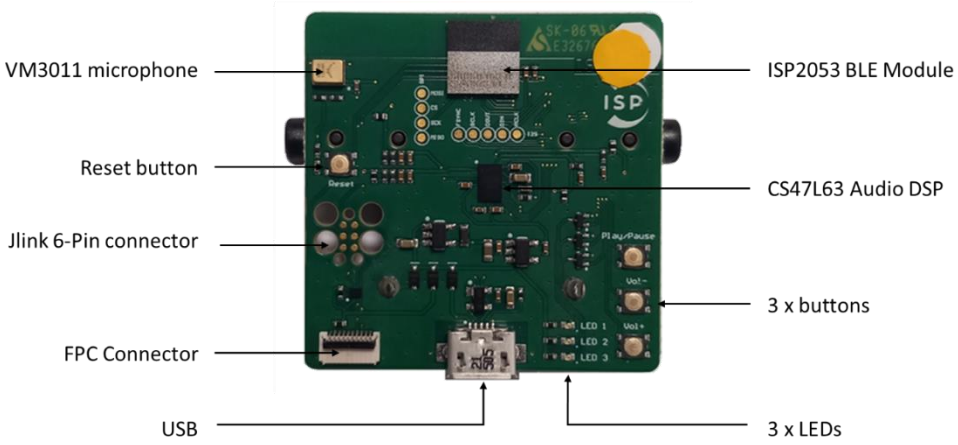



*Figure 1: ISP2080 Top View*

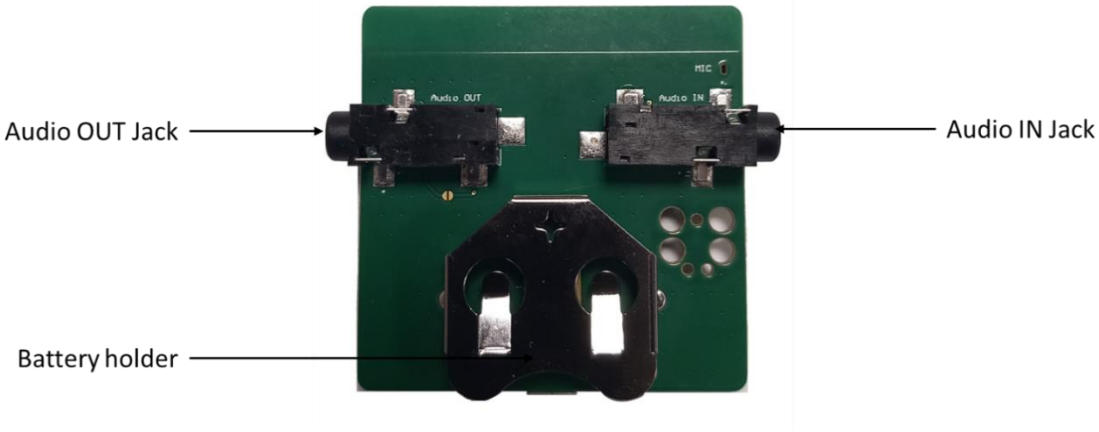
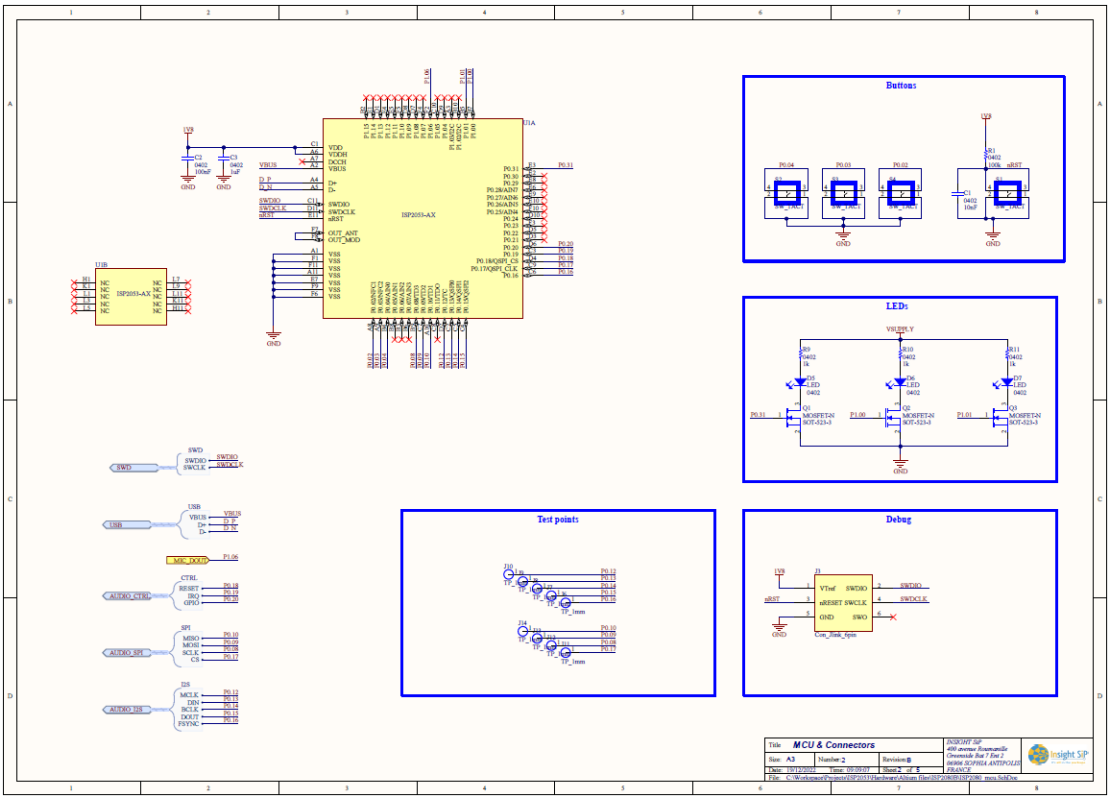
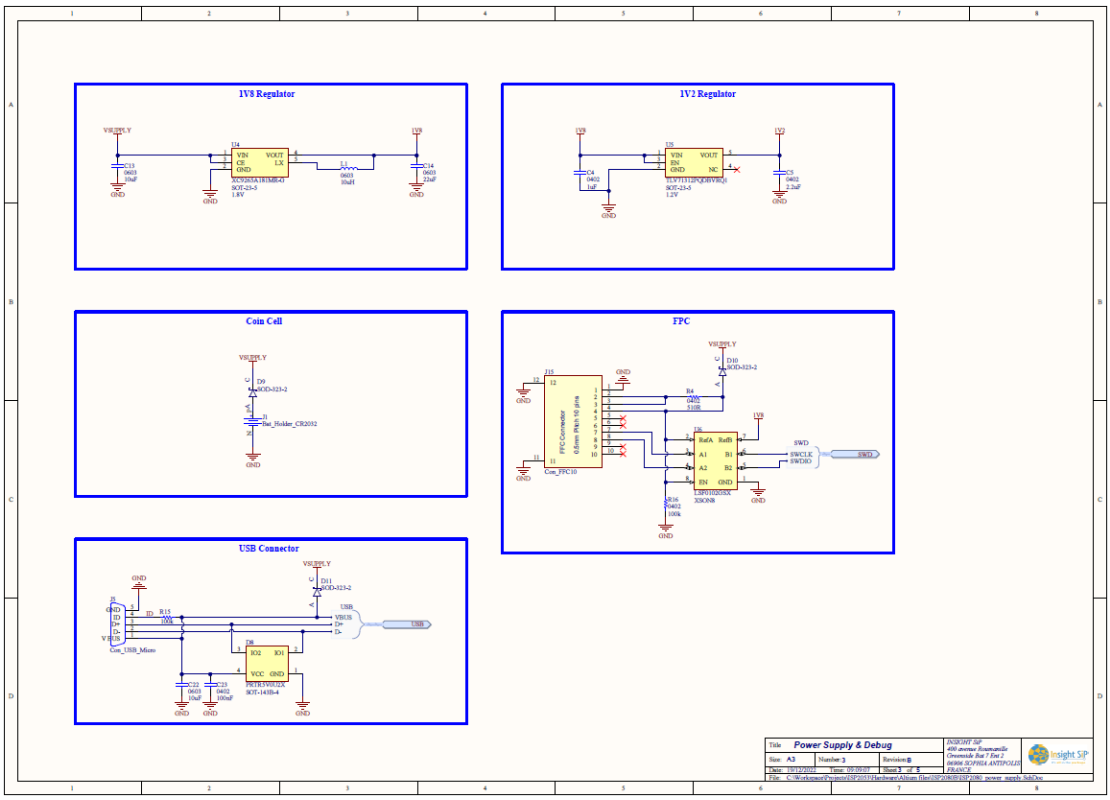
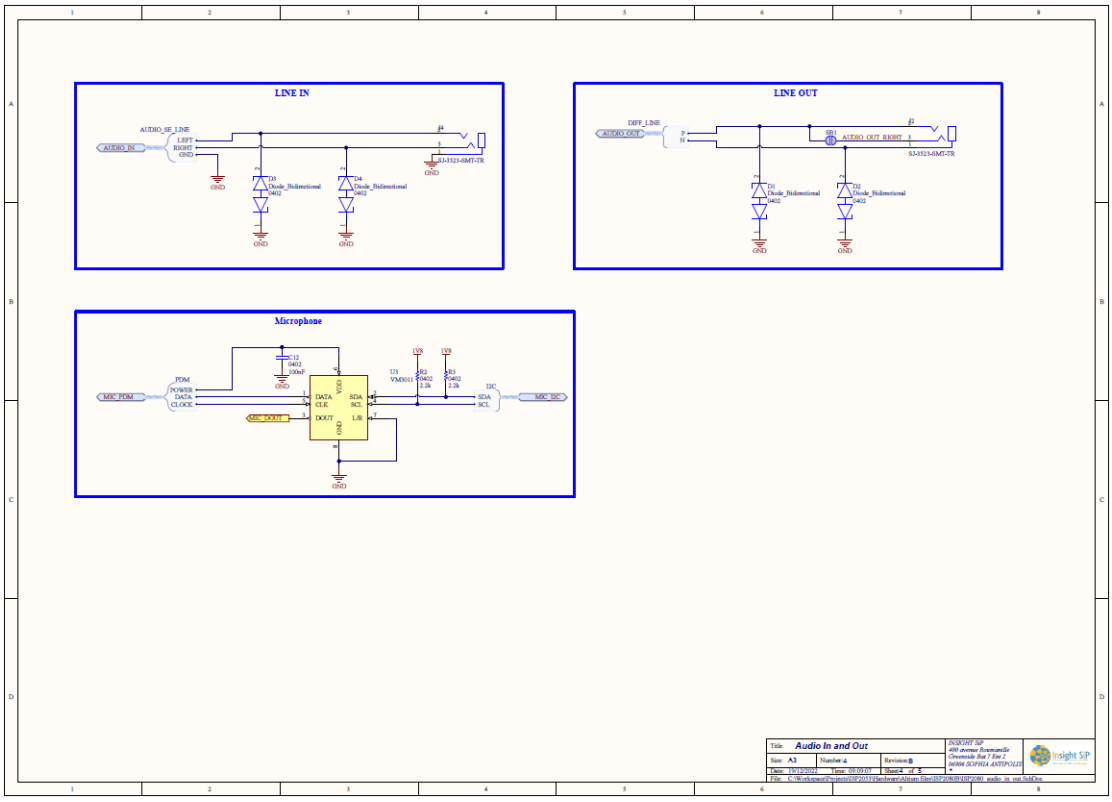


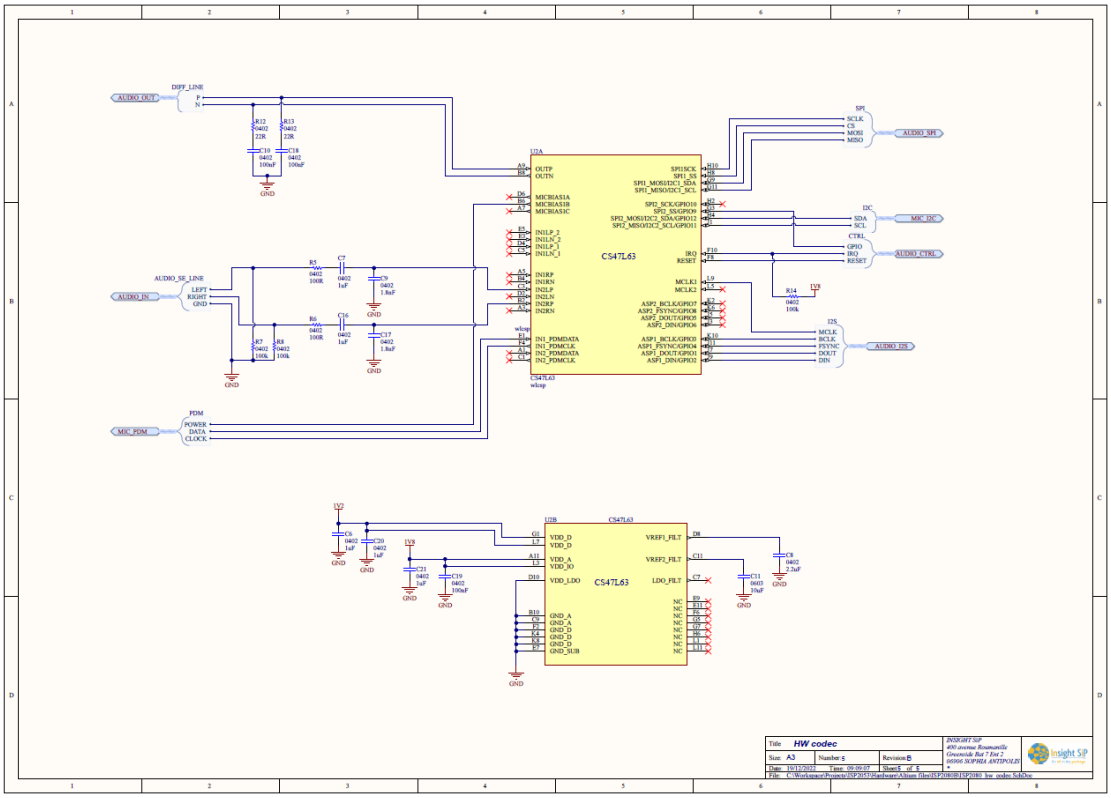

*Figure 2: ISP2080 Bottom View*

## 2.2. Schematics








## 2.3. Recommendations

This schematic is designed as a demonstrator. In a finished product we recommend using a Li-Po battery-based power supply system with a built-in battery charger.

# 3. Software

## 3.1. Overview

The ISP2080 board is compatible with the nRF5340 audio application demo.

This application requires 2 devices:

- A Gateway device responsible for transmitting audio data from an external source to the other device via BLE.
- A Headset device receiving audio data from BLE and playing back to a headset or speaker.

For more detail go to:
https://developer.nordicsemi.com/nRF_Connect_SDK/doc/2.2.0/nrf/applications/nrf5340_audio/README.html

## 3.2. Building the application

### 3.2.1. nRF Connect SDK

The nRF Connect SDK is used to develop on the ISP2080 board. The process is nearly identical to that use for the Nordic Semiconductor nRF5340 Audio kit.

The application source code is available in the nRF Connect SDK (from v2.0.0). The SDK is located in:
*<NCS_ROOT>\v2.2.0\nrf\applications\nrf5340_audio*

This chapter will assume nRF Connect SDK v2.2.0 is being used. There might be some minor differences if another version is being used.

### 3.2.2. Source code modification

Before building the application, we need to modify the source code to remove unnecessary initializations and checks.

Open main.c with a text editor and comment the following lines in the main() function:

- Board_version_valid_check()
- Board_version_get()
- Pmic_init()
- Pmic_default_set()
- Sd_card_init()

These edits are necessary since the ISP2080 PCB is somewhat simplified relative to the Nordic Audio dev kit. The ISP2080 has a simplified power supply that does not use the Nordic Semi NPM1100 PMIC.an SD card. Also, the ISP2080 has neither an SD card nor a board check feature.
Removing these lines does not impact the rest of the application.

It should look like this:

```
void main(void)
{
    int ret;

    LOG_DBG("nRF5340 APP core started");

    ret = hfclock_config_and_start();
    ERR_CHK(ret);

    ret = led_init();
    ERR_CHK(ret);

    ret = button_handler_init();
    ERR_CHK(ret);

    ret = channel_assign_check();
    ERR_CHK(ret);

    ret = fw_info_app_print();
    ERR_CHK(ret);

/*  ret = board_version_valid_check();
    ERR_CHK(ret);

    ret = board_version_get(&board_rev);
    ERR_CHK(ret);

    if (board_rev.mask & BOARD_REVISION_VALID_MSK_MAX14690_PMIC) {
        ret = pmic_init();
        ERR_CHK(ret);

        ret = pmic_defaults_set();
        ERR_CHK(ret);
    }

    if (board_rev.mask & BOARD_VERSION_VALID_MSK_SD_CARD) {
        ret = sd_card_init();
        if (ret != -ENODEV) {
            ERR_CHK(ret);
        }
    }

    ret = power_module_init();
    ERR_CHK(ret); */

#if defined(CONFIG_AUDIO_DFU_ENABLE)
    /* Check DFU BTN before Initialize BLE */
    dfu_entry_check();
#endif
```

*Figure 3: main() code modification*

### 3.2.3. Building and configuration

The SDK provides script to automate building & flashing but this document will focus on building the application using command lines.

Using Toolchain Manager click on "open bash" and go to the audio demo directory.
*cd nrf/applications/nrf5340_audio*

To configure the application, edit one of the following files:

- Prj.conf if the application is built in debug mode.
- Prj_release.conf if the application is built in release mode.

In this file add:
CONFIG_TRANSPORT_CIS=y for a connected mode transport (default)
CONFIG_TRANSPORT_BIS=y for a broadcast mode transport
CONFIG_AUDIO_SOURCE_USB=y for setting USB as audio source (default)
CONFIG_ AUDIO_SOURCE_I2S=y for setting I2S (Jack connector) as audio source
CONFIG_STREAM_BIDIRECTIONAL=y to enable bidirectional transport.
CONFIG_WALKIE_TALKIE_DEMO=y to Configure the demo for a bidirectional communication using the embedded microphone on each side

Depending on the device type build the application with one of the following lines.

- For a "Headset" device:
  *west build -b nrf5340_audio_dk_nrf5340_cpuapp --pristine -- -DCONFIG_AUDIO_DEV=1 -DCONF_FILE=prj_release.conf*
- For a "Gateway" device:
  *west build -b nrf5340_audio_dk_nrf5340_cpuapp --pristine -- -DCONFIG_AUDIO_DEV=2 -DCONF_FILE=prj_release.conf*

Alternatively, the configuration can be directly configured using the command line:
*west build -t guiconfig*

The following window should appear:

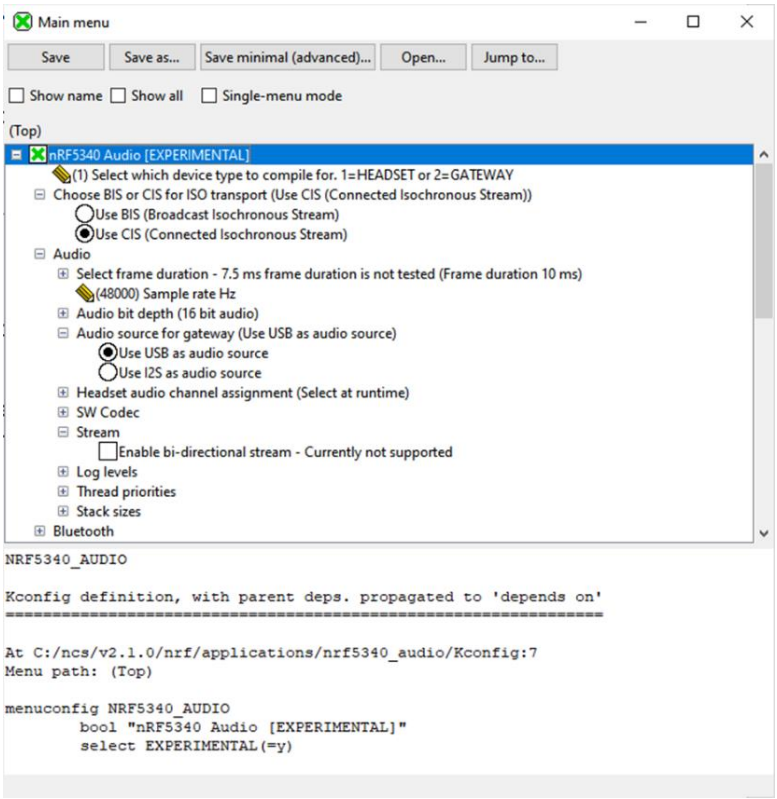



*Figure 4: Zephyr configuration*

After configuration, save, close and rebuild the application without the –pristine parameter (which will erase the configuration)

*west build -b nrf5340_audio_dk_nrf5340_cpuapp -- -DCONFIG_AUDIO_DEV=1 -DCONF_FILE=prj_release.conf*

## 3.3. Flash & Debug

### 3.3.1. Hardware Setup

The ISP2080 board can be flashed with either:

- An Interface Board connected to the ISP2080 with a FPC cable
- A JLink Base probe connected to the ISP2080 with a 6-Pin adapter

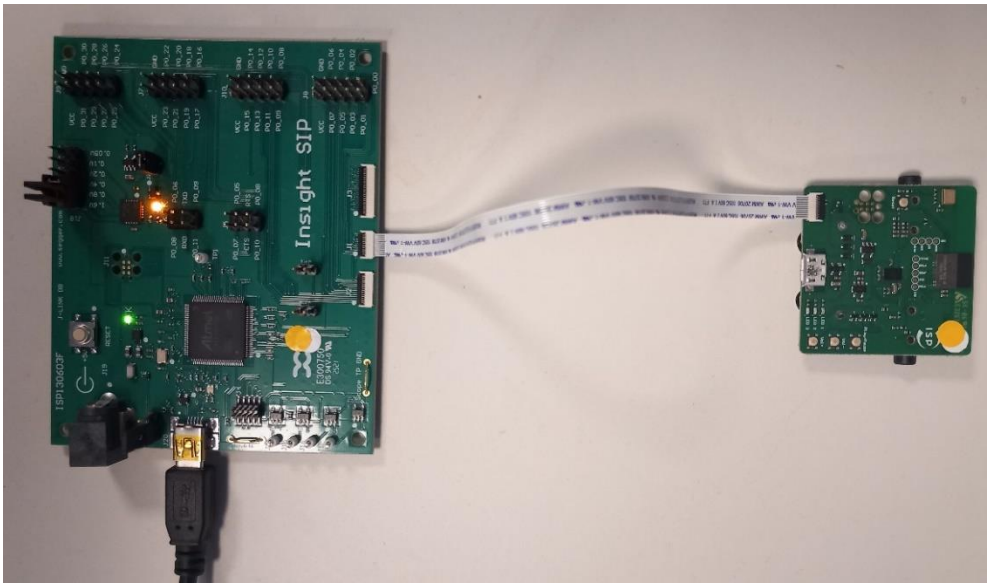



*Figure 5: Flash with Interface Board*

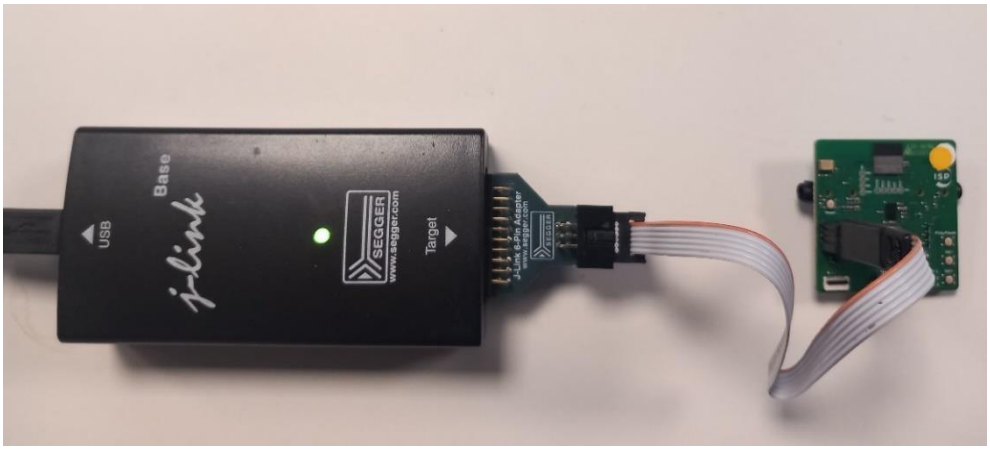



*Figure 6: Flash with JLink programmer*

### 3.3.2. Software procedure

Program the network processor first:

*nrfjprog --program bin/*.hex --chiperase --coprocessor CP_NETWORK -r*

Program the application processor:

*nrfjprog --program build/zephyr/zephyr.hex --coprocessor CP_APPLICATION --sectorerase -r*

If the flashing failed, the module might be locked. You may recover using these commands:

*nrfjprog --coprocessor CP_NETWORK --recover*

and:

*nrfjprog --recover*

# 4. Testing the Demo

## 4.1. PC to Headset demo

### 4.1.1. Description

The "Headset" device will be an ISP2080 connected to a headset using an audio jack cable.

The "Gateway" device will be an ISP2053-TB connected to a computer using an USB cable. This can also be another ISP2080 using its USB connector.

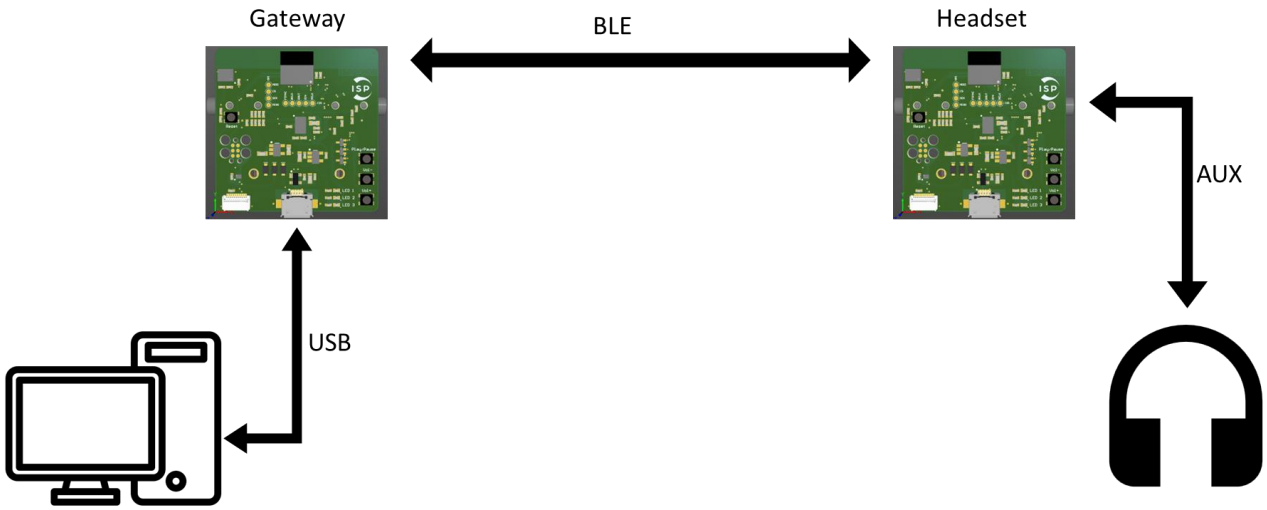



*Figure 7: PC to Headset Demo setup*

### 4.1.2. Building

Configuration in the prj.conf or the prj_release.conf file:

*# nRF5340 Audio*
*CONFIG_NRF5340_AUDIO=y*

Build and flash the headset device with:

*west build -b nrf5340_audio_dk_nrf5340_cpuapp -- -DCONFIG_AUDIO_DEV=1 -DCONF_FILE=prj_release.conf*

*nrfjprog --program build/zephyr/zephyr.hex --coprocessor CP_APPLICATION --sectorerase -r*

Build and flash the gateway device with:

*west build -b nrf5340_audio_dk_nrf5340_cpuapp -- -DCONFIG_AUDIO_DEV=2 -DCONF_FILE=prj_release.conf*

*nrfjprog --program build/zephyr/zephyr.hex --coprocessor CP_APPLICATION --sectorerase -r*

#### 4.1.3. Testing

Plug the headset in the "Audio OUT" Jack connector of the Headset device and insert a CR2032 battery in its battery holder. LED3 should start blinking.

Plug the USB of the Gateway device into the computer. If it is an ISP2053-TB make sure the switch is set to High-Voltage mode so its power can be supplied by USB.

Both devices should connect automatically.
Upon connection:

- LED1 on the Headset device should start blinking.
- The computer should detect a new audio device called "nRF53 audio device".

Now press play/pause button on the Headset device. The demo is now running. Any sound played on the computer will be heard on the headset.

### 4.2. Walkie – talkie demo

#### 4.2.1. Description

Both devices use the embedded microphone as audio input and a headset (with Jack connector) as audio output.

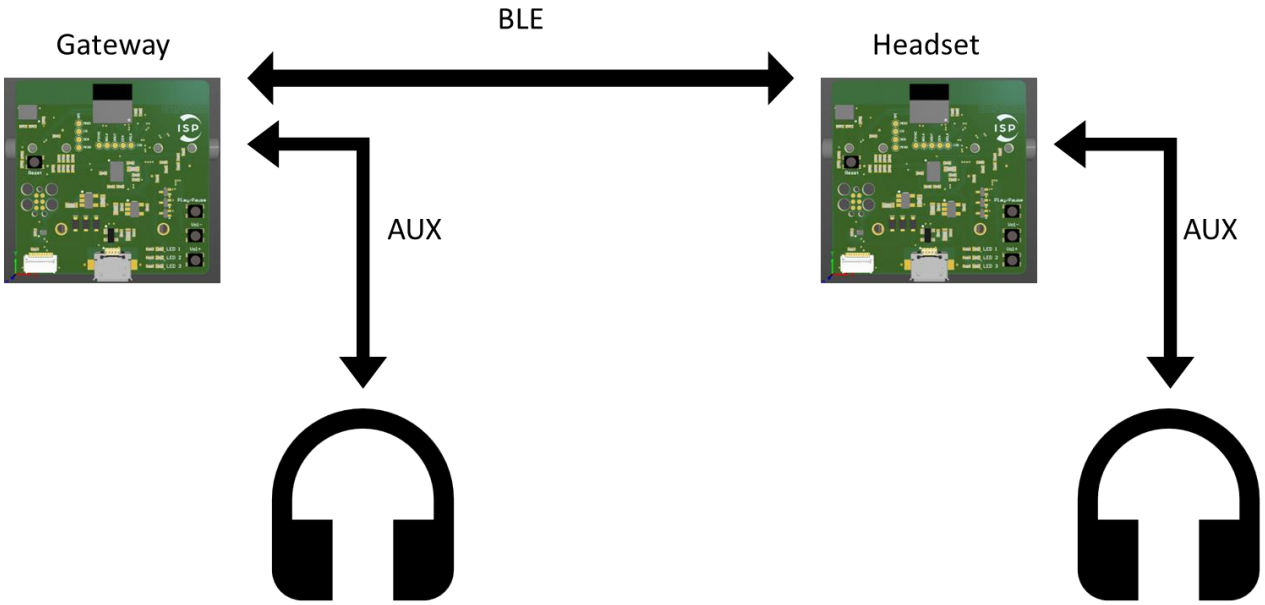



*Figure 8: Walkie-Talkie demo setup*

#### 4.2.2. Building

Configuration in the prj.conf or the prj_release.conf file:

*# nRF5340 Audio*
*CONFIG_NRF5340_AUDIO=y*
*CONFIG_STREAM_BIDIRECTIONAL=y*
*CONFIG_WALKIE_TALKIE_DEMO=y*

Build and flash the headset device with:

*west build -b nrf5340_audio_dk_nrf5340_cpuapp -- -DCONFIG_AUDIO_DEV=1 -DCONF_FILE=prj_release.conf*

*nrfjprog --program build/zephyr/zephyr.hex --coprocessor CP_APPLICATION --sectorerase -r*

Build and flash the gateway device with:

*west build -b nrf5340_audio_dk_nrf5340_cpuapp -- -DCONFIG_AUDIO_DEV=2 -DCONF_FILE=prj_release.conf*

*nrfjprog --program build/zephyr/zephyr.hex --coprocessor CP_APPLICATION --sectorerase -r*

#### 4.2.3. Testing

Plug the headset in the "Audio OUT" Jack connector of the Headset device and insert a CR2032 battery in its battery holder. LED3 should start blinking.

Plug a second headset in the "Audio OUT" Jack connector of the Gateway device and insert a CR2032 battery in its battery holder. LED3 should start blinking.

Both devices should connect automatically.
Upon connection:

- LED1 on both devices should blink.

The demo is now running.