

# Scalable BLE Auracast Audio Broadcasting System for Multi-Receiver Streaming

**Abstract**—This project realizes an immersive, portable surround sound audio streaming system using Bluetooth Low Energy (BLE) Auracast broadcasting, crafted for both indoor and outdoor environments. Our broadcasting unit is built on the nRF5340 chipset, which can transmit left channel, right channel, or mono audio to multiple BLE receivers. Receivers can be other nRF5340 boards or any headphones or speakers with Auracast connectivity. To achieve proof of concept, we configure one nRF board as a broadcaster while two nRF boards are configured as receivers, each connected to a speaker via auxiliary input. We demonstrate extended usability by including a 3rd party Auracast-enabled headset as a 3rd receiver. This project also uses an ESP32 control board to improve the user experience, as it offers WiFi connectivity, physical GPIO controls (for play, pause, volume, stop, etc.), and a RESTful API for remote access. Audio files can be played through SPIFFS storage, such that the firmware does not need to be reflashed for increased convenience. The system is configured and deployed with Nordic’s SDK on Zephyr OS to build both the sink and broadcast applications for BLE transmission. The project serves as a successful minimum viable product (MVP), with a tested broadcasting range of up to 51 ft with custom hardware and 71 ft with third-party headphones.

**Index Terms**—Bluetooth, Bluetooth Low Energy, wireless communication, embedded systems, audio systems,

## I. INTRODUCTION

Traditional multi-speaker systems are cumbersome, often requiring complex setups with WiFi or hardwired connections, which limits their portability and ease of use. Consumers seeking immersive audio experiences in outdoor or temporary setups face challenges due to the lack of simple, scalable solutions. Existing Bluetooth speakers are mostly limited to single-point connection, which prevents true multi-speaker synchronization. Our project addresses this gap by leveraging BLE broadcasting to synchronize and transmit audio across a virtually unlimited number of receiving units, enabling immersive audio anywhere with minimal setup. We configure the audio variant of the Nordic Semiconductor nRF5340 development kit to act as a BLE audio broadcaster that is capable of transmitting a mono audio stream over BLE Auracast. Multiple BLE receivers are programmed to receive and play the broadcast audio in sync. Audio streaming is available via a phone, computer, or a different host device, which can be connected to the broadcaster’s auxiliary input. ESP32 is mapped to basic controls to start and stop the audio, as well as change its volume. To validate the functionality of our project, we test the BLE broadcast range, which is discussed in our testing and results section.

## II. SYSTEM ARCHITECTURE

### A. System Overview

The system we originally conceived of has a few core components. An ESP32 board serves as a user-friendly front-end—a way to interact with the system. Using a phone, computer, or other host device, audio is loaded into the ESP board’s memory wirelessly. Not only does the ESP32 have physical controls for the user, but it is able to continuously stream audio using its own WiFi capabilities, such that these external connections can be shut off after the connection is initiated. The ESP32 board would then stream that audio to our broadcaster. For this project, that job is accomplished by an nRF5340\_audio DK board, configured to be the broadcasting board. These boards have two modes, broadcast and receive, and come equipped with BLE capabilities—specifically Auracast, a way to broadcast audio one-to-many. On the receiving end are two nRF5340\_audio kits flashed with the broadcast sink program and a 3rd party headset. Each board is connected directly from the kits to a speaker via auxiliary output. The 3rd party headset comes preconfigured to receive Auracast streams and connects seamlessly to the broadcaster.

### B. Transmitter Source Code Functionality

The source code running on the transmitter (nRF5340) performs the following tasks:

- 1) Initializes the Zephyr kernel native to the board
- 2) Initializes and configures a Zbus to transport raw audio data through the system.
- 3) Creates the audio stream controller, which: Pulls data from the Zbus channel and passes audio to the Bluetooth LE subsystem on the nRF5340 processor.
- 4) Configures and activates both metadata and audio advertising mechanisms.
- 5) Begins broadcasting the connection for audio sinks, and Auracast enables devices to connect to the stream.

### C. Receiver Source Code Functionality

The receiver firmware on the nRF5340 kits includes the following components:

- 1) Initializes the Zephyr kernel.
- 2) Sets up quality-of-service management and peripherals, including the I2S audio routing protocol.
- 3) Configures physical buttons for:
  - Volume control (up/down)
  - Play/pause functionality
  - Start/stop broadcast search

- 4) Scans for broadcast sources using Zephyr’s Bluetooth API.
- 5) Forwards incoming metadata and audio to the internal Zbus.
- 6) Forwards audio data to the onboard DAC, which then sends data to the kit’s auxiliary output pins.

### III. HARDWARE COMPONENTS

- **3 nRF5340\_audio DK kits** running programs on the application core (cpu\_app):
  - 1 configured as a broadcast source
  - 2 configured as receivers running a broadcast sink
- **1 Windows 10 device:**
  - Used for developing, building, and flashing the nRF PCB
- **1 host machine** (mobile phone or computer):
  - Used for streaming of video/audio
- **1 ESP32:**
  - Intended for lightweight audio streaming
- **1 BLE Auracast-enabled headset**
- **3 auxiliary cords:**
  - 1 from host machine to the transmitter/broadcaster
  - 1 from nRF receiver to speaker 1
  - 1 from nRF receiver to speaker 2

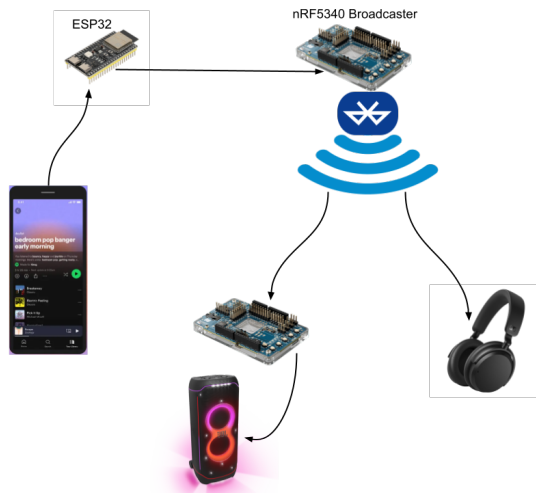


Fig. 1: System architecture overview: audio is streamed from a mobile device to an ESP32 controller, which transmits it via I2S to the nRF5340-based BLE broadcaster. The broadcaster uses Auracast to wirelessly transmit the audio stream to multiple receivers: two nRF5340 boards connected to speakers and a third-party Auracast-enabled headset.

### IV. SOFTWARE IMPLEMENTATION

Our approach utilizes an ESP32 as the central controller for the nRF5340 SoC. It manages audio playback controls (play/pause/volume) through a REST API hosted over a WiFi access point, encodes MP3 files into an analog signal, and

streams audio to the nRF5340 via an I2S interface. These interfaces require thorough software implementation.

We wrote custom C code for the ESP32 to handle initializing the Wi-Fi access point, the application programming interface (API), manage GPIO communications between the chipsets, and stream via I2S.

As for the nRF5340, the degree of complexity was much higher, and we ultimately utilized an example provided in the open-source Nordic SDK to handle Bluetooth BIS communications for both the transmitter and receiver.

### V. DATA COLLECTION & ANALYSIS

For our project to be practical, we must ensure that adequate quality, range, latency, and reliability standards are met. For us, these are the four most important facets of audio streaming. To verify this, we define a few baseline metrics for success. Our project is practical if: 1. The audio quality maintains most frequencies from 20 Hz to 20 kHz with minimal attenuation. 2. Each receiver can receive enough data to not affect quality at 5 m from the transmitter. 3. The user can watch videos with a latency below 150 ms for a smooth video watching experience. 4. The system must satisfy the previous three thresholds indoors, outdoors, and in various ambient conditions. Measurement of these categories was done by two human testers. 20 Hz - 20 kHz is a rough guideline for perceptible audio quality. Future work includes testing with external microphones to test the frequency response.

We collected broadcast range distances for a 3rd-party device and the nRF audio kits. In addition, we tested audio latency using video streaming with readable lip movements. The broadcast ranges were tested manually and observed through receiver feedback. Overall, our custom receivers had a shorter range than the 3rd-party headphones, which will be discussed in the testing and results section.

### VI. TESTING & RESULTS

First, to test audio quality, we listened to the broadcast indoors and outdoors at 5 m from the transmitter. We tested multiple receivers to ensure that our audio quality testing was robust, with different speakers connected to the receivers for diversity. In both environments and under all tested conditions, the audio quality is on par with Bluetooth Classic to the human ear (not tested with microphones). This means none of the authors experienced noticeable quality reduction from a standard Bluetooth setup. Since the range of human hearing is about 20 Hz to 20 kHz, we informally confirm that the audio quality passes the baseline threshold.

To test our range, we manually walked away from the transmitter with the receiver hardware to test the broadcast range. Playback latency was also tested qualitatively. Our key metrics were: 71 ft broadcast range for 3rd party headphones and 51 ft for our project’s custom receivers. These are greater than the 10 m baseline metric as defined above. We also achieved a nearly imperceptible latency ( $<150$  ms) at 5 m and, therefore, robust audio synchronization. Lastly, to test reliability, we compared the system’s functionality under

normal operation (within 5 m of the broadcast source) but in different environments. Outdoor performance was close to ideal since there was minimal co-channel interference from other RF equipment. No other broadcasters were within 25 m of any component, and the system was as far from buildings as possible to minimize multipath interference. Indoors, the setup worked flawlessly within 5 m of the transmitter despite being near a Wi-Fi router operating at 2.4 GHz, which is the same as BLE. If this scales up to be a commercial product, we would test our product in extreme ambient conditions (which we did not do because the hardware is expensive) and in the presence of strong RF noise.

## VII. CHALLENGES & TROUBLESHOOTING

The most significant challenge in the backend software development was understanding dependencies and existing definitions from the Zephyr OS and Nordic's samples. Zephyr defines many macros, routines, and enumerated types that are difficult to keep track of, especially since the VS Code extension is not optimized for macOS. We switched to developing on Windows, and many of these issues were resolved. However, tracing function, macro, and enum types were still a hassle in some example projects.

Another challenge working with the software was accessing the serial output from the nRF boards. The built-in application provided by Nordic did not connect properly to the COM ports on our host devices, and PuTTY also would not recognize the target boards. To fix this, we again migrated to a Windows machine. The VS Code extension for the SDK includes a serial terminal emulator that shows logging information crucial to understanding the operation of the running programs. Nordic's software prefers Windows to macOS for development.

With regards to the ESP32, the initial MP3 we tested was too large (around 3 minutes, 4.2 MB) for the limited DRAM and SRAM space. To fix this, we switched to a file-based playback approach using SPIFFS storage. This allowed us to directly upload smaller MP3 files without needing to reflash the firmware each time, significantly improving the efficiency and stability of our implementation while retaining full functionality.

Furthermore, we initially were unable to build and flash code to the ESP32 because the project was configured for the default ESP32 instead of our variant of the ESP32. To resolve this, we explicitly set the target board inside the ESP-IDF to be the ESP32-S3, allowing us to successfully build and flash the chip's firmware. This is a good lesson: always explicitly set the chip rather than relying on the default, as failing to do so can result in wasted time debugging a simple issue.

## VIII. CONCLUSION

In this project, the authors develop a complete backend system for an extendable multi-receiver audio system. Each device is configured in mono mode for ease of extension, since extending multiple L/R channels requires users to configure which devices should correspond to each channel. The system utilizes the new standard Auracast Bluetooth Low Energy

technology, released in 2022. Excitingly, our system may provide a cheaper alternative to synchronized audio playback offered by JBL™ and Sonos™ while providing much more flexibility in the type of audio playback devices (portable speaker, home theater system, headphone, boombox, sound-bars, etc.). Currently, each nRF5340 audio DK is \$159 USD, which is added to the price of each speaker to get the total price for an end user, plus the additional cost for the transmitter kit. This means the total price is  $\$(159 + 159 * \text{num\_receivers} + \text{total\_receiver\_cost})$ . Hopefully, BLE-enabled SoCs with audio capabilities will become cheaper in the near future, making our project more enticing.

Future improvements include building a web-based frontend for a better user experience that provides a direct upload feature so that users can drop local files for playback. Additionally, the audio could be split into multiple synchronized channels for an authentic surround sound. We expect that, with decreasing cost for the same functionality as dictated by Moore's Law, the Nordic development kits (and other similar kits) will become cheaper in the near future. This excites us for the future of this project and any early adopters who would like to create their own custom audio setups.

## REFERENCES

- [1] Electromaker, "BLE 12 Click," *Electromaker.io*, 2025. [Online]. Available: <https://www.electromaker.io/shop/product/ble-12-click>. [Accessed: May 13, 2025].
- [2] Zephyr Project, "Getting Started Guide — Zephyr Project Documentation," *Zephyrproject.org*, 2015. [Online]. Available: [https://docs.zephyrproject.org/latest/develop/getting\\_started/](https://docs.zephyrproject.org/latest/develop/getting_started/). [Accessed: May 13, 2025].
- [3] HearSoundly, "Bluetooth Low Energy Audio and Auracast Explained - For Hearing Aid Wearers," *YouTube*, May 28, 2024. [Online Video]. Available: <https://www.youtube.com/watch?v=hYKPS02I9Q0>. [Accessed: May 13, 2025].
- [4] Bluetooth SIG, "LE Audio," *Bluetooth Technology Website*. [Online]. Available: <https://www.bluetooth.com/learn-about-bluetooth/feature-enhancements/le-audio/>. [Accessed: May 13, 2025].
- [5] Nordic Semiconductor, "Making a 5.1 home-theater-like system with nRF5415 DK," *Nordic DevZone*, Feb. 7, 2025. [Online]. Available: <https://devzone.nordicsemi.com/f/nordic-q-a/118730/making-a-5-1-home-theater-like-system-with-nrf5415-dk>. [Accessed: May 13, 2025].
- [6] element14 Community, "Nordic Bluetooth LE Audio Development Kit nRF5340 DK," Dec. 9, 2022. [Online]. Available: [https://community.element14.com/products/roadtest/rv/roadtest\\_reviews/1646/draft\\_for\\_nrf5340#pifragment-19571=6](https://community.element14.com/products/roadtest/rv/roadtest_reviews/1646/draft_for_nrf5340#pifragment-19571=6). [Accessed: May 13, 2025].
- [7] Nordic Semiconductor, "Technical Documentation: I2S," *Nordicsemi.com*, 2025. [Online]. Available: [https://docs.nordicsemi.com/bundle/ps\\_nrf5340/page/i2s.html](https://docs.nordicsemi.com/bundle/ps_nrf5340/page/i2s.html). [Accessed: May 13, 2025].
- [8] This Week in Hearing, "What you need to know about Auracast broadcast audio from Bluetooth," *YouTube*, Jun. 7, 2023. [Online Video]. Available: <https://www.youtube.com/watch?v=s6i6t2i2wcE>. [Accessed: May 13, 2025].