UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXV Ciclo

A Modular, A^{2}B-based Architecture for Distributed Multichannel Full-Digital Audio Systems

Coordinatore:

Prof. Marco Locatelli

Tutor:

Prof. Carlo Concari

Dottorando: *Nicholas Rocchi*

Anni Accademici 2019/2020 – 2021/2022

I would like to express my deepest gratitude to my PhD tutor Prof. Carlo Concari and to Prof. Andrea Toscani, who assisted me during these years providing support and encouragement. I am also grateful to Prof. Angelo Farina for his valuable advice and guidance.

I would like to extend my gratitude to ASK Industries, who allowed me to undertake this research project, and in particular to Alessandro Costalunga, Lorenzo Ebri, and Carlo Tripodi for the experience and for sharing their knowledge with me.

Thanks should also go to the colleagues of the Acoustics laboratory of the University of Parma, and in particular to Marco Binelli, Daniel Pinardi and Leonardo Saccenti for the stimulating discussions and for the time spent together. Special thanks to Lorenzo Chiesi for being a source of inspiration, and to Marco Simonazzi for his support and for his valuable suggestions.

Finally, I would like to express my deep gratitude to my family, who encouraged me in doing this experience, and to my girlfriend Irene who always trusted in me and supported me in difficult times.

Contents

List of Figures

List of Tables

List of Acronyms

A^2B	Automotive Audio Bus
ACC	Acoustic Contrast Control
ADAT	Alesis Digital Audio Tape
ADC	Analog to Digital Converter
AES	Audio Engineering Society
ANC	Active Noise Cancellation
AOP	Acoustic Overload Point
AVB	Audio Video Bridging
BMCA	Best Master Clock Algorithm
BUJ	Bounded Uncorrelated Jitter
CAN	Control Area Network
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Converter
DDJ	Data-Dependent Jitter
DFU	Device Firmware Upgrade
DJ	Deterministic Jitter
DoA	Directions-of-Arrival
DUT	Device Under Test
ECU	Electronic Control Unit
ESS	Exponential Sine Sweep
FEM	Finite Elements Method
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GPIO	General Purpose Input Output
I^2C	Inter-Integrated Circuit
I^2S	Inter–Integrated Circuit Sound
IGMP	Internet Group Management Protocol

ILZ	Individual Listening Zone
IR	Impulse Response
LD	Level Difference
MAC	Media Access Control
MADI	Multichannel Audio Digital Interface
MEMS	Micro Electro-Mechanical System
MIDI	Musical Instrument Digital Interface
MOST	Media Oriented Systems Transport
MSB	Most Significant Bit
MTU	Maximum Transmission Unit
PCB	Printed Circuit Board
PDM	Pulse Density Modulation
PJ	Periodic Jitter
PLL	Phase-Locked Loop
PM	Pressure Matching
PTP	Precision Time Protocol
QoS	Quality of Service
RJ	Random Jitter
RTP	Real-time Transport Protocol
S/PDIF	Sony/Philips Digital Interface Format
SC	Spatial Correlation
SCF	Synch Control Frame
SH	Spherical Harmonic
SIP	Session Initiation Protocol
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
SPL	Sound Pressure Level
SRF	Synch Response Frame
SRP	Stream Reservation Protocol
TDM	Time Division Multiplexing
THD+N	Total Harmonic Distortion plus Noise
TIE	Time Interval Error
TJ	Total Jitter
TSN	Time Sensitive Networks
UAC	USB Audio Class
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
WFS	Wave Field Synthesis

Introduction

Immersive audio, 3D audio and spatial audio are topics that saw great advancements in the last decade, and they gained interest for both academic and industry research. Some application fields are teleconferencing systems [1], virtual and augmented reality [2], automotive [3], [4] and consumer electronics [5]. The recording and the reproduction of immersive audio tracks entail the usage of multichannel systems such as microphone and loudspeaker arrays. As a rule of thumb, the more transducers a system has, the greater is the spatial accuracy that can be reached. However, by increasing the number of transducers, and so the number of audio channels, the system complexity increases. In fact, more connections are required and the processing power of the electronic devices that manage the audio data must be higher, and hence their cost increases too. Moreover, the choice of a suitable audio transport protocol plays a key role in the development of multichannel audio systems.

In the last decade, audio-over-ethernet (i.e., Audio Video Bridging - AVB) and audio-over-IP (i.e., Dante and Ravenna) technologies became more and more popular thanks to their versatility and high bandwidth. These technologies leverage ethernet cables and TCP/IP protocols to deliver multichannel high-quality audio. On the other hand, they typically introduce from 0.5 ms to 2 or 5 ms of latency, which may cause limitations in applications that require real-time processing. In addition, powerful electronic devices like Field Programmable Gate Array (FPGA) or ARM-based processors are required to manage the protocol stacks, especially when the system has a high channel capability. Older technologies such as AES10, also known as Multichannel Audio Digital Interface (MADI), are still employed thanks to their simplicity and effectiveness, while few adopt an analog signal distribution.

In this work, a new audio distribution architecture is proposed. It aims to overcome the limitations that were previously described by leveraging the Automotive Audio Bus (A^2B), a new digital bus developed by Analog Devices. A^2B allows to

transport up to 32 audio channels together with command data and power over an Unshielded Twisted Pair (UTP) cable. It is a cost-effective solution as the access to the bus is managed by dedicated transceivers, which do not require programming but only an initial configuration. Moreover, A^2B introduces an overall latency of just two samples (i.e., less than 50 µs at 48 kHz). Another relevant parameter related to clock propagation is jitter. In fact, each A^2B transceiver reconstructs the clock from the previous one, possibly leading to a clock deterioration in terms of jitter. For this reason, the A^2B clock propagation was investigated, and a jitter model useful to identify audio quality degradation, especially in microphone arrays, was developed.

Although A^2B was initially designed for the automotive industry, its features make it suitable for other scenarios too. This dissertation presents three possible applications of the proposed architecture:

- *Audio distribution in a car*: this is the natural application of A^2B as it was developed for automotive. In this work, it will be described the use case of the A^2B architecture for the Individual Listening Zone (ILZ) application. ILZ is a proprietary technology developed by ASK Industries S.p.A., which is intended to deliver personalized audio content in an environment where multiple listeners are present, in this case the car cabin.
- *Wave Field Synthesis (WFS) listening room*: WFS is a spatial sound reproduction technique based on the reconstruction of a sound wave front by means of a large number of loudspeakers, which are typically arranged linearly around the listening area. Since the spatial accuracy increases as the distance between the loudspeakers decreases, this application requires a high number of adjacent loudspeakers to cover the listening area. In addition, the loudspeakers must be driven individually. As it will be described, the proposed architecture was employed to simplify the wiring and the system architecture, thus reducing the system cost.
- *Microphone arrays*: they are becoming more and more popular in several applications ranging from teleconferencing to immersive audio recording systems. In this case, A^2B was exploited to realize a planar triangular microphone array equipped with four capsules. This microphone array was designed in partnership with ASK Industries S.p.A. The peculiarity of this microphone array is that it can be combined with other

units to build planar, three-dimensional, or even distributed *meta-arrays*, namely array of arrays.

Despite the many advantages of A^{2}B, it presents some limitations when employed for applications different from automotive. The first one is the cable length. In fact, an A^{2}B bus can be up to 40 m (or 80 m for the new transceivers) long, which may limit the field of application. To overcome this limitation, the proposed architecture includes a protocol conversion board, which converts data from A^{2}B to one of the most common professional audio technologies (e.g., AVB, MADI, Dante, or Ravenna), allowing to reach greater distances. Another limitation comes from the static nature of this technology. As it will be better explained in the next sections, devices on the A^{2}B bus require an initial configuration to work, performed for example by a microcontroller, which must know in advance all the characteristics of the devices on the bus. In the automotive field, this is a desirable feature as the bus will remain unchanged once installed. On the contrary, professional or consumer audio applications entail for example connecting and disconnecting the devices on the bus, changing the type of devices connected, or changing the channel routing. To overcome this limitation, a new A^{2}B feature called *A^{2}B Plug-and-Play* was developed, allowing to dynamically change the number, the type, or the order of the devices on the bus.

This dissertation is organized as follows:

- Chapter 1 provides a description of the A^{2}B technology presenting its features and functioning. Then, the most common audio protocols are described. Finally, a comparison of the presented technologies is provided.
- Chapter 2 describes the proposed data transport architecture. In this chapter it is also presented the clock propagation analysis performed on an A^{2}B bus. Measurements of jitter and latency were carried out on each A^{2}B node to investigate the impact of the bus on beamforming performance of transducer arrays, such as microphone and loudspeakers arrays. Lastly, the A^{2}B Plug-And-Play feature is described.
- Chapter 3 presents three case studies exploiting the proposed architecture, namely the audio distribution in a car for the ILZ application, a WFS listening room and microphone arrays. Each application is described in a dedicated section presenting the general functioning, the devices that were developed and the obtained results. The audio amplifiers

were evaluated by characterizing the thermal performance and by measuring the audio performance parameters such as THD+N, output noise, and frequency response. Instead, the performance of the microphone arrays was compared by measuring parameters such as directivity factor, half-power beamwidth, spatial correlation, and level difference.

1 Protocols Comparison

1.1 What is A^{2}B?

A^{2}B is a technology developed by Analog Devices in the last years to satisfy the new requirements of the automotive field [6]. It is a digital audio bus, which can transport multi-channel audio as well as control data and power over a single UTP cable. A^{2}B is based on a daisy-chain topology consisting of a single main node and multiple subordinate nodes. The access to the bus is made possible by A^{2}B transceivers. Currently there are two product families available: AD242x and AD243x. The main difference between the two families is the power delivery. AD243x chips can supply on the bus up to 50 W, allowing to supply subordinate nodes on the bus such as power amplifiers. For both families, the bus has a maximum bandwidth of 50 Mbit/s, allowing to transport up to 32 audio channels. The supported sample rates are 44.1 kHz and 48 kHz, and the channel widths are 12, 16, 24 and 32 bits. Other improvements of the AD243x family concern the maximum bus cable length and the maximum number of nodes connected to the bus. For the AD242x chips, the maximum cable length is 40 m and a maximum of 10 subordinate nodes can be connected to the bus. AD243x chips increase the cable length to 80 m, and they allow connecting up to 16 subordinate nodes. The maximum distance between nodes remains 15 m for both families. Specifications of the two transceivers are shown in Table I.

A block diagram of an A^{2}B network is shown in Figure 1. The red box represents the main board, which comprises an audio host, a target processor and an A^{2}B transceiver. The audio host and the target processor can be the same device (e.g., microcontroller, FPGA or DSP). The audio host manages the audio data flow, and it communicates with the A^{2}B transceiver through Inter–Integrated Circuit Sound (I^{2}S) or Time Division Multiplexing (TDM).

Table I - AD2428 and AD2435 transceivers specifications

Feature	AD2428	AD2435
Max number of subordinate nodes	10	16
A^2B bus power	2.7 W	50 W
Nominal bus bias voltage	4 V to 9 V	7 V to 24 V
Max network cable length	40 m	80 m
Max node to node cable length	15 m	15 m
I^2S/TDM support	Yes	Yes
PDM inputs	4	4
SPI over distance	No	Yes

The target processor is typically a microcontroller, and it is mainly needed to configure the network at the power on of the system. The A^2B transceiver, once configured by the target processor, manages the communication over the bus and passes audio data and Inter-Integrated Circuit (I^2C) or Serial Peripheral Interface (SPI) commands between the subordinate nodes and the audio host and the target processor. The blue boxes in Figure 1 represent the subordinate boards. They can be equipped with several devices depending on the application, but they must have an A^2B transceiver, which receives data on the bus from the previous node and it transmits to the next node. In addition, A^2B transceivers offer a Pulse Density Modulation (PDM) interface to communicate, for example, with Micro Electro-Mechanical System (MEMS) devices, such as microphones and accelerometers.

At the power on of the system, both main and subordinate transceivers must be initialized to establish the communication on the bus. This task is carried out by the target processor, which has stored the topology of the network (i.e., the number of nodes and their peripherals, and the number of channels exchanged). The initial configuration consists of two phases: *discovery* and *initialization*. During the discovery, the target processor searches for new nodes on the bus and then it initializes them. The main parameters of the transceiver to be configured are the I/O ports configuration, the I^2S/TDM or PDM format settings and the channel slots to add or use. The target processor not only configures the A^2B subordinate transceivers, but it can also initialize their peripherals.

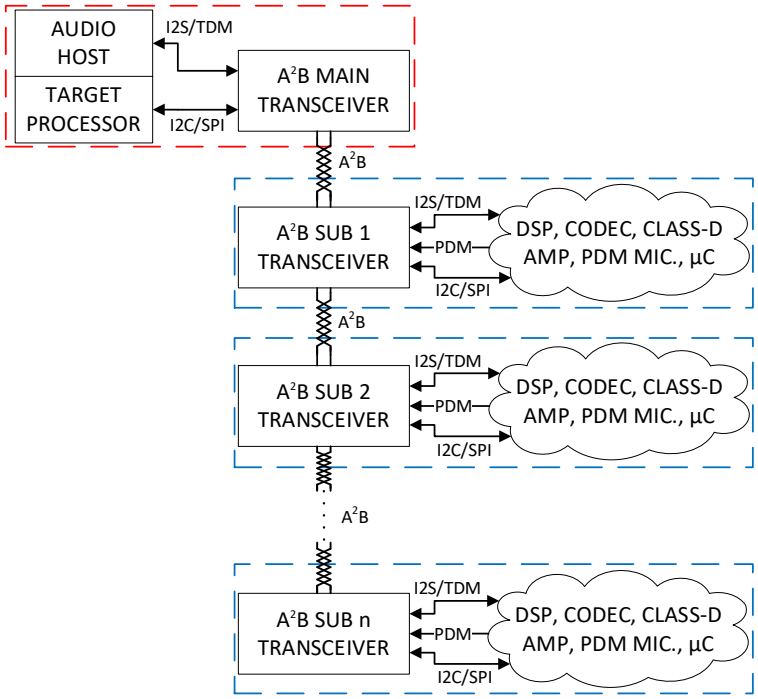


Figure 1. Block diagram of an A²B network composed by a main node and multiple subordinate nodes.

In automotive applications, for which the A²B technology was developed, the network is permanent. Once the system is installed, the number of nodes, their order, the number of channels, and the routing do not change. In these cases, the target processor knows from the start the characteristics of the network and successfully configures it at the power on. This feature is not particularly suited for non-automotive applications, such as professional or consumer audio distribution systems, where the topology of the network, the number of devices, or the channel routing may change frequently depending on the context. In addition, users may need to add or remove devices or even change the channel routing while the network is working. Since the A²B standard features do not allow reconfiguring the network dynamically, in Section 2.3 will be presented an application-level solution that makes A²B a plug-and-play protocol.

Once the initial configuration is finished, audio data flow starts, and the target processor is not required anymore except for optional diagnostic or communication with the devices connected to subordinate nodes. Data exchange over the bus

consists of a *downstream* and an *upstream* phase. The downstream is the communication flow from the main node toward the last sub node, while the upstream starts from the last sub node and finishes at the main node. The duration of these two phases is one sample time. This feature guarantees a completely deterministic latency of 2 samples (less than 50 µs at 48 kHz), independent from the position of the node in the bus. All the sub nodes in the network are kept synchronized by the main node, which receives the clock at the sampling frequency from the audio host. Since excellent device synchronization is crucial when using them together (for example in array applications), performance of the A^2B synchronization mechanism will be investigated in Section 2.2.

Data and clock information are transmitted by the nodes in a packet called *superframe* (Figure 2). The superframe is composed by one header and one payload for each of the two phases (downstream and upstream). The maximum length of the superframe is 1024 bits, so at the sampling frequency of 48 kHz, the baudrate of the bus is 49.152 Mbit/s. The superframe headers for the downstream and the upstream are called Synch Control Frame (SCF) and Synch Response Frame (SRF) respectively and they are both 64 bits long. They consist of:

- A 14-bit preamble used for the subordinate node's Phase-Locked Loop (PLL) synchronization.
- An 18-bit control field.
- 16 bits for I^2C or SPI commands.
- A 16-bit Cyclic Redundancy Check (CRC) field to protect the header.

Each header is followed by a payload containing one audio sample for each channel. When an A^2B transceiver receives the superframe, it can consume audio slots or add data for the subsequent nodes (downstream) or for the previous nodes (upstream).

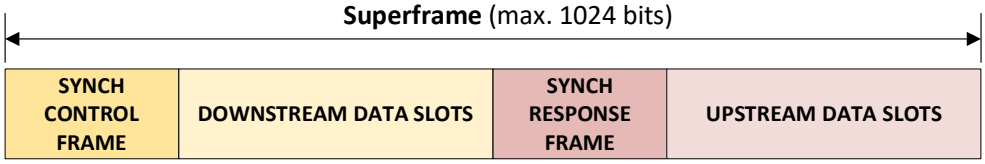


Figure 2. Structure of the A^2B superframe.

If the required number of channels is greater than 32, up to four A^2B networks can be combined to increase the system capabilities to reach a maximum of 128 channels, by exploiting the availability of four I^2C addresses of the A^2B main transceiver. Since the devices of a single bus are synchronized by the clock fed to the main node, no synchronization issues arise if the same clock is fed to all the A^2B networks. It shall be pointed out that the A^2B transceivers are not able to route audio samples between two different networks. Thus, this connection must be implemented at the audio host level.

Another important feature of A^2B is the fault diagnostic. Faults can be detected during nodes discovery or during normal operation and they can also be localized. Types of faults recognized are open wires, wrong port, reverse wires, short wires and short to ground or to power supply. If a fault happens, even if the nodes are connected in daisy-chain, the entire network does not collapse. Only those nodes downstream from the faulty connection are impacted by the failure and they are isolated from the bus.

1.2 Background on Digital Audio Protocols

1.2.1 AES10 (MADI)

AES10, also known as MADI, standardizes data format and electrical characteristics for a multichannel audio digital interface. It has common features with AES3 [7], an earlier stereo digital audio protocol. AES10 was published in 1991 by the Audio Engineering Society (AES) [8] to answer the need of some manufacturers of digital audio equipment to develop a common interface for serial transfer of digital multichannel audio in recording and broadcast studio applications.

MADI is a point-to-point technology, and it allows the transmission of up to 64 channels, 24-bit, at 48 kHz (32 channels at 96 kHz and 16 at 192 kHz) in a unidirectional data stream. The link transmission rate is 125 Mbit/s, irrespective of the sampling frequency or the number of active channels, but the data transfer rate is 100 Mbit/s due to the 4B5B coding. These characteristics make MADI simple, since it does not require user interaction to send or route data once devices are connected together. Supported cables are 75 Ω coaxial and optical.

Figure 3 illustrates the MADI frame in case of a transmission of 64 channels at 48 kHz of sample rate. Channels corresponding to the sample *n* are numbered from 0 to 63. The label *A* or *B* indicates the AES3 subframe type. Figure 4 shows the data format of a MADI channel. Each channel comprises 32 bits and they are identical to AES3 data subframes except for the 4 first bits:

- Bit 0 indicates the first channel transmitted in a sample period and it is used for synchronization.
- Bit 1 indicates if that channel is active or inactive.
- Bit 2 is the type of subframe (A or B) for compatibility with AES3.
- Bit 3 indicates the first frame of an AES3 block.

The next 24 bits are for audio data represented in two's complement, with the Most Significant Bit (MSB) transmitted last (position 27). The last 4 bits are the validity (V), user (U), status (C) and parity (P) of the AES3.

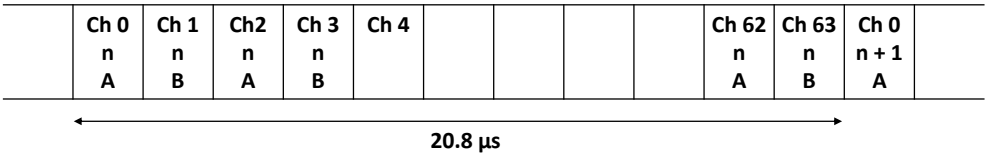


Figure 3: MADI frame with 64 channels at 48 kHz.

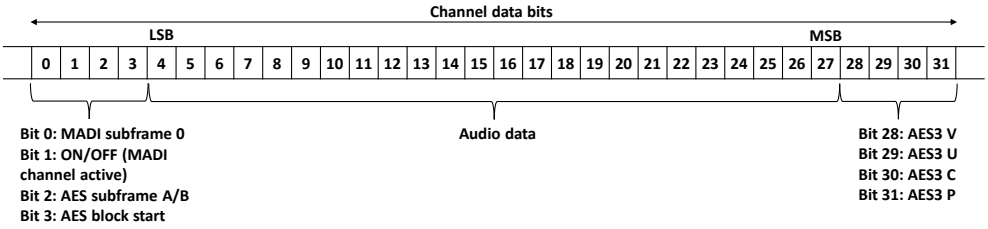


Figure 4. MADI channel data format.

Initially, the hardware that implemented the MADI interface was based on standard silicon chips, but they became obsolete and were discontinued. Currently there is no dedicated chip that implements a MADI interface, so it is typically implemented on an FPGA, which is one of the drawbacks of MADI.

Regarding the transport mean, coaxial cables are cheaper than fiber optic cables, but they are usually limited to distances of about 100 m, while optical cables can reach 10 km.

MADI is an excellent solution to transfer multichannel audio data between two devices. Some manufacturer designed devices that can be daisy-chained, but this is not standardized. Another advantage is that MADI does not need a management software stack and the only settings required are the number of channels, the sampling frequency, and the clock source. MADI is broadly employed in recording studios to connect the different devices. An application example can be seen in Figure 5, where MADI is employed to exchange audio between the mixing console and the preamplifiers, DAC, and a PC. Both daisy-chain and point-to-point connections are shown.

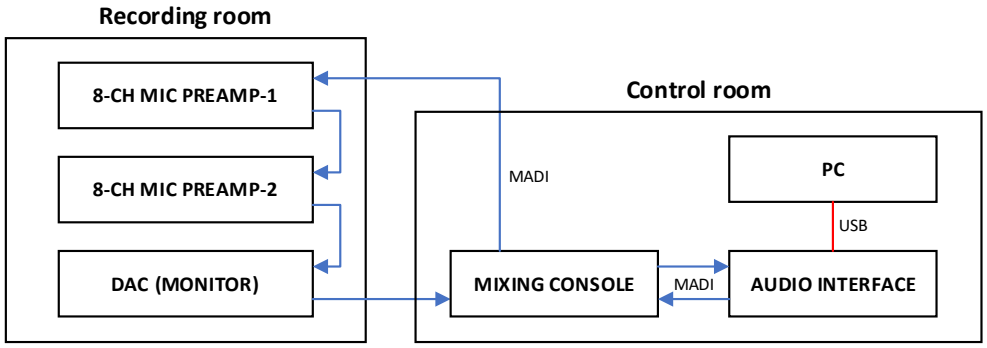


Figure 5. Example of a MADI application in a recording studio.

1.2.2 Audio Video Bridging (AVB)

Audio Video Bridging (AVB) is a set of protocols developed by the IEEE 802.1 standards committee to improve the performance of the already existing and diffused ethernet networks in order to carry audio and video. It is an audio-over-ethernet solution, as it works on the layer 2 of the OSI model. AVB networks have been then extended to Time Sensitive Networks (TSN) since they are employed even outside the audio/video (A/V) field (e.g., automotive, and industrial). To carry time sensitive data, AVB networks offer three essential features:

- Precise timing and synchronization (IEEE 802.1AS-2020 [9]).
- A mechanism to reserve the necessary network resources (IEEE 802.1Qat-2010 [10]).
- A bridging and queueing mechanism to meet the low latency requirements (IEEE Qav-2009 [11]).

These three features aim to fill the lack of timing and synchronization of standard ethernet networks, since they do not have the concept of time, delays are too high, and data can be lost in case of a network congestion.

Much of the AVB functionalities can be implemented in software, but some require hardware support to meet specifications. For example, a *talker* device (i.e., a device that transmits data on the network) must limit the AVB traffic to less than or equal the allocated bandwidth. This feature may be implemented by a packet queueing mechanism or by a Media Access Control (MAC) that supports time-triggered transmission of data. Instead, *Listeners* (i.e., devices that receive data) have a time constraint in presenting samples to the media application level. As another example, the time synchronization protocol requires a timestamping mechanism with an accuracy of tens of nanoseconds. This cannot be achieved by general purpose CPUs but may require hardware timestamping in the ethernet MAC hardware. There are several combinations of hardware and software implementations to develop an AVB device, one can be seen in [12].

The synchronization method described in the IEEE 802.1AS is based on the IEEE 1588-2019 [13] (usually referred to as Precision Time Protocol, PTP), but it extends it with additional specifications (differences are explained in [14] and in section 7.5 of [9]). Since the IEEE 802.1AS supports different media (i.e., IEEE 802.3 full-duplex ethernet, IEEE 802.11, coordinated shared network CSN, and IEEE 802.3 passive optical network EPON) it is usually referred to as *generalized PTP* (gPTP). The aim of this protocol is to create a common notion of time between the devices in the network by periodically exchanging timing information packets. The clock of devices within a gPTP domain must be *syntonized* and *synchronized*. As stated in [15], "*Two clocks are said to be syntonized when their oscillators are precisely the same in frequency"* and "*Two clocks are said to be synchronized when they both agree precisely on what time it is at any moment"*.

The devices in an AVB network can be either *bridges* or *end stations* (or endpoints). As defined in [9], a bridge can be *"either a MAC Bridge or a VLAN-aware Bridge"*, while an end station is a device *"...that acts as a source of, and/or destination for, traffic carried on the LAN or MAN"*. It must be pointed out that transport of timing information happens hop-by-hop. Bridges do not just forward timing information to endpoint devices, but they participate in the synchronization mechanism. In addition, gPTP (as opposed to PTP) does not support links through network switches that do not participate in the synchronization mechanism. This is the reason

why in an AVB network, all the switches must support gPTP. Since this type of switches is more expensive and not so widespread, this feature limited the diffusion of AVB systems in professional audio applications, even if it improves clock synchronization performance. The mechanism to discover devices that support gPTP is the *path delay* measurement and it will be described later. Another limitation of AVB is the operation layer. Indeed, AVB operates at layer 2, thus traffic is not routable between different networks without specialized gateways.

Within a gPTP domain, a synchronization hierarchy is established using an algorithm based on the Best Master Clock Algorithm (BMCA) of the IEEE 1588-2019. This algorithm allows to select one device as the *Grand Master* of the domain. The Grand Master will provide the timing signal called *Grand Master Clock*, to which all devices will synchronize. In addition, the algorithm establishes the network port roles, which can be *master*, *slave*, *passive* or *disabled*. An example of an established clock hierarchy can be seen in Figure 6. A single Grand Master is present and timing information is propagated through the domain starting from it. Each device has a single slave port, which receives synchronization packets from a master port of another device. The packets are then transmitted on any master ports of the device.

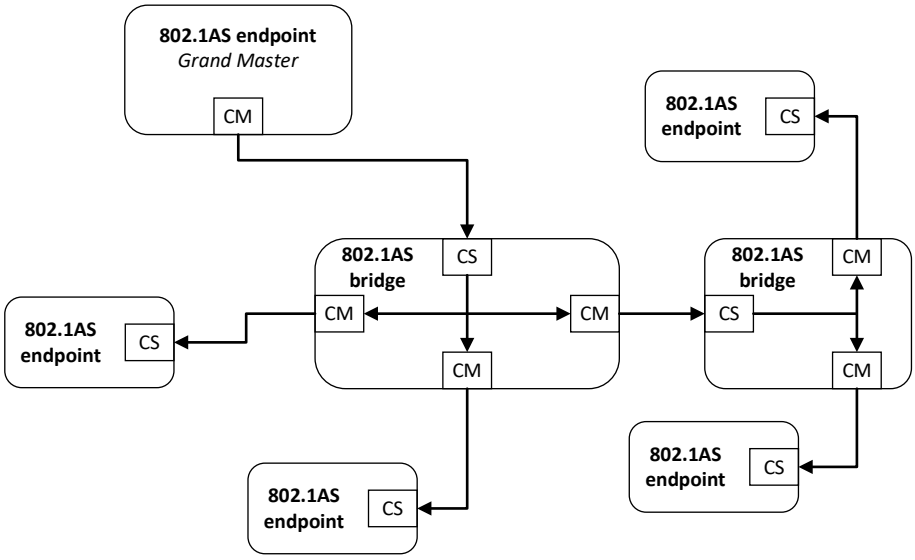


Figure 6. Example of a clock hierarchy in a gPTP domain. CM indicates a clock master port, while CS indicates a clock slave port.

The BMCA algorithm is based on the comparison of the performance of the devices clock, which are shared across the network by transmitting *announce* messages. When the system is initialized, each node that is *master-capable* starts sending announce messages containing its clock parameters. When a device receives an announce message, it compares the received clock parameters with its own. In case the received parameters are better, the node goes into the slave state and stops sending announce messages. A node in the slave state continues receiving announce packets, but it compares them with its currently chosen master. If the received parameters are better, the slave changes the selected master. Eventually, the best Master Clock is chosen. See section 10.3.2 of IEEE 802.1AS for the description of the fields transmitted in the announce packet.

The synchronization mechanism relies on two time intervals: the *residence time* (in a device) and the *path delay* between two devices. The residence time can be easily measured since it is implementation dependent. The path delay is the measure of the propagation delay between each device and in case of full-duplex ethernet as media, this measure is performed by using the *peer delay* mechanism of the IEEE1588-2019. Timestamped packets are exchanged, and the devices use these timestamps, in addition to the packet arrival and departure timestamps, to measure the propagation delay. A diagram of this packet exchange is depicted in Figure 7 (adapted from [14]). The device that wants to measure the path delay is called *peer delay initiator* and it starts the measure by sending a packet called *Pdelay_Req*. Then, it timestamps the time of departure (t_1). The other device, the *peer delay responder*, receives the packet at the time t_2. Then, it responds to the initiator with a message called *Pdelay_Resp*, containing the timestamp t_2. Once again, the time of departure of this packet (t_3) is saved by the peer delay responder. The initiator saves the time of arrival of the *Pdelay_Resp* (t_4) and finally the responder sends a *Pdelay_Resp_Follow_Up* message containing the timestamp t_3. Finally, the initiator knows all the four timestamps and it can calculate the propagation delay, by assuming a symmetrical link.

The propagation delay is measured as:

$$p_{delay} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} \tag{1.1}$$

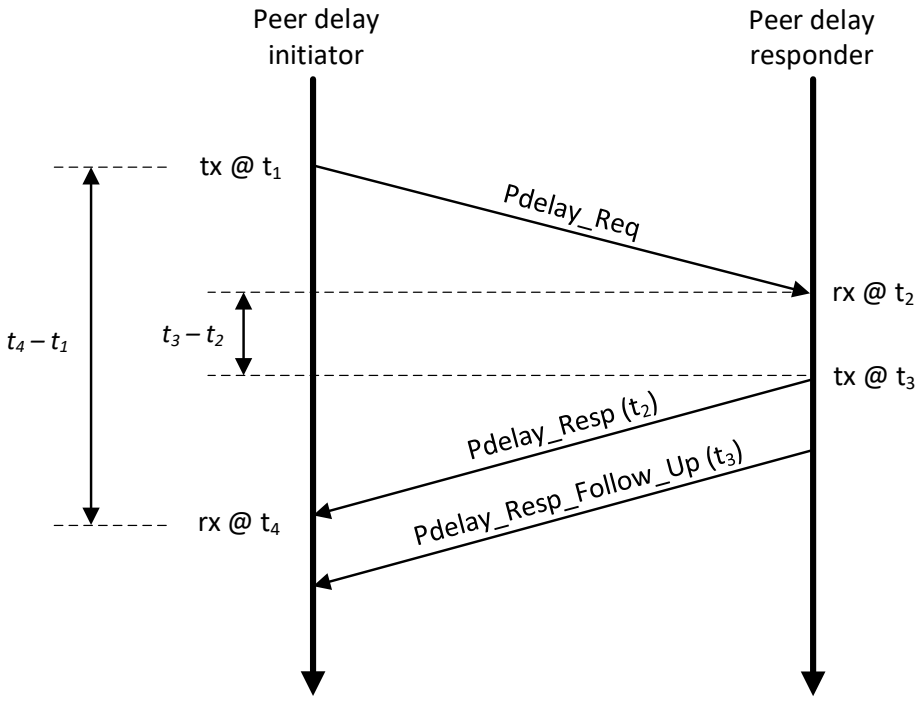


Figure 7. Diagram of the packet exchange during the path delay measurement.

Note that t_3 and t_2 are measured on the time base of the responder, while t_1 and t_4 are referred to the time base of the initiator. The propagation delay is accurate only if both time intervals are referred to a common time base, that is, the clocks of the two devices are syntonized. For example, interval $t_3 - t_2$ can be referred to the initiator time base by multiplying it by the measured rate ratio of the initiator relative to the responder. On the contrary, interval $t_4 - t_1$ can be referred to the responder time base by multiplying it by the measured rate ratio of the responder relative to the initiator. The propagation delay measurement can also be referred to the Grandmaster Clock time base, by multiplying each term by the rate ratio of the Grandmaster Clock relative to the time base in which the term is expressed. For example, an initiator can measure the rate ratio by using the departure and arrival times of *Pdelay_Resp* messages. The propagation delay is used for two purposes: to determine if a device can participate in the gPTP domain and to be employed for the transport of synchronization information.

The path delay calculation mechanism works only if the clocks are syntonized over the network, and the timestamps are as accurate as possible. In case of full-

duplex ethernet networks, clock syntonization is carried out by using *Sync* and *Follow_Up* messages. This mechanism is equivalent to the peer-to-peer Transparent Clock described in IEEE 1588-2019. A time-aware device periodically receives messages on its slave port. The default period is called *Sync interval* and it is equal to 125 µs. Syntonization information is obtained from the arrival timestamps of Sync message and from the data contained in the Follow_Up message. Figure 8 (adapted from [9]) shows an example with three adjacent time-aware systems, indexed *i*-1, *i*, and *i*+1.

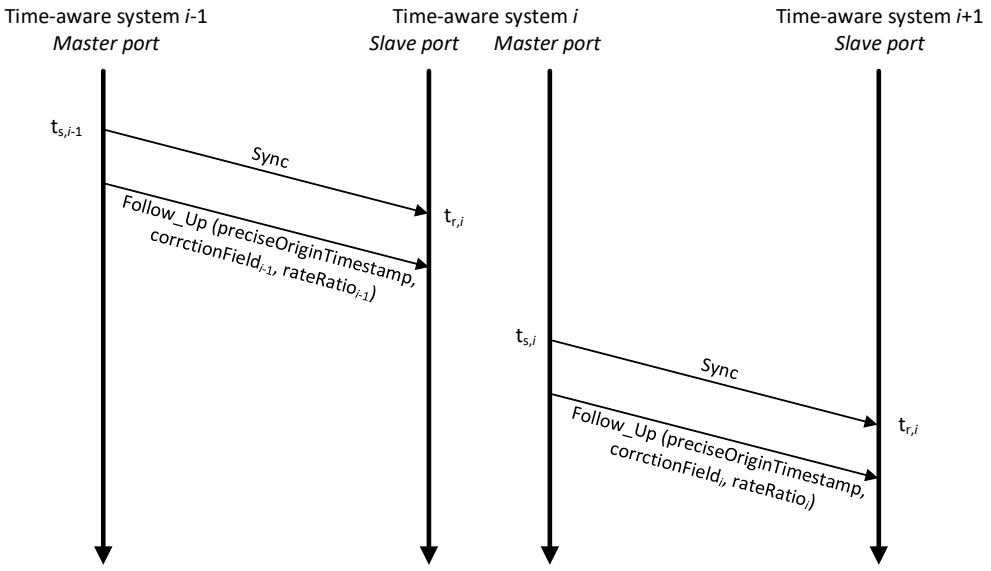


Figure 8. Transport of time-synchronization information.

First, the system *i*-1 sends a Sync message to the system *i* at the time $t_{s,i-1}$, which is received by the system *i* at the time $t_{r,i}$. Later, the system *i*-1 sends the Follow_Up message, which contains 3 fields:

- *preciseOriginTimestamp*: it contains the time of the Grand Master Clock when it was sent the original Sync message.
- $correctionField_{i-1}$: it contains the difference between the synchronized time when the Sync message is sent (in this example it is $t_{s,i-1}$) and the *preciseOriginTimestamp*.
- $rateRatio_{i-1}$: it is the ratio of the Grand Master Clock frequency to the frequency of the local system.

At the time $t_{s,i}$, the system i sends a Sync message to the system i+1, which is then followed by a Follow_Up message. This time the fields of the Follow_Up message are set as follow:

- *preciseOriginTimestamp* is set equal to the *preciseOriginTimestamp* contained in the Sync message sent by system i-1.
- *correctionField* is set equal to the sum of the previous *correctionField* received on the slave port, the propagation delay measured by the slave port, and the time interval $t_{s,i}$-$t_{r,i}$ multiplied by the newly computed *rateRatio* (i.e., the residence time of system i).
- *rateRatio* is set equal to the previous *rateRatio* received on the slave port multiplied by the current neighbor rate ratio measured by the slave port.

Devices have now a common notion of time. However, to exchange audio data, they must also share the *media clock*, which is the clock used to synchronize the sampling and playback devices, namely the sample rate. This feature is defined in the standard IEEE 1722-2016 [16]. Audio streams must contain media clock information by means of timestamps, which will then be converted to the gPTP timebase by each device. A talker may timestamp each sample and send them to the listener, which must be capable of recovering the media clock information. The listener timestamps at regular intervals its media clock with the gPTP reference clock and then it uses this measure to derive the local media clock period. An important feature introduced by the IEEE 1722-2016 is the *presentation time*, namely a point in time (referred to the gPTP) at which audio must be played. It is inserted by talkers in every data packet, and it allows different listeners to reproduce samples at the same time.

Other essential features of AVB are traffic shaping and bandwidth reservation. Used together, they guarantee a communication with a deterministic low latency (for AVB *class A* devices the default value is 2 ms, see [16]) and low jitter. Traffic shaping consists in evenly distribute in time packet transmission, so to avoid bursts of traffic. Depending on the AVB traffic class and Quality of Service (QoS) parameters, streams are prioritized with tags (as described in [11]). However, traffic shaping alone cannot guarantee a deterministic low latency because network resources may not be available. AVB networks have a bandwidth reservation mechanism, called Stream Reservation Protocol (SRP), which is defined in [10]. Before the beginning of a transmission, a talker stream must be requested by one or more listeners and the required bandwidth must be allocated. Bandwidth availability is verified through the entire path that connects talkers and listeners and so bridges must participate in this

process too. SRP consists in a series of messages exchanged between talkers and listeners. Firstly, the talker sends a *talker advertise* message, which includes the source and destination MAC addresses, a talker unique ID, the QoS requirements (e.g., traffic class) and the accumulated worst-case latency, which is recalculated at every bridge. Bridges verify bandwidth availability and if it is sufficient, they propagate the talker advertise message. Otherwise, they propagate a *talker failed* message. Eventually, the listener receives the talker advertise message and it will know if the requested bandwidth is available and the latency of the path. Then, it responds with a *listener ready* message, which will be forwarded by the bridges to the talker. When bridges receive this message, they also lock the resources previously requested. Lastly, when the talker receives the listener ready message, it starts transmitting the stream.

Finally, it must be pointed out that other higher layer protocols are needed for device interoperability. For example, the IEEE1722.1-2013 [17] defines procedures for device discovery, connection management and device control for A/V systems that communicates with IEEE1722-2016 streams. However, this standard offers several options that can be implemented, and some may be contradictory, thus limiting device interoperability. The lack of a precise application layer, together with other issues when connecting systems of different manufacturers (e.g., media clock and stream format), brought to a limited diffusion of AVB.

In 2018, the Avnu alliance [18], a consortium of professional audio and video manufacturers, announced Milan. Milan is a set of specifications built on top of the AVB standards, with the aim to define device requirements at network and application layers (e.g., media clock, stream format, redundancy, and software control) to improve device interoperability.

AVB was initially designed for professional A/V applications, but due to the potential of AVB/TSN networks, it has been employed also in automotive and industrial applications. Automotive field has different requirements compared to professional A/V (e.g., lower-prices, deterministic or real-time response for safety reasons). However, the most evident difference between professional A/V and automotive systems is that the latter have static configuration and topology, well known when the system is designed. These differences led to the definition of an AVB automotive profile (see Appendix A of [15] and [19]). The principal changes concern:

- Pre-configured gPTP clock tree for fast startup and reduced protocol traffic.

- Static SRP – Pre-configured stream and class reservations.
- Optimized traffic classes for automotive infotainment.
- Added status reporting.
- Added exception and diagnostic reporting.
- No 1722.1 (AVDECC) support.

Figure 9 shows a possible configuration of an AVB system in a car, where AVB is employed to transport both video and audio data. In this example, the head unit exchange audio with the amplifier by means of an ethernet cable. The head unit also acquires signals from cameras, and it may reproduce video on the two rear displays. It is possible to add an AVB compliant switch in case the head unit does not provide enough ethernet ports or the length of the harness can be optimized.

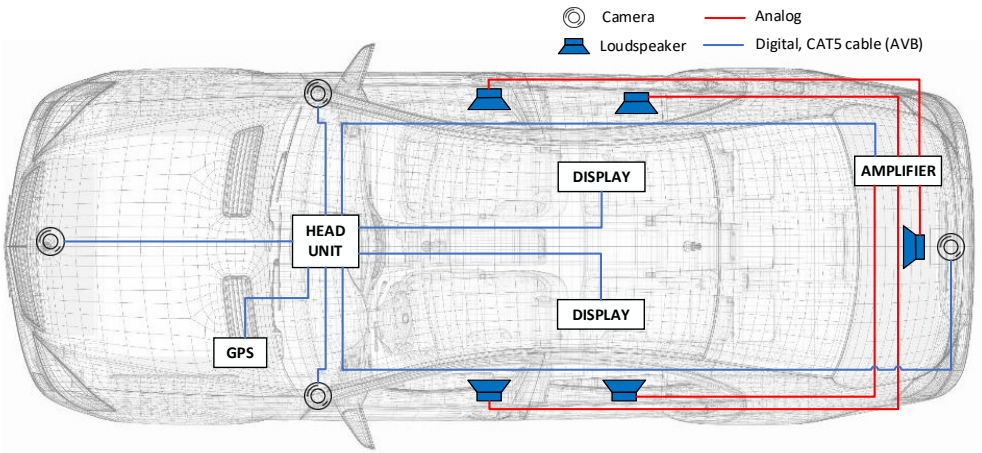


Figure 9. Example of an AVB system for automotive applications.

Figure 10 shows instead a possible configuration of an AVB system for professional audio applications. Several devices can be connected in the same network, namely mixing consoles, personal monitor mixers, stage boxes, and PCs. Depending on the network topology and the number of devices, multiple AVB compliant switches can be added. Several manufacturers developed their AVB-based systems following the architecture shown in Figure 10. However, these systems lack of interoperability mainly because of the absence of an application software layer needed to configure them. Milan tries to fill this gap, but as it will be explained in the next sections, other protocols were used in the last years to overcome this issue.

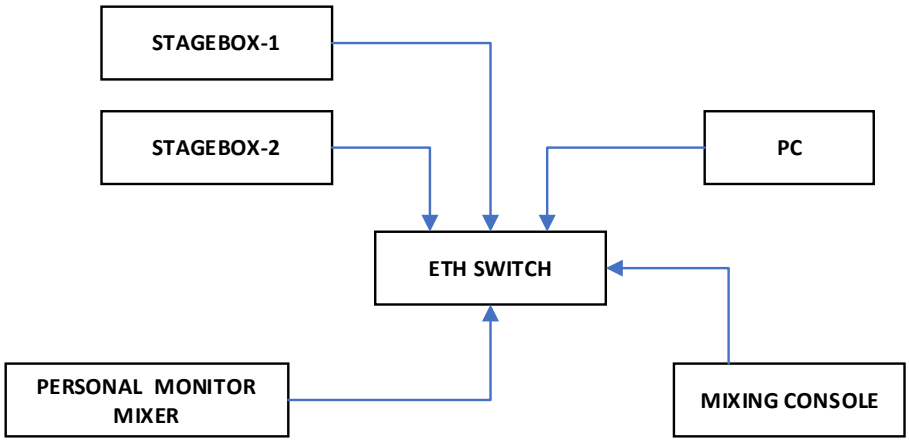


Figure 10. Example of an AVB system for professional audio applications.

1.2.3 Dante

Dante is a network protocol developed by Audinate in 2006 [20] for audio distribution in professional or consumer applications. As opposed to AVB, Dante is a proprietary audio-over-IP protocol. Dante can be implemented in products by purchasing from Audinate the hardware modules, chips, or software. Adopting a proprietary solution may help in reducing development costs, while limiting customization and implementation choices. Dante shares with AVB some concepts (e.g., clock synchronization and prioritization), but it implements them in a different manner.

Clock synchronization is carried out by using version 1 of IEEE1588 (IEEE1588-2002), so standard ethernet switches can be employed when connecting Dante devices. A latency value is set for each device (from 0.25 ms up to 5 ms), and Audinate suggests, as a general rule, to add 100 µs per switch. Devices with different latency values can coexist in the same network, but if the transmitter and the receiver of a stream have different latencies, the higher value will be used. In addition, multicast streams have a fixed latency value of 1 ms. Dante allows for *unicast* or *multicast* streams. The former is used when audio need to be exchanged between two devices, while the latter is used when a single source sends the same audio channels to multiple receivers. In contrast to AVB, where bandwidth is reserved, Dante requires a QoS management to guarantee that packets are delivered in time. Dante employs Differentiated Services (DiffServ), also known as Differentiated Services Code Point (DSCP), which prioritizes packets by using a field of the IP header.

Audio format supported is 24 or 32 bits and the sample rates are from 44.1 kHz up to 192 kHz. The maximum number of channels depends on the Dante platform employed. If using, for example, a Dante Brooklyn II, the maximum number of channels is 64 inputs and 64 outputs at 44.1 kHz or 48 kHz. Finally, Audinate has started also supporting video distribution in its products.

Regarding the application level, Audinate developed a PC software called Dante Controller, which allows the user to configure the signal routing and the devices settings. Routing and settings are stored in the device's memories, so that Dante Controller is not essential for the network to work. Other useful applications are the Dante Virtual Soundcard, which turns a PC in a Dante-powered device, able to manage up to 64 input and 64 output channels, and the Dante Domain Manager, whose main purpose is to organize a Dante system into independent groups of devices called Dante Domains and to protect system configuration by means of user accounts with different privileges.

In 2015, Audinate added AES67 support in its products, to achieve interoperability between products from different audio network solutions. When using AES67, the devices automatically negotiate a PTPv2 clock (defined in IEEE1588-2019), and they exchange audio using multicast AES67 audio streams. A Dante device will discover the AES67 multicast streams only if the Session Announcement Protocol (SAP) is employed. Anyway, Audinate's advice is to employ just one network solution for a system and to use AES67 only to connect different network solutions when interoperability is required [21].

Dante is broadly employed for installations such as clubs or conference rooms. Figure 11 shows a possible architecture for a conference room where the audio transport occurs via Dante. As for an AVB application, all the devices are connected by an ethernet switch. The PC can play or record audio thanks to the Dante Virtual Soundcard and it can be employed to route the audio channels between the devices with the Dante Controller. The Dante adapter converts line input/output or Bluetooth signals to the Dante network.

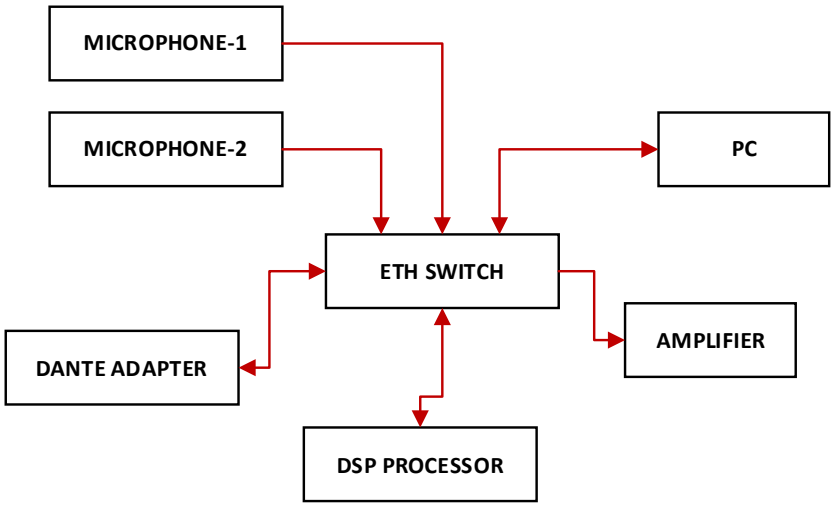


Figure 11. Example of a conference room system employing Dante.

1.2.4 Ravenna

Ravenna is an open technology developed by ALC NetworX for the real-time distribution of audio and other media content in IP-based network environments. Its main features are low latency, full signal transparency and high reliability. In contrast to Audinate, ALC NetworX developed just the technology, while the product implementations are carried out by individual manufacturers.

As stated in [22], *"One way of looking at RAVENNA is to see it as a collection of recommendations on how to combine existing standards to build a media streaming system with the designated features"*. The adoption of widespread standard network protocols, defined and maintained by international standardization organizations (e.g., IEEE, IETF and AES) allows Ravenna to operate on existing network infrastructures.

As for the other audio network technologies, all the nodes in the network must be synchronized to a common wall clock. Ravenna achieves this through the IEEE1588-2019, also known as PTPv2 Precision Time Protocol, which is operated on the IP-level. The media clock can be synthesized by default from the PTPv2 reference time, or optionally, from a stream. The possibility to generate the media clock from the PTP absolute time is one of the major differences with AVB [23], [24], which instead, reconstructs the media clock from an incoming media stream. The advantage of deriving media clocks from absolute time is that multiple sampling

frequencies can be simultaneously supported in the network, and they are locked to each other (at least a sample rate of 48 kHz must be supported). Another difference from AVB is that Ravenna does not have the concept of *presentation time*. Data packets are instead tagged with the *generation time*, and the presentation time is then calculated by the receiver.

Streaming is supported both in unicast and multicast mode, and the protocol used to transport media data is the Real-time Transport Protocol (RTP) [25], which uses User Datagram Protocol (UDP) over IP. 16-bit and 24-bit payload formats are supported, and 32-bit support is recommended too. Audio streaming is managed by using the Real Time Streaming Protocol (RTSP) and the Session Description Protocol (SDP). RTSP is an application-level protocol that is used to establish and control streaming sessions between a transmitter (an RTSP server) and a receiver (an RTSP client). SDP instead is a protocol commonly employed in media streaming applications to describe the media streams (i.e., stream name, payload formats, number of channels, access information, etc.). The QoS mechanism employed by Ravenna is the DiffServ as it is widely supported by most modern managed switches. Finally, Ravenna can optionally support full network redundancy.

As for Dante, also Ravenna is compliant with AES67, or more precisely, AES67 can be seen as a Ravenna profile since they share most of the operating principals [26]. In particular, Ravenna and AES67 adopt the same protocols for synchronization and transport, but the AES67 packet setup and the payload format are a subset of the Ravenna options.

Ravenna can be employed as an alternative to Dante, even though it is more adopted in broadcasting thanks to its compatibility with SMPTE ST 2110, a standard for data transport over IP networks commonly used in broadcasting.

1.2.5 AES67

Since the diffusion of audio network protocols, the interoperability has been one of the main issues. In fact, even if sharing the same transport mean, these protocols rely on different standards, and so interoperability is not guarantee. To address this problem, in 2013 the Audio Engineering Society published the standard AES67 [27], which defines the protocols and functions to support in order to achieve device interoperability. Instead of inventing new protocols, the approach during the

development of AES67 was to use existing standards in an interoperable manner. The main functionalities addressed by AES67 concern:

- Synchronization and media clocks
- Transport, encoding and streaming
- Session description
- Connection management
- Discovery

The standard chosen to synchronize devices is the IEEE1588-2019 (PTPv2). An exception is made for AVB devices since they use the IEEE802.1AS. An annex of AES67 offers synchronization methods for these devices. In addition, the IEEE1588-2019 also defines a grandmaster election algorithm (BCMA). The media clocks are derived from the network clock and advanced at the same rate. Three sampling frequencies are supported: 44.1 kHz, 48 kHz, and 96 kHz.

The media data is transported across the network using IP version 4, in both unicast and multicast streams. The multicast traffic is supported even by unmanaged switches, but only managed switches, correctly configured, allow to avoid network flooding by employing the Internet Group Management Protocol (IGMP). IGMP is used to forward multicast packets only to their designation port. For this reason, AES67 devices must support IGMP. QoS is another fundamental feature that must be supported to prioritize traffic. The DiffServ method has been chosen and each AES67 device must implement it. At least three classes of traffic must be supported (see Table II, from [27]). Data is transported by means of RTP over UDP. Two payload formats can be used:

- L16: 16-bit linear format
- L24: 24-bit linear format

Another important parameter is the packet time, namely the real-time duration of the media data contained in a packet. In general, a short packet time means a lower latency, but it introduces overhead and higher data rate. On the contrary, a long packet time leads to higher latency and high-capacity buffers, while reducing the traffic overhead in the network. To address interoperability requirements, AES67 devices must support 1 ms packet time. This means that at 44.1 kHz and 48 kHz a packet will contain 48 samples, while at 96 kHz it will contain 96 samples. Of course, to improve interoperability, other packet times are recommended (see Table III, adapted from [27]).

Table II - QoS classes and DiffServ associations

Class name	Traffic type	Default DiffServ class (DSCP decimal value)
Clock	IEEE 1588-2008 Announce, Sync, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up packets	EF (46)
Media	RTP and RTCP media stream data	AF41 (34)
Best effort	IEEE 1588-2008 signaling and management messages. Discovery and connection management messages	DF (0)

Regarding the maximum number of channels per stream, it is limited by the packet time, the encoding format and the network Maximum Transmission Unit (MTU). AES67 defines the number of channels per packet that must be supported from 1 to 8. As for the packet time, a device may support streams with more channels (see Table IV, adapted from [27]). Finally, AES67 defines the sender and receiver buffer dimensions as follow: the receiver buffer capacity must be at least 3 packet times, while the sender must transmit data at the nominal rate with a variation of no more than 17 packet times or 17 ms whichever is smaller.

Table III – AES67 required and recommended packet times

Packet time	Samples per packet (at 44.1/48 kHz)	Samples per packet (at 96 kHz)	Notes
125 µs	6	12	Compatible with class A AVB transport.
250 µs	12	24	High-performance, low-latency operation. Interoperable with class A and compatible with class B AVB transport.
333 µs	16	32	Efficient low-latency operation.
1 ms	48	96	Required common packet time for all devices adhering to this standard.
4 ms	192	n.a.	For applications desiring interoperability with EBU Tech 3326 or transport over wider areas or on networks with limited QoS capability.

Table IV – Examples of maximum channel capacities per stream

Format, sampling rate	Packet time	Maximum channels per stream
L24, 48 kHz	125 µs	80
L24, 48 kHz	250 µs	40
L24, 96 kHz	250 µs	20
L24, 48 kHz	1 ms	10
L24, 48 kHz	4 ms	2

All these settings must be exchanged and agreed between two devices before the audio streaming starts. The format used to describe the streaming settings is the SDP, but to achieve interoperability, AES67 imposes additional SDP requirements and recommendations about packet time, RTP clock, media clock and clock source, and payload types. Although, AES67 clearly defines the SDP attributes to guarantee interoperability, it does not specify a protocol to exchange this information. The device discovery, which allows to know the network participant and the available streams, has been intentionally excluded too, even though a list of some widely used protocols has been given (e.g., Bonjour, SAP and Networked Media Open Specifications – NMOS).

The last feature that AES67 addresses is the connection management, which "*is the procedure and protocols used to establish a media stream between a sender and one or more receivers*" [27]. AES67 specifies the Session Initiation Protocol (SIP) as the connection management protocol for unicast connections. Usually, SIP is employed in a client-server fashion, where servers can be located anywhere in the network, and they can be reached by the devices. However, in small networks, the server installation may not be convenient and so AES67 specifies that the devices must support a serverless mode (peer-to-peer), even though they should work as well in a network featuring servers. Multicast streams management can be accomplished without a connection management protocol. Instead, a receiver can be connected to an existing multicast streaming by using the IGMP protocol (if it knows the streaming information, which is advertised during discovery).

Figure 12 shows an example of application where AES67 enables the interoperability between Dante and Ravenna systems. Inside the stadium, the equipment employed for audio acquisition and reproduction employs Dante, while the broadcast adopts Ravenna. In these conditions, it is not possible to exchange audio between

the two systems. Usually, MADI can be employed for this purpose, but additional hardware is required for the conversion. On the contrary, thanks to AES67 it is possible to exchanging audio on ethernet cables without the need for additional conversions and hardware.

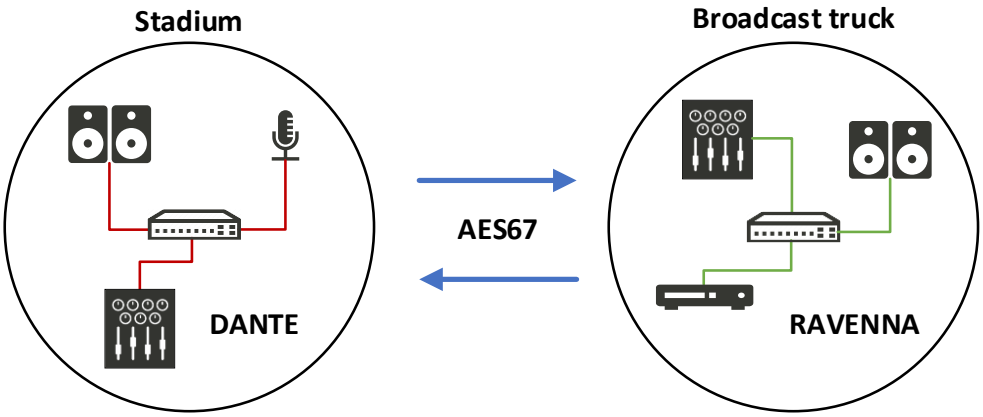


Figure 12. Example of how AES67 is employed to enable interoperability between systems.

1.3 Comparison

The list of protocols of the previous section does not presume to be exhaustive, as many other digital audio protocols have been used during the years or they are still employed. For example, there are some proprietary protocols (e.g., Livewire and Q-LAN) that were not mentioned because mainly employed in closed ecosystems.

Many characteristics shall be considered for a protocols comparison, see Table V for a summary of the main differences. Firstly, digital audio protocols can be divided by the technology on which they are based into *network* protocols (i.e., AVB, Dante, Ravenna and AES67), which are typically based on IP networks, and *point-to-point* protocols (i.e., A^2B and MADI). Even if manufacturers are increasingly adopting network technologies, point-to-point protocols offer several advantages, and they are particularly useful when a high number of channels have to be transferred between two devices. MADI, for example, is still employed in broadcast and live sound applications thanks to its low latency and simplicity [28]. On the contrary, network protocols offer a higher flexibility at the cost of ease of use and

performance. The channels can be routed to any device available in the network, without the need to change the wirings in case multiple devices require the same data. Moreover, the network can be easily expanded by adding new connections to the switches. On the other hand, network protocols typically have a higher latency than point-to-point protocols, and they may be more susceptible to audio failures due to erroneous switch configuration or network congestions. In addition, there are differences on the type of switch that can be employed. AVB requires AVB-capable switches, namely switches that implement gPTP and SRP. Dante, Ravenna and AES67 can be theoretically used in ordinary networks with unmanaged switches. However, there are several scenarios where managed switches are preferred or even mandatory, in order to manage the traffic with QoS and IGMP protocols. Examples may be medium and big installations where a high number of devices are present in the network, installations where there is significant non-audio traffic in the network or in applications where several multicast streams are required. Another crucial characteristic is *licensing* [29]. In fact, protocols can be developed by companies as *proprietary* or *open* standards. Licensing greatly affects the adoption process of a protocol when designing a system, since both types have pros and cons. Proprietary protocols, such as A^2B and Dante, are typically developed and controlled by a single company, which may sell devices or grant licenses to companies who will create products supporting that technology. When adopting proprietary technologies, customers are bounded to them, and interoperability with other solution is not guaranteed or even not possible to achieve. Moreover, if the owner stops the development of that technology, customers must adopt other solutions. On the other hand, the success of a proprietary solution guarantees a high-quality level for both performance and support, and in addition, it incentivizes competitors to create rival solutions. Open standards (e.g., MADI, AVB, Ravenna, and AES67) are generally developed by standardization organizations or consortiums. Since many people with different backgrounds and goals participate in the creation of a technology, this typically ensures greater interoperability compared to proprietary solutions, and as a consequence, it facilitates broader adoption.

Then, *topology* and *transport mean* shall be considered. There exist several network topologies, but the most common are bus, star, daisy-chain, ring, tree, spanning-tree, and mesh [30]. Dante and Ravenna can work on any network topology, while AVB is the only protocol that employs a spanning-tree topology, namely any physical network topology is allowed, but switches will deactivate their ports to

break loops and they will forward data only to the required nodes. The transport means of these protocols are ethernet or fiber optic cables. The standard version of MADI defines only point-to-point connections, nevertheless there are implementations that allows to connect devices in daisy-chain. The transport means are coaxial cables or fiber optic cables. A^2B nodes are instead connected in daisy-chain by default and the transport mean is an UTP cable.

Regarding *performance*, data is summarized in Table V. The considered parameters are maximum channel capacity, latency, maximum bit depth, and maximum sample rate. Other important features are synchronization and fault tolerance. A^2B and MADI are synchronous protocols, while AVB, Dante, Ravenna and AES67 rely on IEEE1588 protocol (AVB on IEEE802.1AS) to achieve synchronization. Fault tolerance is an essential feature in professional audio applications, as in many scenarios audio interruption due to losses or interruptions is not acceptable. For this reason, Dante and Ravena allow to create a secondary network (separated from the primary network) that will be employed, without audio glitches during the transition, when the primary network fails. Even if not standardized, AES67 and AVB devices may employ a redundant network, but it is implementation dependent. The same applies to MADI, where several devices offer a dual connection mode, consisting in copying data to both links. On the other hand, A^2B has a fault detection feature, but it does not offer redundancy or an optional ring topology. The nodes where the fault occurred can be precisely detected, and the fault can be isolated by disabling downstream nodes. In this way, the nodes upstream the fault remain active and continue to exchange audio without interruption.

Finally, the presented protocols differ by use cases. A^2B was developed for automotive applications, but it offers several advantages also in other application fields [31], and it can be seen as a complementary technology of network-based protocols. For example, in stadiums or shopping centers, A^2B may help in local zones audio distribution, being effective in connecting peripheral devices (e.g., paging microphones, speakers and audio sources) to the backbone network. A similar approach can be applied to teleconferencing systems of meeting rooms, which are typically composed by distributed microphones and speakers. A^2B allows to simplify the design of such devices by using dedicated transceivers requiring minimum configuration effort, thus keeping low the cost of such devices. The other mentioned technologies are mainly employed for professional audio applications. MADI is currently widespread in broadcasting, but it finds application in recording studios and live

sound too. Network-based technologies, namely AVB, Dante, Ravenna and AES67, are mostly interchangeable. All of them are employed for live sound, but only Dante, Ravenna and AES67 are suitable for installations. In fact, in contrast to AVB, they operate at IP level, and they require common network switches, thus making them compliant with the existing network infrastructures. AES67 and Ravenna are being used in broadcasting too, thanks to their open standard nature and their interoperability. In fact, Ravenna is fully compliant with AES67 and SMPTE ST 2110, a standard for data transport over IP networks commonly used in broadcasting. On the contrary, AVB has been adopted by several manufacturers as a transport protocol only within their ecosystems, mainly for live sound. This is due to the particular network requirements and the lack of interoperability guidelines. In addition, AVB is being employed also in automotive, for in-vehicle networking, and industrial fields.

Table V - Technologies comparison

	A²B	**MADI**	**AVB**	**DANTE**	**RAVENNA**	**AES67**
Network-based	No	No	Yes	Yes	Yes	Yes
Topology	Daisy-chain	Point-to-point/ daisy-chain	Spanning-tree	Any network topology	Any network topology	Any network topology
Transport mean	UTP	Coaxial/ fiber	Copper/ fiber	Copper/ fiber	Copper/fiber	Copper/ fiber
Type of switch	n.a.	n.a.	AVB-capable	Managed preferred	Managed preferred	Managed preferred
Licensing	Proprietary	Open standard	Open standard	Proprietary	Open standard	Open standard
Synchronization	Synchronous	Synchronous	802.1AS-2020	IEEE1588-2002	IEEE1588-2019	IEEE1588-2019
Max. channel capacity	32 x 32	64 x 64	Bandwidth dependent	Bandwidth dependent	Bandwidth dependent	Bandwidth dependent
Latency	2 samples	Implementation dependent (typ. 4 samples)	Up to 2 ms (class A)	Configurable 0.25 ms – 5 ms	Configurable 1 ms guarantee	1 ms, others are recommended
Max. bit depth	32	24	32	32	32	24
Max. sample rate	48 kHz	192 kHz	192 kHz	192 kHz	192 kHz	48 kHz, 96 kHz is strongly recom-mended
Fault tolerance	Fault detection. Devices upstream fault continue to work	Dual MADI connections	Not standard-ized Impl. dependent	Redundancy on 2nd network	Redundancy on 2nd network	Not standardized
Uses	Automotive	Broadcast Live Studio	Live Automotive Industrial	Live Install	Live Broadcast Install	Broadcast Live Install

2 Data Transport Architecture

This section will provide the general description of the proposed architecture. The aim of this work is to develop an innovative system to acquire and distribute audio signals. The main features of the developed system are scalability, modularity, full-digital data path, high number of channels, and distributed design. The application fields that can leverage from it are automotive, professional audio, and industrial. Each of these applications has its own pros and cons, but all take advantage of the simple design and wiring of the proposed architecture. The next subsections will present the various architecture parts and the features of the system.

2.1 Proposed Architecture Description

Regardless the application field, the proposed architecture can be divided into three main parts:

1. Peripheral devices
2. Transport layer
3. Processing and protocol conversion

A block diagram of the architecture concept with the three main parts highlighted, is shown in Figure 13. The p*eripheral devices* are the end devices that exchange data through A^2B. Example of peripheral devices are amplifiers, processors, preamplifiers, stage boxes, audio interfaces, or transducers such as loudspeakers, microphones, and accelerometers. Each peripheral device is connected to an A^2B transceiver that acts as a subordinate node and gives access to the bus. A^2B, namely the *transport layer*, is employed to connect the various peripheral devices until the maximum number of subordinate nodes or the bandwidth limits are reached. In some

applications this limit (i.e., 32 channels for each A²B bus) may be too strict, so the architecture has been designed to be modular. The maximum number of channels can be increased by adding more A²B networks.

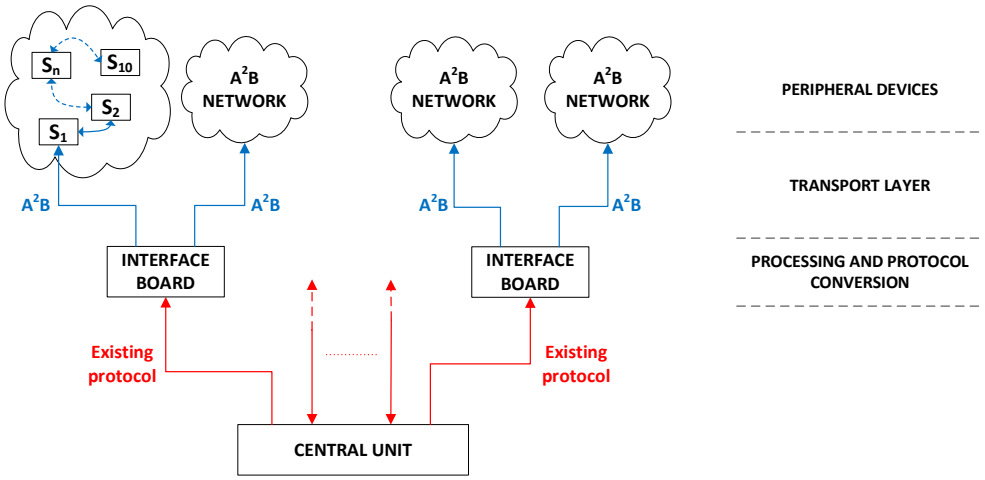


Figure 13. Block diagram of the proposed architecture.

All the A²B networks are then connected to interface boards, namely the *processing and protocol conversion* part of the architecture. The interface boards have three main purposes:

- management of the A²B networks
- signal processing
- protocol conversion

As explained in Section 1.1, A²B does not require active control during normal operation, but it is only needed at the startup of the system, when the nodes on the bus are configured. This task can be carried out for example by a microcontroller on the interface board. Other required operations for A²B management are channel routing and synchronization of different networks. Even if channel routing in a single bus is established during the first configuration of the system, routing across different busses cannot be directly fulfilled by A²B transceivers, but signals must be managed by external devices (e.g., DSPs, microcontrollers, FPGAs). In addition, these devices must keep the networks synchronized, as it will be discussed in detail in the next section. If required by the application, the interface board may also implement signal processing, for example by means of a DSP or an FPGA. Finally, if data must

be exchanged with an existing system, the interface board should also implement a protocol conversion between A^{2}B and the required protocol. Moreover, as several A^{2}B networks can be combined to increase the system capabilities, so different interface boards can be added to the architecture, depending on the implementation details. Some use cases will be presented in Chapter 3.

The presented architecture has been designed to be full-digital. The only A/D and D/A conversions occur in the peripheral devices or in the existing system, which is interfaced with the architecture, thus limiting the number of conversions, and keeping the transport path free of the well-known problems of analog data transmissions. Such problems typically include bulky wiring, noise immunity and Signal to Noise Ratio (SNR) degradation. In addition, this architecture inherits great versatility thanks to its scalability and modularity. The number of devices, and so channels, can be changed depending on the application just by adding nodes to the A^{2}B networks or interface boards. This feature allows to easily realize systems with a massive number of channels, which are more and more required. Other fundamental characteristics are the low latency and synchronization. A^{2}B allows to keep latency as low as possible, since it is equal to just 2 samples (about 42 µs at 48 kHz) regardless the number of channels and nodes involved in the network. The interface boards contribute to increase the overall architecture latency by implementing signal processing and protocol conversion. Of course, the precise value of the introduced latency is implementation dependent, but it can vary between few samples, in case of a simple conversion to MADI, to 0.5 ms or 2 ms for network protocols. Nevertheless, it should be pointed out that the very low latency value of A^{2}B leaves enough resources for the signal processing algorithms, which are necessary in current audio applications (e.g., Active Noise Cancellation - ANC, echo cancellation).

From another point of view, A^{2}B can be seen as TDM over distance, thus overcoming the limited distance reachable by TDM. TDM is by far the most widespread interface used to exchange multiple audio channels on a single data line and it is mainly employed on Printed Circuit Boards (PCBs) for inter-IC communication or in short wirings within electronic systems. The A^{2}B transceivers convert the TDM signals coming from the peripheral devices to the A^{2}B format and vice versa, providing two main advantages. The first is that A^{2}B signals can be transmitted up to several meters thanks to their differential nature, which increases the noise immunity. The second is that the clock is not transmitted on a separate line, but it is extracted

from the data flow. By doing this, A^{2}B nodes communicate over a single UTP cable, thus simplifying the wiring.

Finally, one of the principal advantages of the proposed architecture is the simple and straightforward design of devices. Thanks to dedicated transceivers, A^{2}B manages bus access, clock, and data distribution. Furthermore, minimum software management is required to configure the network. These characteristics are particularly useful when designing transducers or simple devices. Some applications may require distributing transducers in space and this implies issues in data transfer. The A^{2}B features previously mentioned and the increasingly availability of transducers with digital input/output interfaces (e.g., PDM or TDM microphones and accelerometers) allow to simplify the design process by removing the signal conditioning section and A/D or D/A conversions. Hence, it is possible to develop simple transducers, only composed by the transducer itself and the A^{2}B transceiver, without the need of microcontrollers or other processing units for the management of the device.

Certainly, there are also drawbacks. A^{2}B was designed to satisfy new requirements of the automotive field, but when employed for other applications, it has some limitations to overcome. First of all, the maximum cable length may be a constraint. A network composed by AD242x transceivers reaches a maximum distance of 40 m, while the maximum connection between two nodes is 15 m. AD243x transceivers allow increasing the overall network cable length to 80 m, but the distance between the nodes remains the same. When demanded by the application, this limit can be overcome in two ways. The first one, offered by the proposed architecture, is to employ A^{2}B networks to locally wire the devices and to exploit the interface boards to reach longer distances with other protocols. The second one is to insert in the A^{2}B bus dummy nodes that do not exchange audio data, but act as repeaters. On the other hand, this method can lead to saturate the maximum number of nodes in the network, namely 10 for the AD242x family and 16 for the AD243x.

Another feature which is particularly convenient in automotive applications is the initial configuration of the devices in the network. This assumes to know in advance, and more precisely in the design process, every detail of the network, including how many devices will compose the network, their order, the number of channels to exchange, and their peripherals. When designing an A^{2}B network for automotive applications this level of knowledge is available since the scenario is static. On the contrary, in case of professional or consumer audio applications the scenario is dynamic and there are many variables involved. The nodes may be connected and

disconnected several times during the normal operation of the network. Even the type of device may change and so the channel routing, which is typically configured by the end-user. Hence, in such applications, it is not possible to know in advance the network configuration and a mechanism to manage the dynamic connections should be provided. Section 2.3 describes in detail a dynamic configuration mechanism developed in this work, which makes A^2B plug-and-play.

2.2 Clock Propagation Analysis

This section shows the results of a clock propagation analysis carried out on an A^2B bus that were published during the PhD [32].

As it was explained in Section 1.1, A^2B has an intrinsic mechanism to keep nodes synchronized. Each transceiver has a PLL that locks on a preamble which is sent at the sampling frequency before each superframe. Due to the daisy-chain topology of the A^2B, the clock is propagated along the nodes, and so its quality in terms of jitter and latency degrades as the number of nodes increases. This suggests two facts. Firstly, the clock source of the network, namely the clock fed to the main A^2B node, should be as clean as possible. Secondly, some applications (e.g., beamforming with microphone or loudspeaker arrays) require that transducers are sampled synchronously because different sampling intervals may lead to a deterioration of the system performance. For these reasons, a clock propagation analysis was carried out, with the aim to develop a clock propagation model useful to evaluate the impact of the bus on the array performance.

First of all, some background on jitter and its effects will be provided. Total jitter (TJ) is defined as the deviation of transition edges from their ideal position to the actual one in time domain. It can be mainly decomposed into unbounded random jitter (RJ) and bounded deterministic jitter (DJ), depending on the underlying sources producing them [33]–[36]. RJ is typically described by an unbounded Gaussian distribution, and it is quantified by its standard deviation. However, because of the contribution of the DJ, jitter distribution is not often Gaussian. Moreover, DJ can be expressed in terms of subcomponents, namely the periodic jitter (PJ), the bounded uncorrelated jitter (BUJ), and the data-dependent jitter (DDJ). These components have various origins. Ground bounce and power supply switching noise that couples to data or clock signal lines usually relates to PJ [37]. Crosstalk between aggressor

and victim channels causes BUJ, while asymmetry in rise time and fall time, or by bandwidth limitation of the channel causes DDJ [38].

The amount of total jitter, with its distribution and spectrum, can be measured in time domain and it can be expressed as peak-to-peak or RMS values. Even though, this method is always possible, it can be inadequate due to the different impact of RJ and DJ. Alternative approaches including tail fitting algorithms or FFT analysis will be described in the next paragraphs.

Jitter Decomposition Based on PDF

Let us consider the probability density function of TJ, which is the convolution of the RJ and DJ pdf since RJ and DJ are independent. The effect of the RJ is smearing with its tail the tailless deterministic jitter distribution that, in the Dual-Dirac model, is usually approximated by two delta functions in $x = \pm\mu$ [34]. Starting from a Gaussian distributed RJ with mean μ and standard deviation σ, the cumulative function of the RJ is given by

$$CDF(x) = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right) \tag{2.1}$$

where *erf* is the error function. From the experimental cumulated histogram of the total jitter, *CH(x)*, we define the *Q* scale as

$$Q = \sqrt{2}erf^{-1}\left(2CH(x) - 1\right) \tag{2.2}$$

where erf^{-1} is the inverse error function. When represented on the *Q* scale, a pure Gaussian distribution appears as a straight line with slope σ^{-1} and intercept μ. Left and right tail intercepts of the general TJ distribution will be called μ_L and μ_R respectively. Hence, σ can be estimated by least squares linear fitting Q for very large values of x. The effect of the DJ components, which is often referred to as Dual- Dirac DJ ($DJ_{\delta\delta}$), is represented by the quantity $\mu_R - \mu_L$. However, the normalized *Q* scale described in [39], better estimates the contribution of DJ pdf, since the *Q* scale generally underestimates the peak-to-peak DJ and overestimates the RJ [34], even if the DJ follows a distribution formed by two Dirac-delta functions. In addition, [40] demonstrated that the jitter distribution tails approach a Gaussian pdf multiplied by a term that is inversely proportional to the TJ and proposed an enhanced method of decomposition.

Jitter Decomposition in Time and Frequency Domains

The other method for jitter decomposition is the FFT analysis, which can be performed by real-time oscilloscopes. The jitter spectrum can be obtained by applying averaging techniques and the narrow-band peaks in the spectrum can be interpreted as PJ or DDJ. Conversely, the noise floor in the average spectrum points out the power of RJ.

Clock Jitter Effects

Clock jitter results in an equivalent additive noise in data converters. For a sinusoidal input signal $x(t) = A\sin(2\pi f_i t)$ and a sampling clock affected by the timing jitter t_j, the equivalent noise induced by jitter is approximatively given by $x_{nj}(t) = A2\pi f_i t_j \cos(2\pi f_i t)$. Assuming, for the sake of simplicity, only RJ with standard deviation σ_j, the noise power related to jitter is:

$$N_j = \left(\frac{A^2}{2}\right)\left(2\pi f_i \sigma_j\right)^2 \tag{2.3}$$

This contribution must be compared with the quantization noise power, that is of the order of $q^2/12$, where q represents the Least Significant Bit (LSB) of the converter. It is apparent that N_j is of some interest mainly for high-frequency or high-resolution ADCs, stimulated by high-level signals. For instance, for a 14-bit ADC operating at 100 MHz, the jitter should be below 100 fs for a worsening of the SNR smaller than 3 dB. A timing jitter σ_j=200 ps at f_i=10 kHz reduces the effective resolution to 16 bits [41]. Similar considerations can be made for D/A converters followed by analog low-pass filters, when only the in-band noise power due to jitter is considered [34].

Experimental Results

A network composed by an A^{2}B main node, and 10 subordinate nodes was employed to analyze the clock propagation along the nodes. A block diagram is shown in Figure 14. The main node is the evaluation board EVAL-AD2428WD1BZ, which is based on the A^{2}B transceiver AD2428. The audio host is the ADAU1452, a DSP of the family SigmaDSP by Analog Devices. The peculiarity of the SigmaDSPs is that they are programmed by a graphical development software called SigmaStudio, thus simplifying the algorithm development process. The same graphical tool is employed for configuring the A^{2}B network too. The audio host exchanges audio with

the A^{2}B main transceiver, and it provides the clock to the network. Since all the subordinate nodes synchronize to the main node clock, it is fundamental that it has low jitter. In the considered network, the internal PLL of the DSP will supply the clock to the main node. All the subordinate nodes employ an AD2428 as A^{2}B transceiver and are equal in hardware. Jitter and latency measurements were carried out with an oscilloscope on the frame clock and the bit clock lines of the TDM interface of the A^{2}B chip. The model employed is the Teledyne LeCroy HDO6000 high-definition oscilloscope (12 bits) featuring 350 MHz bandwidth and 2.5 GSa/s of sampling rate. Since the vertical scale and the time range of the HDO6000 were set, respectively, to 500 mV/div and to 10 ms or less, the jitter measurement floor of the instrument was estimated to be smaller than 5 ps.

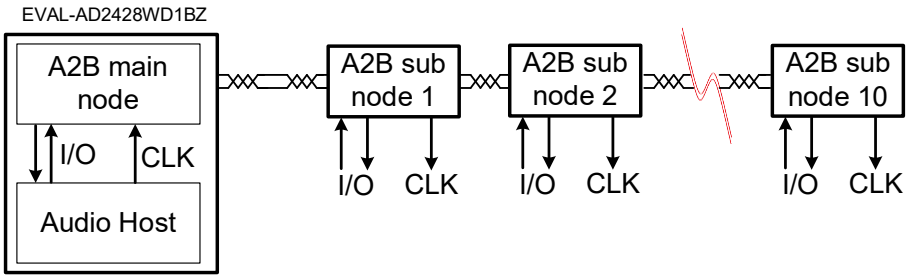


Figure 14. Block diagram of the network employed for clock propagation analysis.

First, frame clock measurements are considered. Among the different ways to express jitter, two are particularly meaningful for clock recovery systems, the *cycle-to-cycle* jitter and the *Time Interval Error* (TIE). The cycle-to-cycle jitter measures how much the clock period changes between any two adjacent cycles, and it shows the instantaneous dynamics that a clock recovery PLL might be subjected to [42]. Instead, TIE expresses "long-term effects" showing the cumulative effect that even a small amount of period jitter can have over time, by measuring how far each active edge of the clock varies from its ideal position [42].

Cycle-to-cycle jitter of the frame clock (48 kHz) was measured for each node. Histogram of Figure 15 shows the results for two different cable lengths between the main and the first subordinate node. The cable length between the other nodes was a few centimeters, thus simulating the use case of microphone or loudspeaker arrays. The cycle-to-cycle jitter of the main node is nearly two orders of magnitude lower than that of the subordinate nodes because it is generated by a local oscillator, while

the others are reconstructed from the superframe rate. Moreover, the cycle-to-cycle jitter does not show dependance from the cable length or the number of nodes.

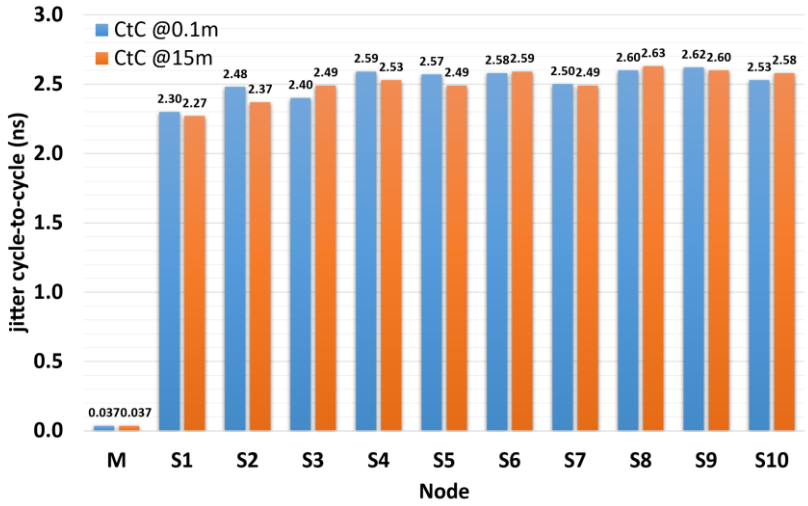


Figure 15. Cycle-to-cycle jitter values of the frame clock for each A^2B node and for two different cable lengths between the main and the first subordinate node.

Then, the HDO6000 has been leveraged to directly measure the TIE of the frame clock. Figure 16 shows an acquisition of 1 MSamples of TIE values for ensuring a statistical uncertainty well below the 1% [34] The upper trace is the acquisition of the frame clock signal (48 kHz), while the lower trace is the histogram of 1 MSamples of TIE. This measure was repeated for the first, the fifth and the tenth node and data was post-processed to decompose DJ and RJ. By fitting the TIE histograms with a Gaussian function (Figure 17), obtaining an adjusted R-squared always better than 0.9999, it was concluded that RJ is preponderant. Then, the CDFs were plotted in Q-scale (Figure 18). However, left and right tails showed the same slope, hence the normalized Q-scale transform was not used.

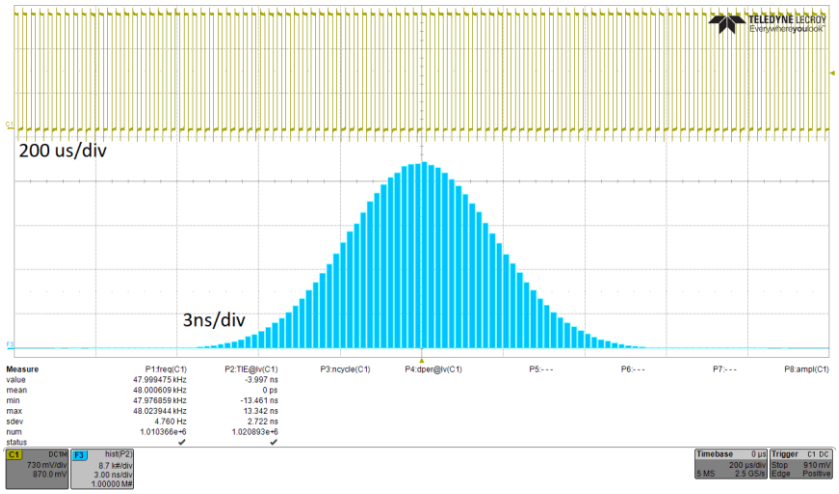


Figure 16. Oscilloscope acquisition of the frame clock (upper trace) and TIE histogram (lower trace). The histogram has 1 MSamples of TIE values.

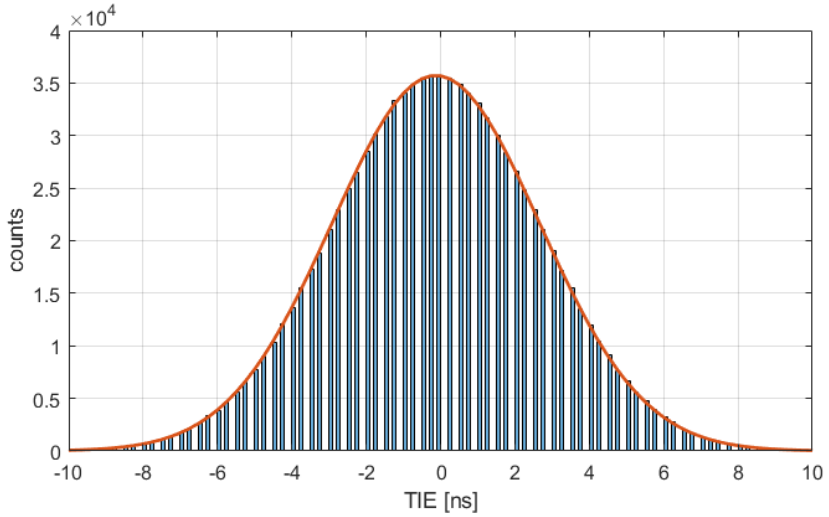


Figure 17. TIE histogram of the tenth subordinate node fitted with a Gaussian function.

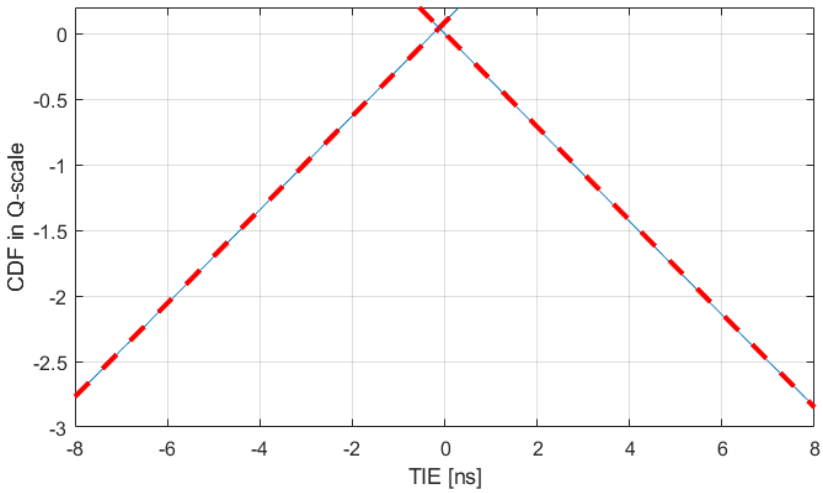


Figure 18. CDF in Q-scale of the tenth subordinate node, with the fitting straight lines, from left and from sides.

Obtained values of DJ and RJ for the first, fifth and tenth subordinate nodes are shown in Table VI. Since the tail fitting method assumes that the right (left) peak of the dual-Dirac DJ distribution does not contribute to the left tail (right, respectively), an accurate decomposition of the two contributions is possible only when the two Dirac pulses are sufficiently separated. The extracted DJ contribution, however, was of one order of magnitude smaller than RJ component and therefore the accuracy of the method is questionable. Then, 32 TIE spectra were averaged and it was observed a narrow-band peak ranging from 170 ps to 140 ps at 12 kHz, which was also previously observed in [43]. In addition, the TIE spectra confirmed the reported TJ in the band above 100 Hz.

Table VI - DJ and RJ from CDF in Q-scale

Node	DJ	RJ
1	100 ps	1.6 ns
5	170 ps	2.2 ns
10	250 ps	2.8 ns

Finally, TIE measurements on the frame clock were carried out for all the A^2B nodes and for two different cable lengths between the main and the first subordinate

node. From results shown in Figure 19 it was observed, as for cycle-to-cycle measurements, that the TIE of the main node is about two orders of magnitude lower than that of the subordinate nodes, and that the cable length do not worsen the clock recovery mechanism. On the contrary of cycle-to-cycle, TIE increases as the number of node increases. This effect is caused by the clock recovery that each subordinate node performs, and consequently the jitter accumulates along the bus.

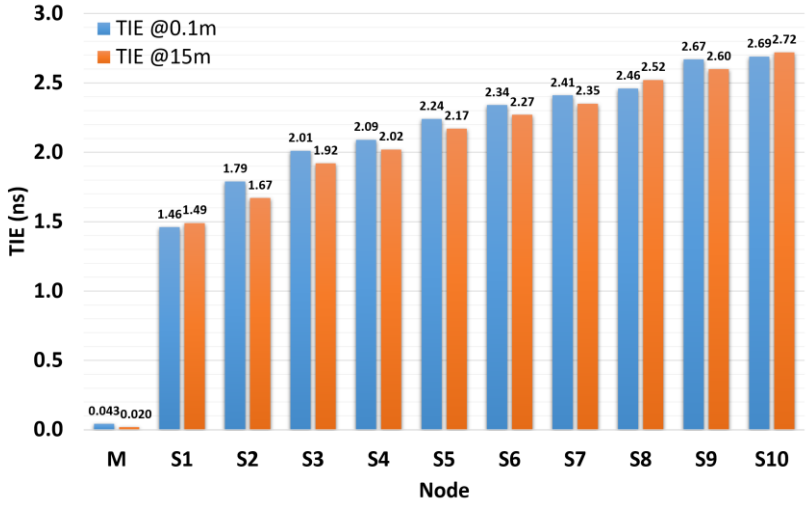


Figure 19. TIE values of the frame clock (48 kHz) for each node at two different cable lengths between the main and the first subordinate node.

Afterwards, measurements were carried out on bit clock too, since some transducers (e.g., PDM microphones) use this clock for the oversampling. As for frame clock, also for bit clock 5 MSamples TIE histograms were acquired for the first, the fifth, and the ninth subordinate nodes. The results were fitted with a Gaussian function, and it was observed a preponderance of the RJ, since the adjusted R-squared was always better than 0.9999. DJ and RJ were again decomposed by using the CDF in Q-scale and the TIE spectrum. The CDF showed results similar to those observed for the frame clock. The TIE spectrum (Figure 20) was evaluated by averaging 34 spectra obtained by acquiring TIE for 10 ms, i.e., approximately 30720 x 34 samples. As it can be seen, there are several narrow-band peaks at the frequencies equal to $(k \times 48) \pm 12$ kHz. The bit clock spectrum corresponds to the frame clock spectra in the frequency range from 100 Hz to 24 kHz. The peak levels, due to DJ, slightly decrease by moving from the first to the last subordinate node, probably for a

beneficial dithering effect of the increasing RJ. The results of DJ and RJ obtained from the CDF in Q-scale and TIE spectrum are summarized in Table VII.

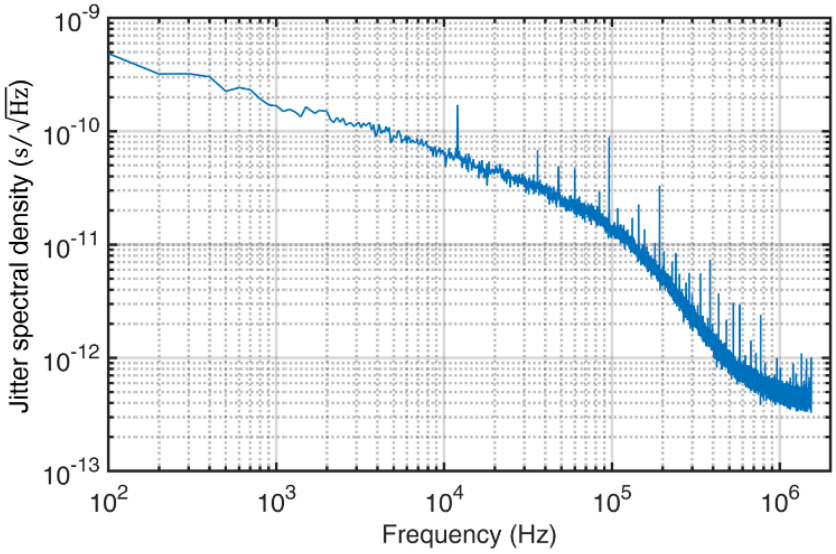


Figure 20. Bit clock jitter spectral density of the first subordinate node.

Table VII - DJ and RJ of bit clock

	CDF in Q-scale		Averaged spectra	
Node	**DJ**	**RJ**	**DJ**	**RJ**
1	103 ps	1.6 ns	225 ps	1.6 ns
5	246 ps	2.3 ns	223 ps	2.3 ns
9	254 ps	2.9 ns	195 ps	2.9 ns

Another useful feature of the A^{2}B transceivers is a two levels depth spread spectrum on the bit clock line, which can be employed to reduce the radiated power per unit bandwidth [44]. The effects on the jitter of this operational modes (low and high spread spectrum) were investigated for the first, fifth and ninth subordinate nodes. Figure 21 shows the measured TIE histograms with the low depth spread spectrum mode enabled. Firstly, the histograms are clearly the results of the convolution of RJ and DJ, and secondly as the number of nodes increases, the components of the spread spectrum modulation are less noticeable. Again, the normalized CDF in Q-scale was employed to decompose jitter components (Figure 22), and the TIE spectral density

was obtained for the first subordinate node with high depth spread spectrum mode enabled (Figure 23). Note that there is an increment of narrow-band components above 144 kHz, and particularly at 336 kHz (7.3 ns). Table VIII summarizes the results of the decomposition of DJ and RJ for the low and high depth spread spectrum modes.

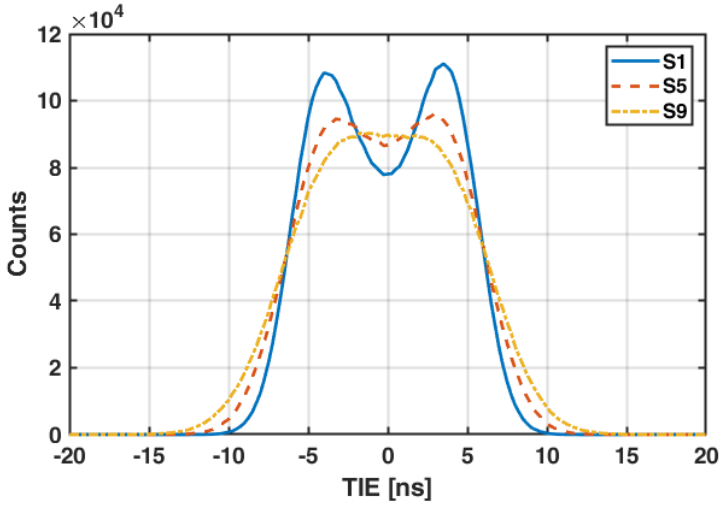


Figure 21. TIE histograms of the first, fifth and ninth subordinate nodes, with low depth spread spectrum enabled.

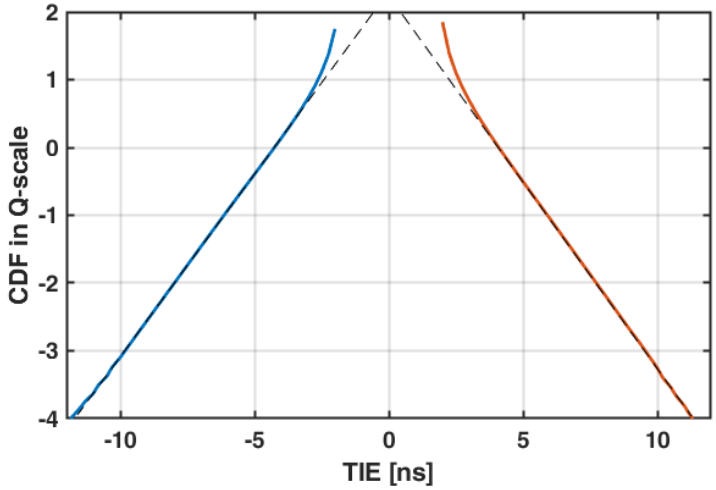


Figure 22. Normalized CDF in Q-scale of the first subordinate node, with fitting straight lines, from left and from sides.

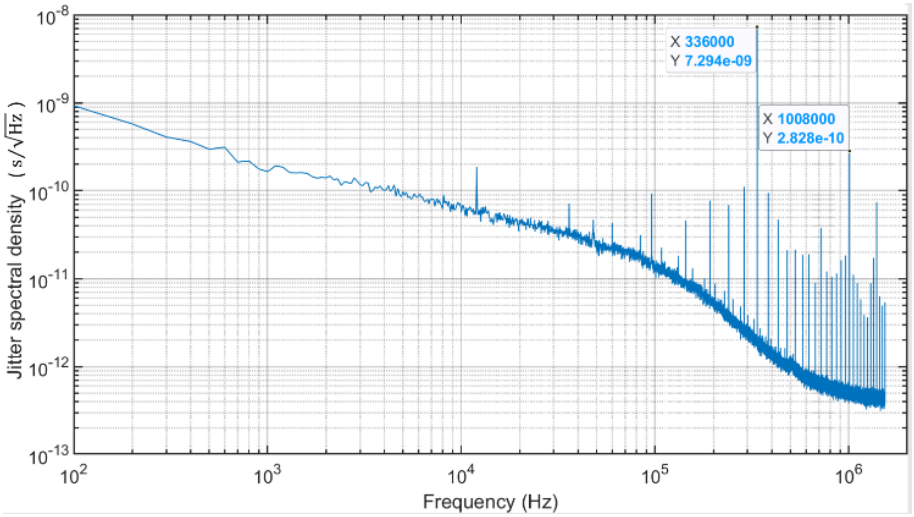


Figure 23. Bit clock spectral density for the first subordinate node, with high depth spread spectrum mode enabled.

Table VIII - DJ and RJ from CDF in Q-scale

	Low depth spread spectrum mode		High depth spread spectrum mode	
Node	**DJ**	**RJ**	**DJ**	**RJ**
1	8.16 ns	1.86 ns	17.6 ns	2.02 ns
5	8.35 ns	2.44 ns	17.6 ns	2.56 ns
9	7.87 ns	3.11 ns	17.4 ns	3.17 ns

Finally, in Figure 24 are shown two oscilloscope acquisitions of TIE histograms of bit clock (fifth subordinate node) for the two spread spectrum depths. As it can be seen, the behavior is similar for the low and high depth modes, but in the high depth mode the DJ is greater with respect to the low mode. In addition, the TIE standard deviation, as expected, higher with the high depth spread spectrum.

When considering loudspeaker and microphone arrays or more generally transducer arrays, it is crucial to know the sampling clock delay between each device. For this reason, the delay introduced by the A^2B transceivers was investigated. Figure 25 shows an oscilloscope acquisition of the frame clock edges (48 kHz) of the main node (upper trace), first, fifth and tenth subordinate nodes (from top to bottom). Jitter

contribution is visible thanks to the oscilloscope persistence, but it is negligible compared to the latency.

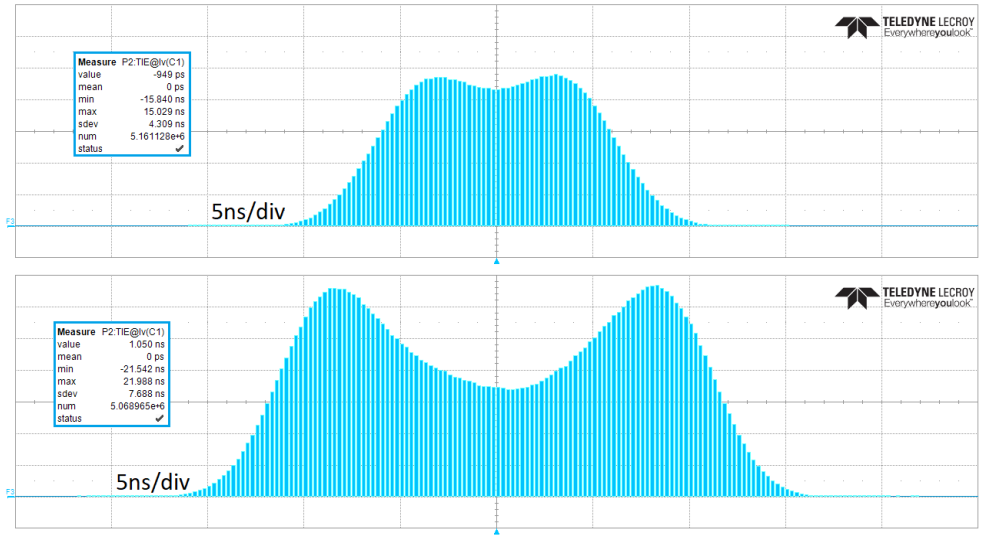


Figure 24. Oscilloscope acquisition of TIE histograms of the bit clock of the fifth subordinate node. Upper trace is with low depth spread spectrum, while lower trace is with high depth spread spectrum.

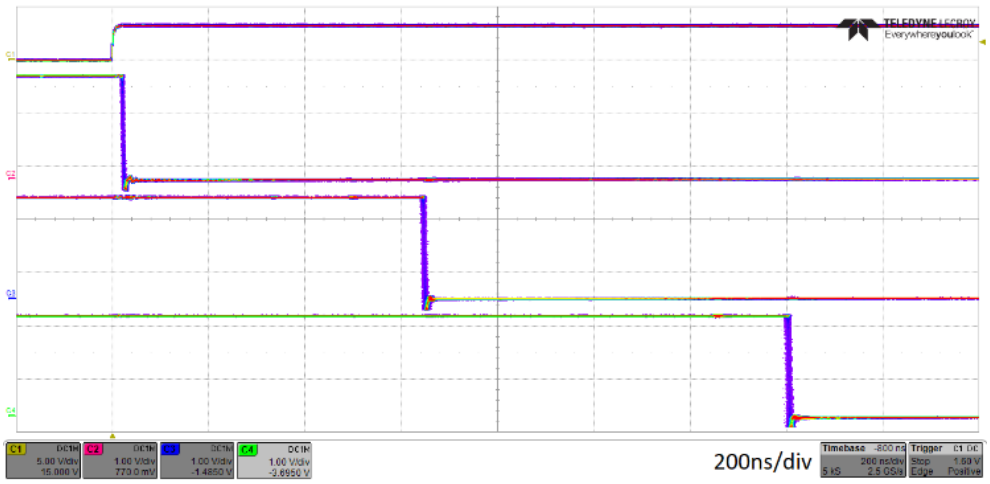


Figure 25. Clock edges of the frame clock. Main node, first, fifth, and tenth subordinate nodes from top to bottom.

The delay introduced by each node has been measured for two different cable lengths between the nodes and the time reference has been taken on first subordinate node (Figure 26). Note that the measured delay values fit a straight line, which means that the delay introduced by each node is constant. When a cable of 4 m is added between each node, the delay values increase along the network, thus the propagation delay is not negligible.

Statistical measures on the delay introduced by the nodes were carried out to prove that the delay is constant and deterministic. For this purpose, the skew between two clock edges of two consecutive nodes was measured and a histogram with 150 kSamples was obtained. The oscilloscope acquisition of the clock skew between the eighth and the ninth nodes is shown in Figure 27. The upper grid contains the positive edges of the two frame clock signals (the skew is measured at the 50% of the signal level), while the lower grid shows the skew histogram. Results obtained for each node are illustrated in Figure 28. The values of the histogram represent the mean value of the distribution of the skew measured with the oscilloscope, while its standard deviation is indicated over each bar. As it can be seen, the latency between each node is constant, with a mean value of 154.14 ns, which is compliant with the declared value [45].

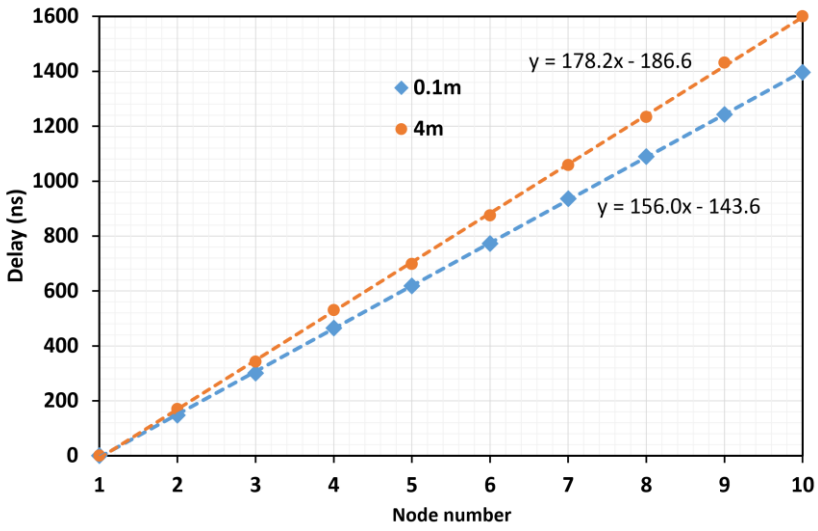


Figure 26. Delay propagation in an A^2B network, measured with respect to the first node and for two different cable lengths between the nodes.

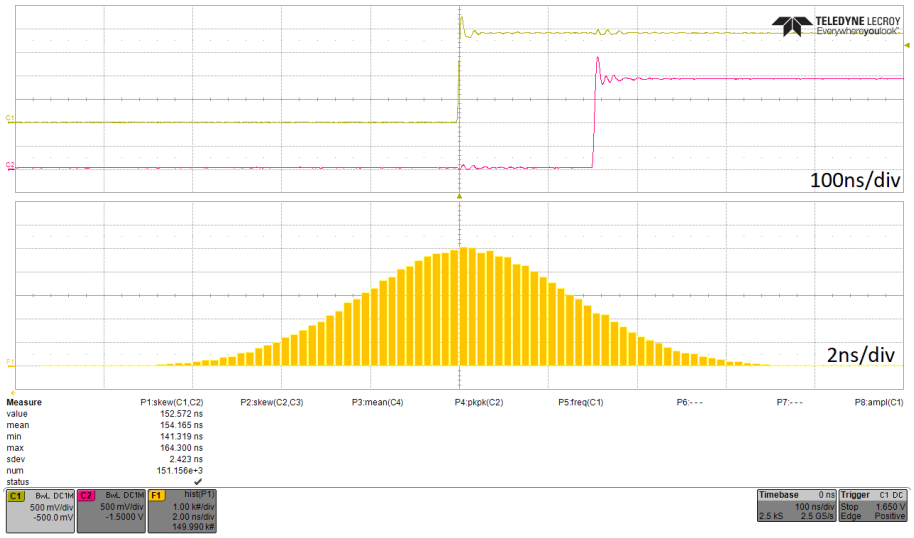


Figure 27. Oscilloscope acquisition of the clock skew between two nodes (eighth and ninth). Upper grid shows the two clock signals (yellow trace is eighth node, red trace is ninth node), while lower grid is the clock skew histogram.

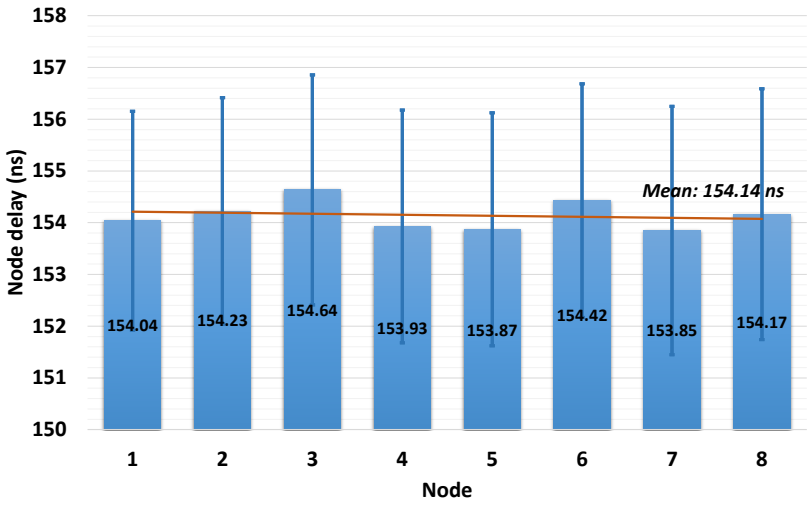


Figure 28. Delay introduced by each node with respect to the previous one. The value of each bar is the mean value of the histogram. The standard deviation is shown over each bar.

A very interesting feature of the A^{2}B transceivers is the compensation of the propagation delay. In fact, each subordinate node can introduce a programmable delay on the start of the audio frame with a resolution of about 20.35 ns, namely 1/49.152 MHz at a sampling frequency of 48 kHz. A new clock skew measure was carried out between the first and the ninth subordinate nodes (after compensation) and the oscilloscope acquisition can be seen in Figure 29.

Finally, the propagation delay introduced by the cables was investigated by measuring the clock skew between the first and the second subordinate nodes for cable lengths between a few centimeters and 14 m. Results are shown in Figure 30 and it can be observed that experimental points fit a straight line with equation $y = mx + q$, where q is the delay of a single node and m is the inverse of the propagation speed of the cable ($2.7 \cdot 10^8$ m/s).

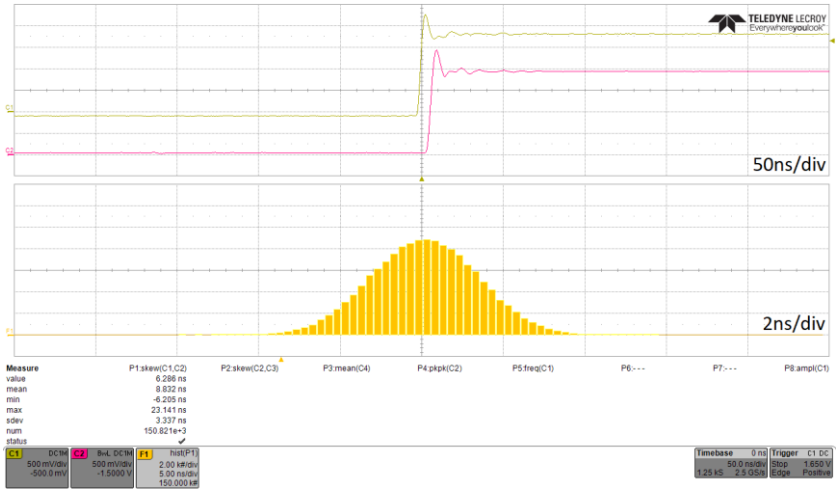


Figure 29. Oscilloscope acquisition of the clock skew (with delay compensation) between the first and the ninth subordinate nodes. Skew histogram in the bottom grid.

In conclusion it was possible to create a clock propagation model of an A^{2}B bus composed by a main node and ten subordinate nodes. It was observed that for each node jitter is predominantly random (RJ) and that its deterministic component (DJ) can be neglected. In addition, TIE jitter has an unbounded Gaussian distribution, allowing to quantify it with the standard deviation of the distribution. In addition, TIE jitter slightly increases as the number of node increases, as shown in Figure 19. Regarding the latency, it has been demonstrated that each subordinate node

introduces a deterministic latency of about 154 ns, and that it can be compensated with a resolution of about 20 ns. If low jitter requirement is not mandatory for a particular application, it is possible to enable the spread spectrum feature to reduce the electromagnetic emissions. When this feature is enabled, the RJ component increases, and the DJ component is no longer negligible. As it will be explained in Section 3.3, the developed clock propagation model is useful when designing transducer arrays, such as microphone arrays. For example, by introducing the model in the post-processing simulations, it is possible to verify if jitter degrades the beamforming performance of the array, by reducing its directivity.

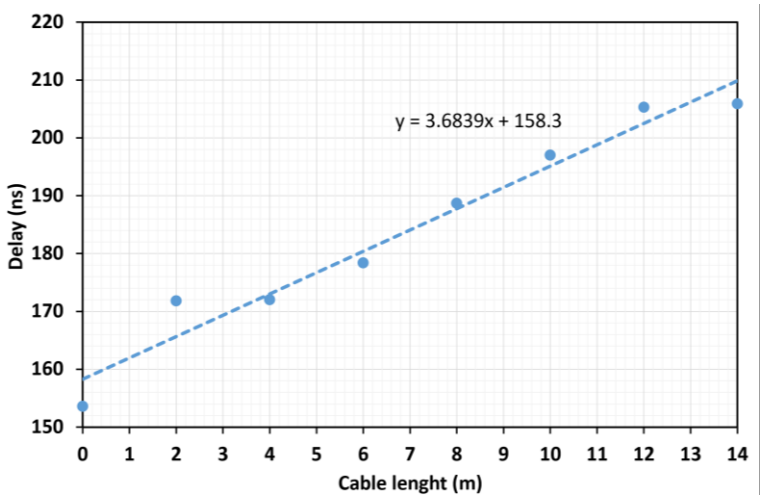


Figure 30. Delay between the first and the second subordinate nodes for different cable lengths.

2.3 A²B Plug-and-Play

As explained in Section 1.1, a microcontroller is essential only for the first initialization of the A²B network, hence the details of the nodes in the bus must be known in advance. Being not the typical scenario for professional or consumer audio applications, where for example devices can be added or removed at need, a new A²B feature has been implemented allowing to dynamically change nodes and their settings without prior knowledge. Since the new nodes are immediately configured and audio starts being exchanged as soon as a node is attached to the bus, this feature

was called A²B Plug-and-Play. Its operation is different from similar application layers of the other protocols presented in this dissertation. A²B is in fact the only technology where a single device must initialize and configure the others.

The A²B Plug-and-Play is based on two mechanisms of A²B, namely *partial discovery* and *node configuration from EEPROM.* Partial discovery consists in reconfiguring nodes, that for some reason dropped from the bus, without reinitialize all the nodes of the bus, thus allowing to keep the other nodes working without audio interruptions, while trying to recover the lost devices. Node configuration from EEPROM indicates the possibility to initialize a subordinate node by reading data from its EEPROM, instead of using the data of the microcontroller that performs the initialization. The A²B Plug-and-Play functionality take advantage of these two mechanisms. In fact, partial discovery is useful to configure nodes that are connected to the bus at a later time, without interrupting audio exchange and since the target processor cannot know in advance the nodes present in the bus, their configuration will be read from their EEPROM. Therefore, when a new node is added to the bus, it is configured while the other nodes are working, and its configuration will be read from its EEPROM. Furthermore, this mechanism allows configuring I²C peripherals connected to the node (e.g., codecs, sensors, integrated amplifiers) since their configurations can be stored in the EEPROM too.

Two additional operations were implemented: the management of nodes disconnection and the periodic discovery of new nodes. The former is required because when nodes are disconnected from the bus, the others A²B transceivers send an error message to the main node, and even if they continue working, channels slots must be reassigned. Hence, the target processor that implements the A²B Plug-and-Play updates the channel slots and timing information in each node. Instead, the periodic discovery of new nodes, which is not provided by default by A²B, is required because typically the bus configuration is established in advance and there is no need to add new nodes. The A²B partial discovery mechanism is leveraged for this purpose. The target processor periodically launches a partial discovery starting from the last known node. If the discovery fails, it means that no new node was connected to the bus, otherwise the target processor configures the new node. The polling time (convenient values may be a second or a few seconds) depends on the application and the desired system responsiveness.

The channels routing depends on the application too, but since the proposed architecture was designed to exchange data between a central unit and the peripheral

devices (typically transducers), it was opted to configure the new nodes with a fixed routing. To be more specific, an A^{2}B device declares its number of input and output channels (a value saved in the EEPROM) and these are consequently routed to/from the interface board. Other scenarios may require a dynamic routing, and in this case, an additional application-level software should be implemented.

A custom discovery process was implemented starting from the *Modified Discovery Flow* described in the appendix A of [46]. The main operations involved in the custom discovery process are illustrated in Figure 31. This process is divided into four main parts:

1. Main node initialization
2. Prepare for discovery
3. Discovery and configuration
4. End of discovery

The *main node initialization* is done at the turn on of the system, after the PLL of the main node has locked, and it consists of the initialization of the registers necessary for the system to work (e.g., TDM settings, GPIO, and interrupts). The A^{2}B transceivers do not have an internal memory, thus this operation must be carried out by a microcontroller every time the system is turned on, or the transceiver is reset.

The A^{2}B chips drive an external switch to enable bus power to the next node or to perform diagnostic operations. During the *prepare for discovery* stage, switches of all the nodes are set as explained in the A^{2}B programming manual. In particular, the transceiver's register involved is the `A2B_SWCTL` (address 0x09) which is set to 0x21 for each node, except for the last node that is set to 0x01.This operation is mandatory when trying to discover a new node and it allows avoiding false failure messages.

Then, the discovery of a new node starts following the scheme of the standard *modified Discovery Flow*: each time a new node is discovered, it is immediately configured. The target processor does not have prior knowledge of the subordinate node configuration, so it reads it from the EEPROM by using the remote peripheral I^{2}C access.

Lastly, during the *end of discovery*, the `A2B_SWCTL` register of each node, including the main node, is set to 0x01 (normal operation). The process ends by initializing few registers of the main node (`A2B_PLLCTL`, `A2B_SLOTFMT`, `A2B_DATCTL` and `A2B_I2SRRATE`) and by setting the bit `NEWSTRCT` in the `A2B_CONTROL` register of the main node to make things take effect.

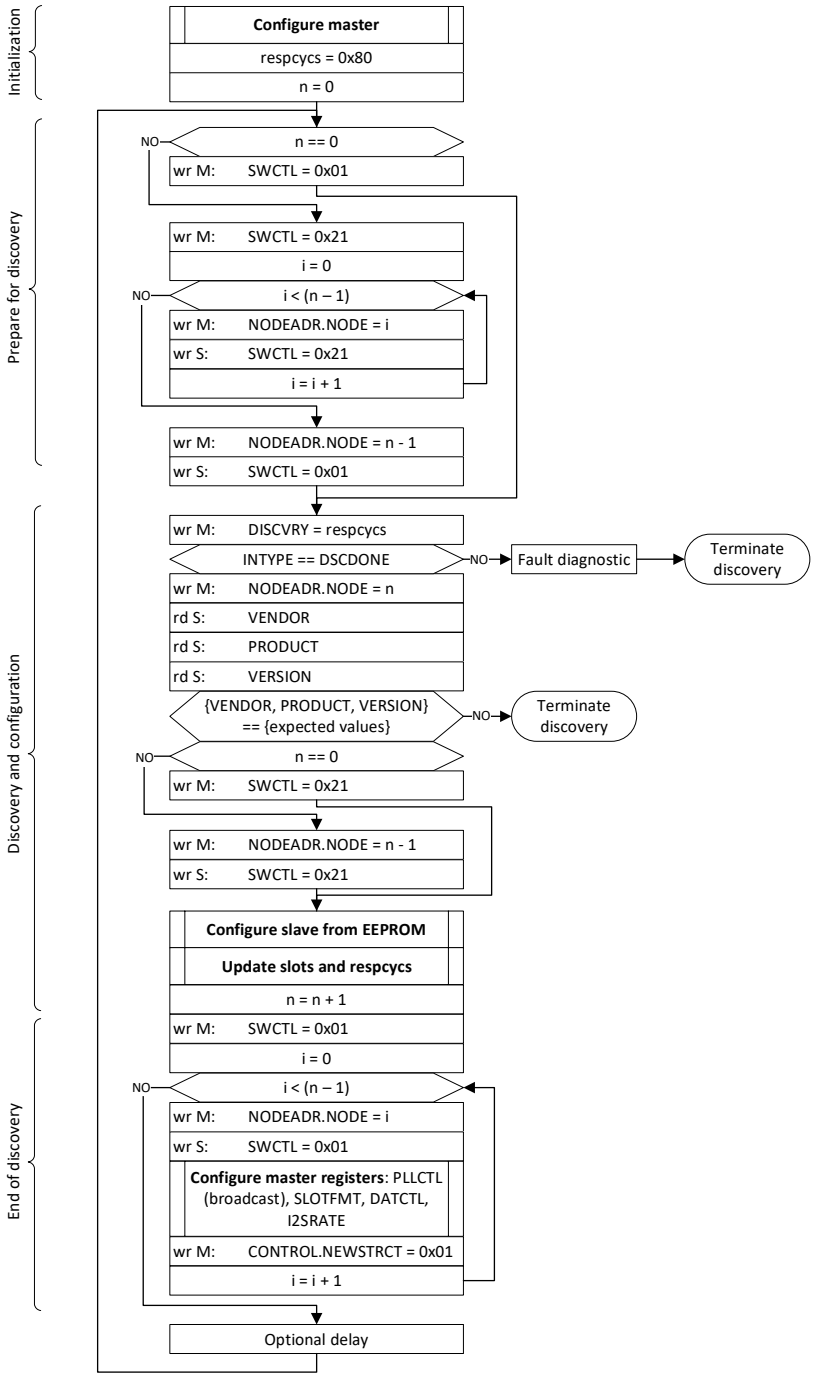


Figure 31. Flow chart of the discovery mechanism of A²B Plug-and-Play.

The configuration of a subordinate node from its EEPROM is described in appendix C of [46]. This document states that the memory (EEPROM or similar) must be connected to the subordinate transceiver through I^2C at it must use the address 0x50. In addition, it is described the data format of the node configuration stored in EEPROM. However, the implementation of the A^2B Plug-and-Play feature, requires additional information that will be described in the next paragraphs.

The configuration memory is organized in *configuration blocks*, namely blocks of data composed by a header and a payload in which the register values of the devices are contained. The first block (Table IX) contains general information about the device, the number of configuration blocks, and it is employed to identify if the EEPROM contains configuration data. The module vendor, product and version IDs indicate the transceiver that will be configured (e.g., 0xAD, 0x24, 0x28 respectively), while the number of configuration blocks allows to know how many blocks must be read to configure the A^2B transceiver and its possible peripherals. The last byte is a CRC, useful for detecting errors.

Table IX - Initial block of the configuration memory

Address	Content
0x00	0xAB (Indicates Configuration Memory)
0x01	Module Vendor ID
0x02	Module Product ID
0x03	Module Version ID
0x04	Reserved - value should be ignored
0x05	Number of Configuration Blocks
0x06	Reserved - value should be 0x00
0x07	CRC-8

Next, configuration blocks are found. They can be composed by a header, a payload, and an optional CRC-8. Three types of headers are available (Table X), all of them are 3 bytes long. The *type A* and *type B* headers precede a payload containing the configuration data for the A^2B transceiver or a peripheral device. The first four bits identify the header type, the second four bits together with the second byte contain the number of bytes of the payload. The last byte defines the target device: if the value is 0x00, the payload data is for the A^2B transceiver, otherwise it contains the I^2C address of the peripheral device to configure. The difference between the headers type A and type B is that the former does not use a CRC to protect the payload data,

while the latter indicates that the last byte of the payload is a CRC-8 calculated on the header and the payload. Instead, the header *type C* specifies a delay value, in milliseconds, that the target processors must wait before continue configuring the device. The first 4 bits indicate the header type, while the second four bits together with the second byte contain the delay value. The last byte is a CRC-8 calculated on the first two bytes of the header. Following this header there is no payload.

The payload of each configuration block contains the data to be written in consecutive registers. For this reason, only the first register address is indicated (in the first byte of the payload), and the other bytes are the register values that should be written starting from it.

Table X - Header types

Header Type A (no CRC)		
Byte 0	Header type: 0x0	# Payload Bytes (MSBs)
Byte 1	# Payload Bytes (LSBs)	
Byte 2	0x00 for AD24xx or Peripheral I^2C Address	

Header Type B (CRC used)		
Byte 0	Header type: 0x1	# Payload Bytes (MSBs)
Byte 1	# Payload Bytes (LSBs)	
Byte 2	0x00 for AD24xx or Peripheral I^2C Address	

Header Type C (delay, no payload)		
Byte 0	Header type: 0x2	Delay (MSBs)
Byte 1	Delay (LSBs)	
Byte 2	CRC-8	

The A^2B Plug-and-Play functionality requires other fields that are not specified in the default mechanism of a subordinate node configuration. Additional fields are specified in a custom configuration block that follows the initial block. The header format is the same as type A, except for the header type which is 0xF and the payload number of bytes is fixed to 16. The structure of the custom configuration block can be seen in Table XI. The information included in the payload are a byte fixed to

0xFE for the custom payload identification, a byte for Plug-and-Play versioning, product and version ID of the device to configure, and number of input and output channels. The other bytes are reserved for future use. Since the standard configuration memory does not specify the audio slots that a node adds or consumes (namely the input and output channels), the most important fields that the custom configuration block adds are the input and output channels of the device. In fact, they can be used by the target processor to verify if there is enough available bandwidth and, in that case, to reserve the slots. In case a node can be added to the bus (the current number of nodes is less than ten), but there is not enough bandwidth available (32 channels are already reserved), the new node is added to the bus, and no audio is exchanged with it.

Another modification to the default configuration memory concerns the addition of other two configuration blocks for the transceiver initialization. In fact, some fundamental registers were not considered, thus preventing the device to be configured only from its memory content. An example of the configuration memory for the AD242x transceiver (including the custom configuration block and the additional transceiver registers) is shown in Table XII.

Table XI - Custom configuration block

Address	**Content**	
0x08	Header type: 0xF	# Payload Bytes (0x0)
0x09	# Payload Bytes (0x10)	
0x0A	0x00	
0x0B	0xFE	
0x0C	Plug-And-Play Version	
0x0D	Custom Product ID	
0x0E	Custom Version ID	
0x0F - 0x18	Reserved (0x00)	
0x19	Number of Input Channels	
0x1A	Number of Output Channels	

Table XII - Configuration memory for the AD242x transceiver

Address	Content	
0x00	0xAB	*Initial block*
0x01	Module Vendor ID	
0x02	Module Product ID	
0x03	Module Version ID	
0x04	Reserved	
0x05	0x05	
0x06	Reserved (0x00)	
0x07	CRC-8	
0x08	0xF0 (Type Custom, PB MSBs = 0)	*Header of the custom conf. block*
0x09	0x10 (PB LSBs = 16)	
0x0A	0x00	
0x0B	0xFE	*Payload of the custom configuration block*
0x0C	Plug-And-Play Version	
0x0D	Custom Product ID	
0x0E	Custom Version ID	
0x0F - 0x18	Reserved (0x00)	
0x19	Number of Input Channels	
0x1A	Number of Output Channels	
0x1B	0x00 (Type A, PB1 MSBs = 0)	*Header of 1st configuration block*
0x1C	0x04 (PB1 LSBs = 4)	
0x1D	0x00 (For AD24xx programming)	
0x1E	0x0A (Address pointer for BCDNSLOTS)	*Payload of 1st configuration block*
0x1F	Data for BCDNSLOTS	
0x20	Data for LDNSLOTS	
0x21	Data for LUPSLOTS	
0x22	0x00 (Type A, PB2 MSBs = 0)	*Header of 2nd configuration block*
0x23	0x5 (PB2 LSBs = 5)	
0x24	0x00 (For AD24xx programming)	
0x25	0x1B (Address pointer for INTMSK0)	*Payload of 2nd configuration block*
0x26	Data for INTMSK0	
0x27	Data for INTMSK1	
0x28	Data for INTMSK2	

0x29	Data for BECCTL	
0x2A	0x00 (Type A, PB3 MSBs = 0)	*Header of 3rd configuration block*
0x2B	0x15 (PB3 LSBs = 21)	
0x2C	0x00 (For AD24xx programming)	
0x2D	0x3F (Address pointer for I2CCFG)	*Payload of 3rd configuration block*
0x2E	Data for I2CCFG	
0x2F	0x00 (Data for PLLCTL)	
0x30	Data for I2SGCFG	
0x31	Data for I2SCFG	
…	…	
0x41	Data for PINCFG	
0x42	0x00 (Type A, PB4 MSBs = 0)	*Header of 4th configuration block*
0x43	0x06 (PB4 LSBs = 6)	
0x44	0x00 (For AD24xx programming)	
0x45	0x59 (Address pointer for CLK1CFG)	*Payload of 4th configuration block*
0x46	Data for CLK1CFG	
0x47	Data for CLK2CFG	
0x48	Data for BMMCFG	
0x49	Data for SUSCFG	
0x4A	Data for PDMCTL2	

Finally, more details about the discovery mechanism and the bandwidth calculation are given. As it was shown in Figure 31, the discovery of a new node starts when a value of *response cycles* (RESPCYCS from now on) is written to the register `A2B_DISCVRY` of the main node. RESPCYCS specifies "*the relative time from the start of a synchronization control frame (SCF) to the moment the last subordinate node responds with a synchronization response frame (SRF)*" as stated in [46]. RESPCYCS is also employed to detect if the next node in the bus dropped. In fact, if a node does not receive the superframe within the RESPCYCS time, it will assume that the next nodes dropped, and it will become the last node in the bus, thus starting to send the SRF. The RESPCYCS value depends on the characteristics of the network. To be more specific:

- Number of subordinate nodes
- Number of downstream slots

- Downstream slot size
- Number of upstream slots
- Upstream slot size
- Main node I^2S/TDM channel configuration

Appendix B of [46] explains how to calculate the correct value of RESPCYCS for each node. The formula to calculate the RESPCYCS requires to know the bus configuration, but in the considered scenario nodes and channel reserved change dynamically, thus a correction to the RESPCYCS calculation was made. The target processor begins the discovery of the first node with a value of RESPCYCS of 0x80 (half of the full-scale value) and when a node is discovered and its channels are known, it will calculate the correct value of RESPCYCS according to the formula of appendix B of [46], to verify if there is enough bandwidth for the new node. The requirements to meet are:

- The maximum value of the downstream RESPCYCS is less than the minimum value of the upstream RESPCYCS.
- The number of currently used channels plus the channels to be added by the new node does not exceed 32.
- The number of subordinate nodes discovered is less than 10.

If these requirements are met, the new node is added to the bus and its channels are routed consequently. Note that the RESPCYCS value so calculated is referred to the main and the first subordinate node. RESPCYCS of other nodes will be obtained by subtracting 4 for each node from the RESPCYCS value of the master. As soon as the new node is added to the bus and its channels are routed, the RESPCYCS values are updated in each node.

It must be pointed out that in the transceiver family AD242x it is not possible to choose the channel position of a node in the audio slots of the bus. This concept will be better explained with an example. Suppose that an A^2B network is composed by a main node and two subordinate nodes, and that the main node is transmitting 2 channels to each subordinate node (downstream), while they are transmitting 2 channels each to the main node (upstream). This scenario is depicted in Figure 32. The main node receives from the audio host (on the I^2S/TDM interface) the four channels to be transmitted on the bus. To save bandwidth on the bus the first two channels are for the second subordinate node. In fact, once the samples are packed in the superframe, the first subordinate node will collect the last two audio slots, leaving

bandwidth for other possible downstream transmission. On the contrary, if the main node transmits the data for the first subordinate node in the first audio slots, in the superframe that arrives at the second subordinate node the first two audio slots will still be reserved. The upstream phase occurs in the same way. The last subordinate node will send its two channels in the first audio slots of the superframe, and similarly the previous nodes will continue to add their data in the successive slots. The consequence of this mechanism is that the channel number is reversed with respect to the position of subordinate nodes in the bus. For the downstream, the first nodes will receive the last channels from the main node and the last node will receive the first channels. The same applies to the upstream where the last node will send the first channels and the first node will send the last channels.

The undesired effect when combined with A^{2}B Plug-and-Play is that when the number of nodes changes, for example when a device is added or disconnected, the channels must be shifted consequently.

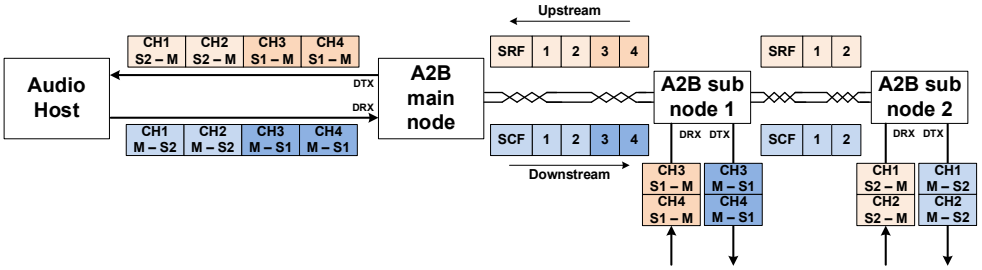


Figure 32. Block diagram of the channel mapping between I^{2}S/TDM and A^{2}B superframe slots.

3 Case Studies

The proposed architecture can be employed in several scenarios. In the next sections, three case studies that take advantage of the proposed A^2B architecture will be described:

1. Audio distribution in a car
2. Wave Field Synthesis listening room
3. Microphone arrays

3.1 Audio Distribution in a Car

Automotive audio has seen a great technology advancement in the last decades, especially thanks to the development of electric and autonomous vehicles. Recent research topics on automotive audio concern active sound control [47], [48], namely the manipulation of the sound field through the use of electroacoustic actuators such as loudspeakers or shakers. Active sound control encompasses applications like ANC, spatial sound reproduction, audio zoning, engine sound quality, and sound generation for electric vehicles. All these technologies require the usage of a high number of sensors (i.e., microphones and accelerometers) and actuators (i.e., loudspeakers and shakers) distributed in the car. This leads to complex and bulky wirings when connecting all the devices with a conventional system architecture comprising a central control unit [49]. In addition, analog wirings would be too expensive and heavy. Figure 33 schematically shows an example of the traditional cabling of the audio system in a car. Each device is connected to the head unit or to the amplifier with a dedicated analog connection, increasing the total cable length and so costs and weight.

Digital busses are currently employed in automotive, partially overcoming these problems: Control Area Network (CAN) is traditionally employed for communication between Electronic Control Unit (ECU) and sensors, while Media Oriented Systems Transport (MOST) is dedicated to audio and video [50]. Currently, MOST is widely employed, but it is an expensive solution. As an alternative, ethernet seems to be a promising technology thanks to its high bandwidth, even though it natively lacks synchronization which is fundamental for audio data transport. AVB [51], [52] is a protocol that aims to overcome this limitation by implementing synchronization between devices and traffic shaping, as described in detail in Section 1.2.2. As for MOST, AVB is an expensive solution for the automotive industry for several reasons. Isolation transformers are required on each line, traditional Cat-5 cables are expensive and heavier with respect to UTP cables, and finally AVB requires FPGAs or ARM-based processors to manage the protocol stack.

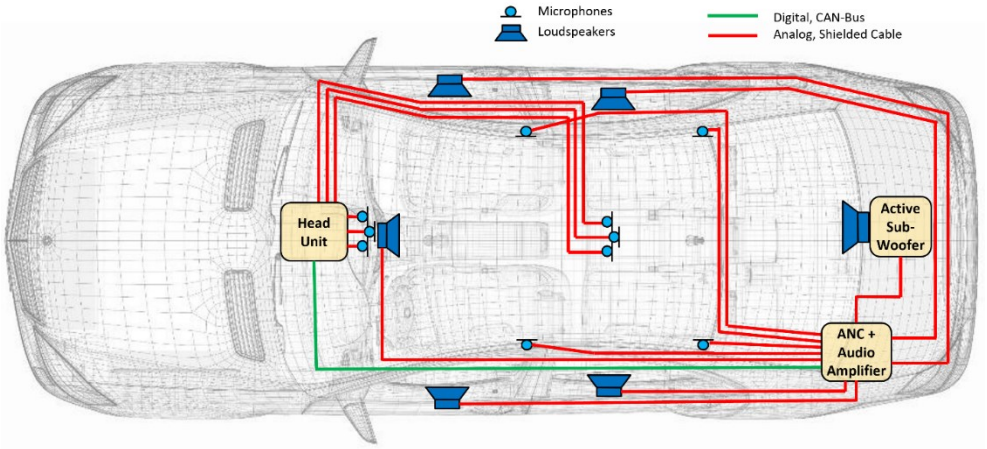


Figure 33. Example of traditional car cabling for audio distribution.

A^{2}B aims to solve most of the previously mentioned problems. Communication is managed by dedicated low-cost transceivers that require only a microcontroller for the initial configuration. Data transport occurs on a single UTP cable, which is light and cheap, and the daisy-chain topology aims to reduce cables length. Finally, A^{2}B introduces a deterministic latency of just two samples (i.e., about 42 µs at 48 kHz). Figure 34 shows an example of A^{2}B cabling. In this case a single A^{2}B network is employed to acquire microphone signals and to implement communication between the amplifier and the head unit, while the connections with the loudspeakers

remain analog. Another example of an A^{2}B system architecture can be seen in Figure 35, where three A^{2}B busses (in the figure they are represented with different shades of blue) are employed to implement connections with both microphones and loudspeakers. Connections with the loudspeakers are made possible by the adoption of the new A^{2}B transceivers family (i.e., AD243x) allowing to transport power together with data, and by equipping loudspeakers with class-D amplifiers, thus creating a distributed audio amplifier. The subwoofer is connected employing a dedicated A^{2}B line to guarantee enough power availability. Microphones instead can be cabled with standard power A^{2}B transceivers.

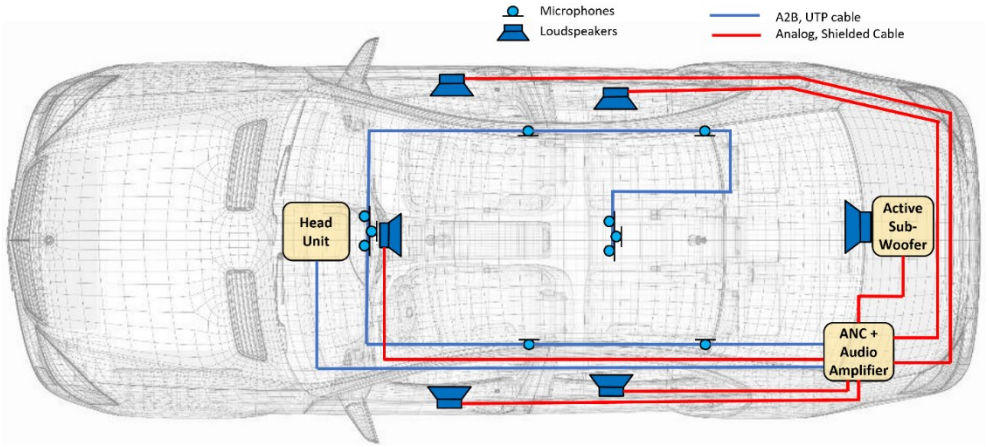


Figure 34. Example of A^{2}B and analog car cabling.

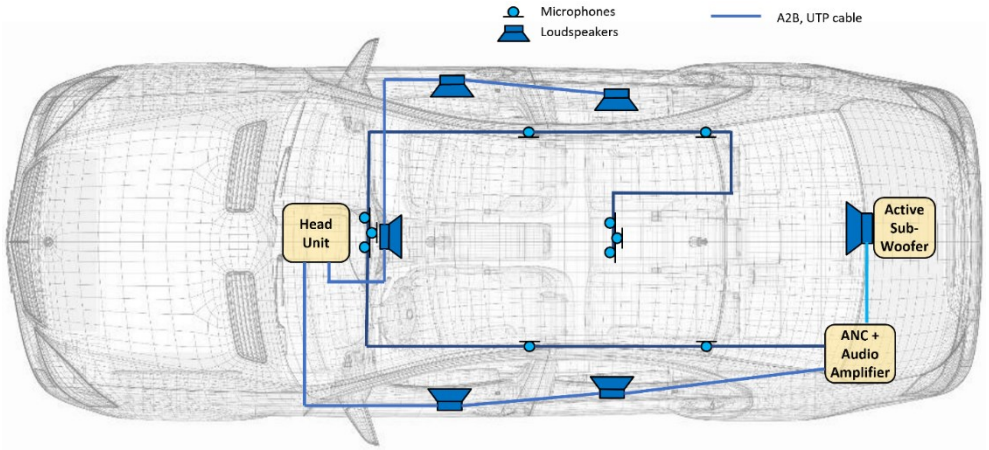


Figure 35. Example of A^{2}B car cabling. Three different A^{2}B bus are employed, and they are represented by different shades of blue.

However, A^2B bandwidth is limited to 50 Mbps, allowing to transport up to 32 audio channels between 11 devices. Although this capability may satisfy most of today's applications, it is much lower than the bandwidth offered by gigabit ethernet (i.e., 1 Gbps). Future technologies will require the transport of a massive amount of data, coming from several cameras, radars and lidars in addition to the increasing number of sensors and actuators. As suggested in [53], a mixed system of A^2B and gigabit ethernet may be the solution for future applications. In fact, A^2B is suitable for audio transport and sensors acquisition, while an ethernet backbone guarantees enough bandwidth for other demanding devices.

In the next sections, the architecture introduced in Section 2.1 will be employed in the automotive application called Individual Listening Zone (ILZ), a proprietary technology developed by ASK Industries S.p.A. ILZ consists in delivering personalized audio content in an environment where multiple listeners are present, in this case the car cabin. In literature there exists several studies dealing with signal processing techniques [54]–[59], but here the focus will be on the design of a distributed A^2B amplifier for ILZ application.

3.1.1 ILZ Architecture

Typically, sound zoning applications (e.g., ILZ) rely on loudspeaker arrays to control the sound field by means of strategies such as pressure matching (PM) or acoustic contrast control (ACC). In addition, loudspeakers positioning plays a critical role in final result. Since this application requires a high number of loudspeakers, it was chosen to adopt an A^2B signal distribution for the reasons explained in the previous section. Thanks to this solution and in order to keep the versatility in loudspeakers positioning, it was opted for equipping each loudspeaker with its own amplifier. In addition, this choice is justified by the fact that class-D amplifiers efficiency is increasing, their audio performance gets better and better and they are cost-effective. Finally, several IC models offer a digital input (typically in an I^2S or TDM format), thus allowing to avoid additional D/A and A/D conversions.

This solution allows to easily realize array topologies such as linear or planar with variable dimensions. An example of the proposed system is shown in Figure 36. Here, the standard car audio system or devices necessary for other applications (e.g., microphones, accelerometers) are not depicted, but only examples of

loudspeaker arrays topologies for the ILZ application. In particular, three types of arrays are represented:

- *linear array*: it is composed by a certain number of loudspeakers positioned in line. It may be positioned above the dashboard.
- *planar array*: in this case square or rectangular solutions are proposed, for example of dimension 3x3 or 3x4. If sufficiently thin, it is suitable for the car roof.
- *headrest array*: some loudspeakers are positioned at the sides of the headrest.

The processing unit is equipped with a certain number of A^{2}B main transceivers, and it implements the signal processing for ILZ. In addition, it should interface with the head unit, and it may integrate an amplifier to drive, for example, the standard car audio system. During the development of the processing algorithm or when trying different system configurations, it is beneficial to interface the processing unit with a development platform, namely a PC or a rapid prototyping system. For this purpose, during this work it was designed and realized an interface board able to convert A^{2}B data to USB. The design of this board, which from now on will be called *A^{2}B Streamer*, will be described in the next section.

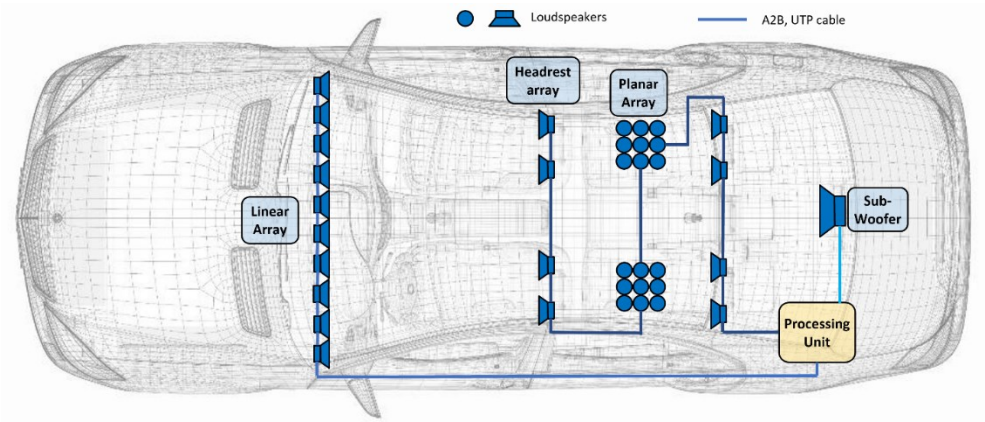


Figure 36. Example of a possible A^{2}B audio distribution for the ILZ application. Different A^{2}B lines are represented by shades of blue.

3.1.2 A²B Streamer

The A²B Streamer is an interface board that has the purpose of converting A²B data into another format that can be read for example by a PC or by a development platform like rapid prototyping systems. For this purpose, the standard USB Audio Class 2.0 (UAC) was chosen. In fact, UAC-2.0 is one of the most diffused protocols in audio soundcards as it implements a multichannel bidirectional communication (the standard endpoint allows to transfer up to 32 channels in each direction at 32-bit depth and 48 kHz of sample rate) with a minimum latency of 500 µs [60]. In addition, UAC-2.0 is natively supported by the most common operating systems like macOS, Linux, and Windows.

A²B Streamer hardware

The A²B Streamer was developed to satisfy the following requirements:

- Support to A²B transceivers AD242x
- Support to A²B transceivers AD243x
- Possibility to switch between main and subordinate node for the AD242x transceivers
- Capability of a maximum of 32 inputs and outputs
- UAC-2.0 support

The A²B Streamer was realized by employing evaluation boards, thus guaranteeing versatility of the system during the development process. In particular, support to A²B was implemented thanks to the evaluation boards EVAL-AD2428WD1BZ and EVAL-AD2435WA3LZ. The former supports the transceivers of the AD242x family, while the latter supports the transceivers of the AD243x family. Both boards can be configured as main or subordinate nodes and they are equipped with a SigmaDSP and an audio codec, which offers a stereo line input and a stereo line output. The SigmaDSP can be employed for signal processing, audio routing and it offers one optical Sony/Philips Digital Interface Format (S/PDIF) input and one output. In addition, the boards have a connection for the USBi interface board developed by Analog Devices to program the SigmaDSPs and to configure the A²B busses with SigmaStudio. Figure 37 and Figure 38 show the two evaluation boards with highlighted the main components and connections. The lines called *BCLK*, *FCLK*, *DATA_OUTx*, and *DATA_INx* are the TDM lines connected with the A²B transceiver, and they are employed to exchange audio with the nodes over the

bus. The EVAL-2435WA3LZ is also equipped with the power supply section necessary to inject power over the A^2B bus.

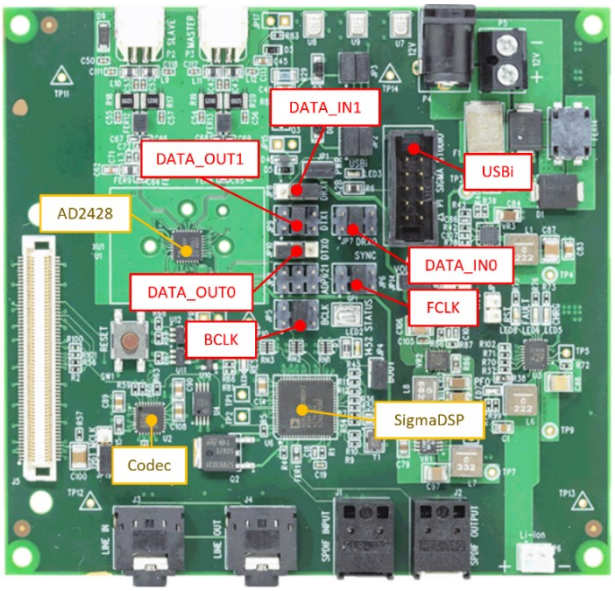


Figure 37. Evaluation board EVAL-AD2428WD1BZ. The yellow boxes indicate the main components, while the red boxes indicate the connections.

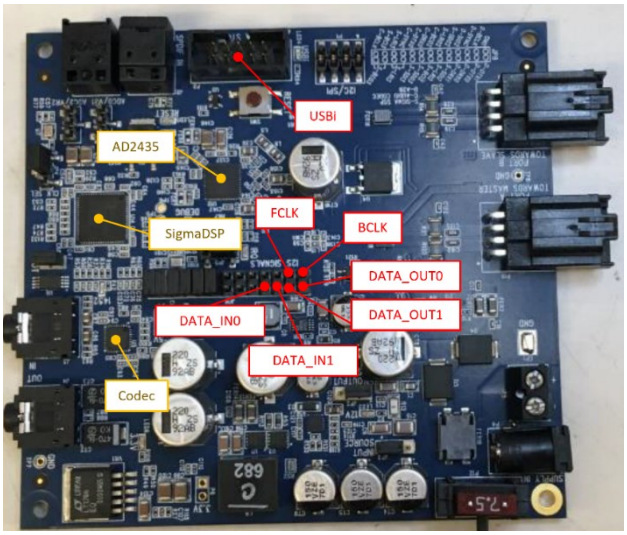


Figure 38. Evaluation board EVAL-2435WA3LZ. The yellow boxes indicate the main components, while the red boxes indicate connections.

The data conversion to USB was implemented thanks to the xCORE-200 microcontroller developed by XMOS. It is a 32-bit multicore microcontroller that can execute multiple real-time tasks simultaneously and it communicates between tasks using a high-speed network, thus allowing to implement software functions that typically require dedicated hardware. The evaluation board employed for the realization of the A^{2}B Streamer is the XK-AUDIO-216-MC-AB that mounts the microcontroller XE216-512-TQ128. This board offers several useful features: USB and ethernet interfaces, GPIO, an 8-channel ADC and an 8-channel DAC, and finally a reference source code implementing an UAC-2.0 compliant endpoint and the ethernet MAC layer. The main components and connections of the evaluation board are represented in Figure 39. *ADC routing* and *DAC routing* are the TDM data lines connecting the microcontroller to the ADC and DAC respectively, while TDM clocks are on a different connector (*TDM clocks*). The *PLL* section contains a PLL multiplier that can be employed to generate a local low-jitter master clock at 24.576 MHz and a CS2100CP by Cirrus Logic, which is a fractional clock multiplier commonly employed to recover a clock signal from external sources (e.g., S/PDIF input, AVB PTP synchronization signal).

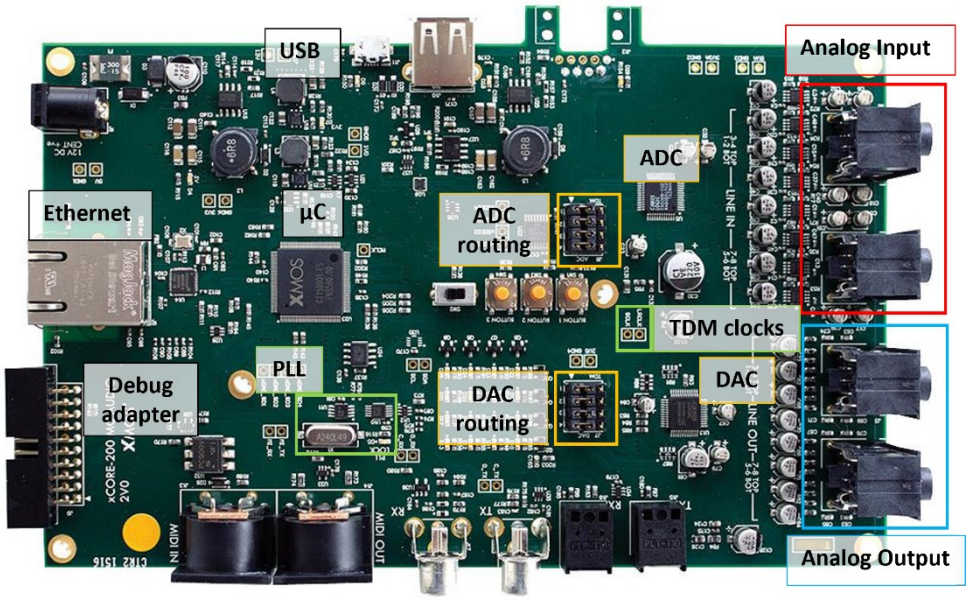


Figure 39. XMOS evaluation board with the main components and connections highlighted.

Finally, Figure 40 shows the block diagram of the A²B Streamer. The boards are power supplied at 12 V, so the car battery can be employed as power supply. Since the A²B line of the AD2435 transceiver should provide up to 50 W, the 12 V external power supply must be dimensioned accordingly. The TDM and I²C lines connect the XMOS to the A²B evaluation boards. TDM is employed to exchange audio data, while I²C can be used to configure the A²B networks as an alternative to USBi programmer. This option allows to adapt to different use cases when the A²B transceivers should act as main nodes. For example, when the A²B nodes need to be changed frequently, it is not convenient to build and flash different microcontroller firmware. In this case, the A²B nodes can be easily configured by the graphical environment SigmaStudio. On the contrary, when the A²B Streamer is employed with a well-defined A²B network, the initial configuration commands can be embedded in the microcontroller firmware. Also, TDM routing was designed to adapt to different scenarios. The A²B transceivers offer a maximum of two TDM input and two TDM output lines, and they can be configured with several options. The most common are TDM2 or I²S, TDM8 and TDM16, where the number indicates the number of audio channels on each line. The XMOS μC can manage up to 32 input and 32 output channels, thus the number of physical lines changes accordingly. If the TDM interface is configured for a TDM8, the number of physical lines will be 4 inputs and 4 outputs, while for a TDM16 there will be only 2 input and 2 output lines. Depending on the number of channels to be transferred for each A²B bus, the TDM connections can be changed dynamically by connecting the appropriate wires between the *ADC* and *DAC routing* connectors of the XMOS board and the TDM connectors of the A²B boards. If for example an A²B bus must accommodate 32 input channels, the TDM interfaces should be configured for a TDM16, and the other A²B main node will not have any input available. On the contrary, when both A²B busses have some input and output channels to be transferred, it is more convenient to employ a TDM8 since more physical lines are available and they can be routed to the different transceivers.

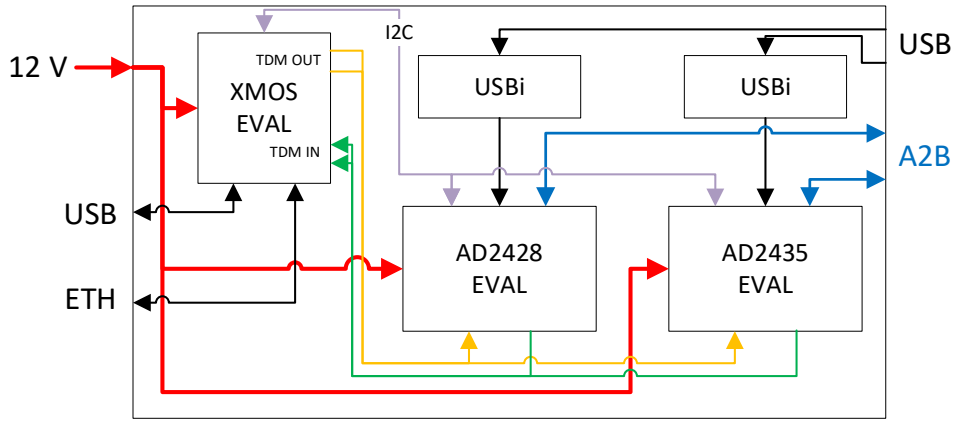


Figure 40. Block diagram of the A2B Streamer.

A²B Streamer firmware

The A²B Streamer firmware has the purpose of converting data between A²B and USB, thus allowing a PC to exchange audio with the A²B nodes.

The USB firmware is based on the reference application by XMOS [61], which provides an USB Audio Class compliant device over USB 2.0. The reference application has several optional features that can be enabled during the compilation process, such as: S/PDIF and Alesis Digital Audio Tape (ADAT) transmitters and receivers, Musical Instrument Digital Interface (MIDI) over Universal Asynchronous Receiver-Transmitter (UART) interface, and a PDM interface, in addition to the I²S/TDM interface, which can be configured in master or slave mode. All these options have been disabled except for the I²S/TDM interface, which is employed to communicate with the A²B transceivers.

The application consists of several tasks running in parallel on different cores of the microcontroller [62]. The block diagram of the firmware is shown in Figure 41, where the main blocks are represented:

- *USB XUD*: it handles the low-level USB I/O.
- *Endpoint0*: it is a control endpoint that is required by each USB device. The operations that are managed by this task are enumeration, audio configuration, and upgrade request (i.e., Device Firmware Upgrade, DFU in Figure 41). Enumeration occurs when the device is connected to the host, and it is interrogated on its functionalities. Audio configuration involves operations like setting the sample rate or volume control.

- *Endpoint Buffer*: it is responsible for the samples exchange between the *decoupler* and the *USB XUD* tasks.
- *Decoupler*: it handles the audio data exchange between the *Endpoint Buffer* and the *Audio Driver*. Audio data to and from the A*udio Driver* is organized into FIFOs, and when the *Endpoint Buffer* requires samples, a buffer is fetched from the FIFO.
- *Audio Driver*: it is responsible for driving the I^2S/TDM port and for sending and receiving samples to/from the *Decoupler*. Further details will be given in the next paragraphs.
- *A^2B Host*: this optional task is enabled when the two A^2B transceivers operate as main nodes and when they need to be configured by the XMOS instead of SigmaStudio. This operation is carried out by means of an I^2C connection. If a simple initial configuration is needed, a *.c* file containing a set of I^2C commands can be exported by SigmaStudio and embedded in the microcontroller. This implementation does not perform line fault diagnostic or rediscovery operations. On the other hand, if more A^2B functionalities are required A^2B Plug-and-Play can be implemented as described in Section 2.3.

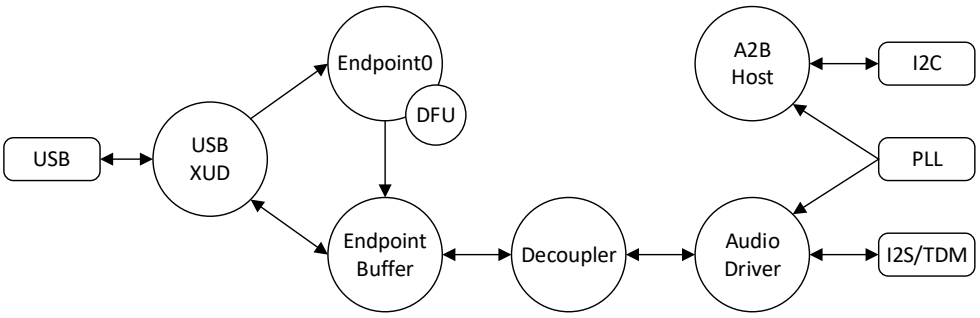


Figure 41. Main tasks of the USB firmware for the A^2B Streamer.

The I^2S/TDM port can be configured in master or slave mode. In fact, when the A^2B transceivers operate as main nodes, they are TDM slaves, as the bit clock is employed to synchronize all the subordinate nodes. On the contrary, when the A^2B transceivers operate as subordinate nodes, they are TDM masters since they reconstruct the bit clock and the frame clock from the bus clock. The working mode of the

TDM interface of the XMOS is set during the compilation phase and it cannot be changed during the execution.

When the XMOS TDM interface is in master mode, the *Audio Driver* task is responsible for deriving the bit clock and the frame clock from the incoming master clock, which can be generated by the external oscillator or by the PLL. The master clock must be a multiple of the desired bit clock. Table XIII indicates common values form different I^2S/TDM configurations and at different sample rates. The A^2B Streamer can be configured to work at 48 kHz in TDM8 or TDM16 modes. When the TDM8 is selected, the master clock can be provided by the on-board PLL, while when the TDM16 mode is desired, the CS2100CP must be configured to double its reference frequency of 24.576 MHz, thus providing a master clock with a frequency of 49.152 MHz.

On the contrary, when the TDM interface is in slave mode, the *Audio Driver* is clocked by the external bit clock signal. In this case, the need for a master clock was removed by using just the bit clock signal. Frequencies for the two available modes are 12.288 MHz for the TDM8 and 24.576 MHz for the TDM16.

Table XIII - Common rates of master clock and bit clock

I^2S/TDM	**Sample rate [kHz]**	**Master clock [MHz]**	**Bit clock [MHz]**
I^2S	48	24.576	3.072
TDM8	44.1	22.5792	11.2896
TDM8	48	24.576	12.288
TDM16	48	49.152	24.576

The A^2B Streamer firmware allows to reach 32 input and 32 output channels at 48 kHz for each TDM interface configuration, namely master or slave and TDM8 or TDM16. The employed XMOS microcontroller offers 2 tiles, namely a set of cores, memory, and I/O. Each tile is composed of 8 logical cores, 256 kByte of SRAM and 8 kByte of OTP memory, and 32 channels of communication between the cores. Table XIV shows the resources occupation on the XMOS microcontroller for the four firmware options of the A^2B Streamer.

Table XIV - XMOS resources occupation.

Firmware	Cores occupation		Chanends occupation		Memory occupation [kB]	
	Tile[0]	Tile[1]	Tile[0]	Tile[1]	Tile[0]	Tile[1]
TDM8 master	2/8	4/8	6/32	21/32	26.5/262	44.7/262
TDM8 slave	2/8	4/8	4/32	20/32	23.6/262	42.7/262
TDM16 master	2/8	4/8	7/32	22/32	34.8/262	47.8/262
TDM16 slave	3/8	4/8	7/32	21/32	25.7/262	43.4/262

3.1.3 Design of the ILZ A²B Amplifier

An active loudspeaker based on an A²B amplifier was developed for the ILZ application. Firstly, project requirements were identified:

- System modularity: it should be possible to modify the loudspeaker number and arrangement when needed, without redesigning the system architecture.
- A dedicated amplifier should be provided for each loudspeaker.
- Each amplifier and loudspeaker should support a power output of 5 W_{rms} on a 4 Ω load.
- Array topologies like linear and planar with 9 loudspeakers should be supported.

Two different system architectures were developed: the first one is based on the AD242x transceivers family, while the second one is based on the AD243x family. Both solutions present different advantages and use cases that will be described in the next sections. Both solutions were developed following the stages represented in Figure 42. Firstly, the system architecture was defined by comparing different solutions. Then, the main components of the amplifier were chosen and the PCB was designed and successively realized. Finally, audio, electrical, and thermal performance were measured.

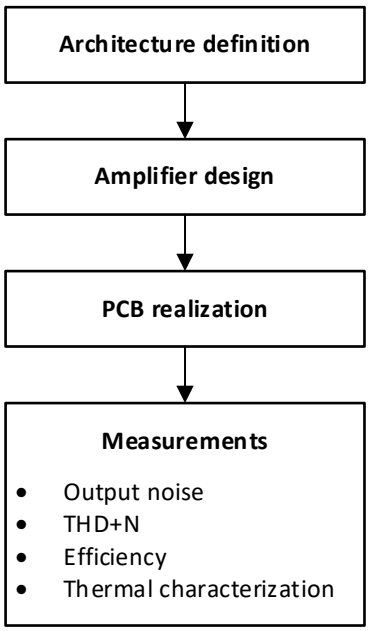


Figure 42. Flowchart of the design of the ILZ A^2B amplifiers.

AD242x architecture

The main difference between the AD242x and AD243x transceivers is that the former does not allow for power transfer together with data. Hence, this solution requires that in addition to an A^2B data line, a power supply connection must be distributed too.

Initially, two solutions were considered. The block diagram of the *solution 1* is shown in Figure 43. Here, it is represented a single A^2B bus composed by 9 subordinate nodes. Each node is equipped with an A^2B transceiver and a mono class-D power amplifier. To evenly subdivide the current load, the power supply (12 V) is distributed with a star and daisy-chain mixed topology. The power supply section of each board is composed by a linear regulator with an output voltage of 3.3 V. This voltage level is necessary to supply the digital components of each board (e.g., digital section of power amplifier). The A^2B transceivers receive the power supply by the A^2B phantom power, thus allowing to avoid an additional DC/DC converter and to perform diagnostic in case of a fault on the main power supply. A^2B allows to transfer GPIO and I^2C commands together with audio data, thus it is not necessary to add other communication connections. I^2C may be necessary for example for the initial

amplifier configuration. With this system architecture a single A^{2}B bus can support up to 10 amplifiers, due to its limitations on the number of subordinate nodes. For example, 3x3 planar arrays or linear arrays with up to 10 loudspeakers can be realized. In addition, the number of main nodes can be increased up to 4, thus reaching a maximum capability of 40 audio channels.

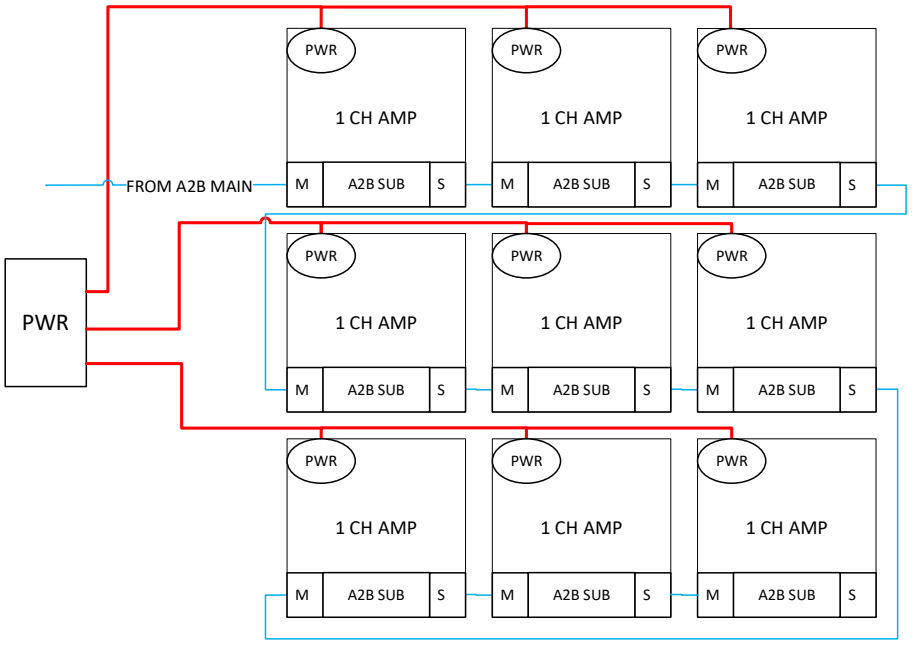


Figure 43. Block diagram of the solution 1. Red lines represent power connections, while blue lines are A^{2}B connections.

Solution 2, which is shown in Figure 44, also consists in an A^{2}B bus to transfer audio information, but it is composed by two different boards:

- *A^{2}B node*: this board is equipped with an A^{2}B transceiver, which acts as a subordinate node, and with a power amplifier.
- *Amp node*: this board mounts only the power amplifier and it receives audio data and configuration commands through TDM and I^{2}C lines driven by the A^{2}B node.

Both boards are power supplied by an external 12 V, which is distributed with a star and daisy-chain mixed topology as for *solution 1*. The power supply section is the same too. Up to three *amp nodes* can be connected to a single *A^{2}B node*. The

reason is twofold: to keep the length of the TDM lines limited and because most of the I²C devices offer up to 4 addresses. The amplifiers can be driven individually as they can select their channel on the TDM line and pass the others to the next amplifiers. This solution allows to overcome the limited number of subordinate nodes of a single A²B bus. In fact, by employing *solution 2*, it is possible to saturate the number of subordinate nodes and the number of channels available in a single bus. This architecture allows transferring up to 128 channels by employing four A²B networks (32 channels for each network).

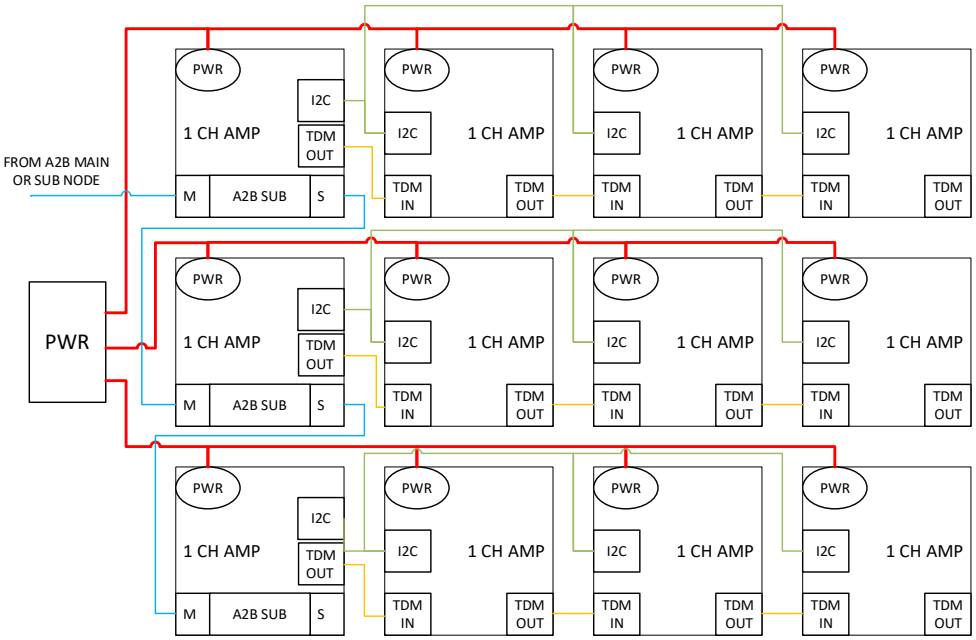


Figure 44. Block diagram of the solution 2. Red lines represent power connections, blue lines are the A²B bus, yellow lines are for TDM, and green lines are for I²C.

The advantages and disadvantages of both solutions were considered to choose the one that better fitted in project requirements. *Solution 1* is optimal in terms of modularity because it involves the development of a single board able to drive a loudspeaker. Nevertheless, the maximum number of channels is quite limited because only 10 subordinate nodes can be connected in an A²B bus. To reach a capability of 40 channels, four A²B main nodes must be employed, thus increasing the system complexity and cost. Wiring is simple because an UTP cable is employed for A²B, and a pair of power cables are employed for power supply.

Solution 2 allows reaching a high number of channels (i.e., 128 audio channels), even though a reasonable number of channels for practical implementations is available by employing just two A^2B busses (i.e., 64 audio channels). Modularity requirement is still satisfied although two different boards must be employed. Wiring is more complicated with respect to *solution 1*. In fact, in addition to the UTP cable for A^2B and the pair of power supply cables, I^2C and TDM connections must be distributed between the boards by means for example of flat cables. Finally, it must be pointed out that, when necessary, *solution 2* can be converted to *solution 1* just by employing the *A^2B node* board.

After the comparison it was chosen to adopt the *solution 2*, as it is more flexible at the price of a more complicated wiring. Afterward the main components were identified:

- **Loudspeaker**: typically, the height of loudspeakers is in the range of 5 to 10 cm, thus limiting the loudspeaker positioning. ASK S.p.A. has developed and patented for the ILZ application the loudspeaker *UltraFlat* that has a height of just 2.5 cm and a diameter of 5 cm. This allows installing the loudspeaker in non-conventional positions such as the car roof, the dashboard, or the headrests. The *UltraFlat* loudspeaker dimensions, and specifically the diameter, were used to define the PCB dimensions. In fact, array topologies require to arrange the loudspeakers side by side and to avoid gaps it was decided to design the PCB with sides no longer than 5 cm. Finally, the loudspeaker was designed to include the heatsink for the power amplifier.
- **Amplifier**: given the reduced dimension of the PCB, it was opted for an integrated class-D amplifier. After the research of the off-the-shelf devices, the TAS6421-Q1 by Texas Instruments was chosen. It is a 1-channel class-D amplifier qualified for automotive applications. Its main specifications are presented in Table XV. In addition, the TAS6421-Q1 offers several useful advanced functionalities for diagnostic and protection: check for open and shorted load, output-to-battery or ground shorts, output current limiting, overtemperature warning and shutdown, and lastly clip detection. The wide power supply voltage range allows employing the battery voltage as power supply. Moreover, this voltage level allows reaching the required output power of 5 W_{rms} on a 4 Ω load. The low output noise voltage is a critical parameter since a high number

of loudspeakers, and so amplifiers, will be installed inside the car. The switching frequency is another relevant parameter. In fact, the higher is the switching frequency the smaller is the output filter composed by at least an inductor and a capacitor. Reducing dimensions of the inductor is fundamental to reduce costs and weight. Finally, the TAS6421-Q1 offers two essential features to implement the *solution 2*: 4 configurable I^2C addresses and TDM4 or TDM8 capability.

Table XV - TAS6421-Q1 main specifications for a 4 Ω load and power supply equal to 14.4 V

Parameter	Value	
Power supply range	4.5 ÷ 26.4	[V]
Input type	I^2S/TDM4/TDM8	
Sample rate	44.1/48/96	[kHz]
Output power	22	[W]
Output noise voltage (zero input, gain level 2)	55	[μV]
Switching frequency	2.1	[MHz]
THD+N (output power 1 W)	0.02	[%]
I^2C addresses available	4	

Once the main components were identified, the two boards were designed and realized. The block diagrams of the *A^2B node* and the *amp node* are shown in Figure 45 and in Figure 46 respectively. As it can be seen in Figure 45, the AD2428, namely the A^2B transceiver, receives data and power supply from the A^2B bus. The amplifier, which instead has a higher power consumption, is powered with an external 12 V power supply. A 3.3 V linear voltage regulator serves to power the digital section of the audio amplifier. The TAS6421-Q1 is configured at the startup of the system by means of the I^2C connection with the AD2428. This process is managed by the target processor connected to the main node, which configures each amplifier by sending the I^2C commands over A^2B. The amplifier of the *amp nodes* that are connected to an *A^2B node* are configured to read a different channel from the TDM line, leaving the others unchanged. Both I^2C and TDM lines of an *A^2B node* are then linked to a connector that is employed for the connection of the *amp nodes*. This last

board is identical to the *A²B node* except for the AD2428 section, which is not present as it can be seen in Figure 46.

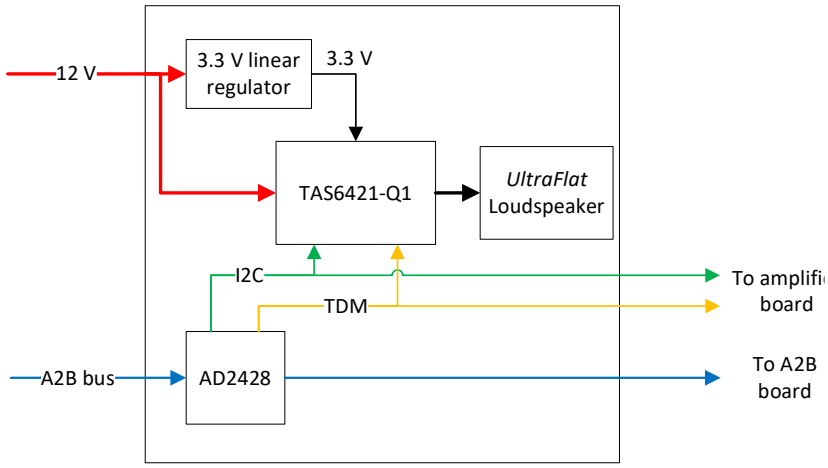


Figure 45: Block diagram of the A²B amplifier board.

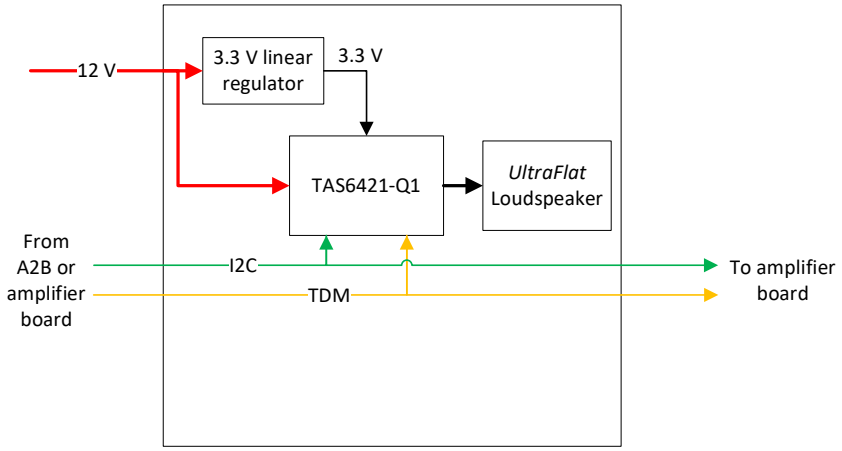


Figure 46. Block diagram of the amplifier board.

Figure 47 and Figure 48 show respectively the top view and the bottom view of the 3D model of the amplifier board. The assembly of the *UltraFlat* loudspeaker and the amplifier is shown in Figure 49. As it can be seen in this figure, the loudspeaker was designed to be mounted over the top side of the PCB, with its center part acting as a heatsink. In Figure 47 are also indicated the centering holes that have the aim to

keep the loudspeaker in position, and the air flow gaps necessary for the air flow on the rear side of the loudspeaker.

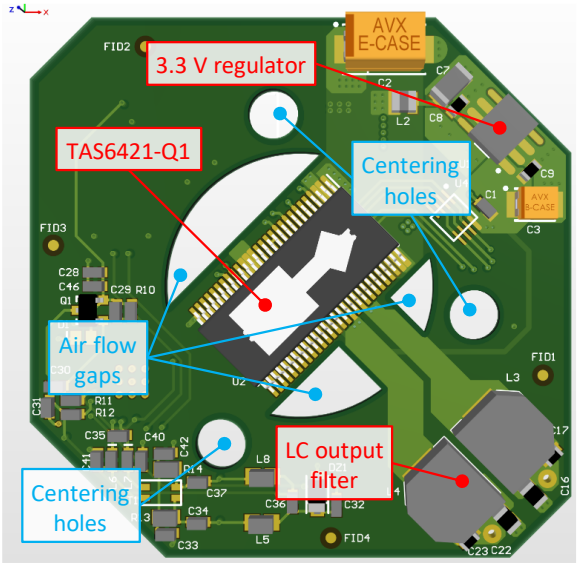


Figure 47. 3D model of the A²B node (top view) with highlighted the main components and mechanical parts.

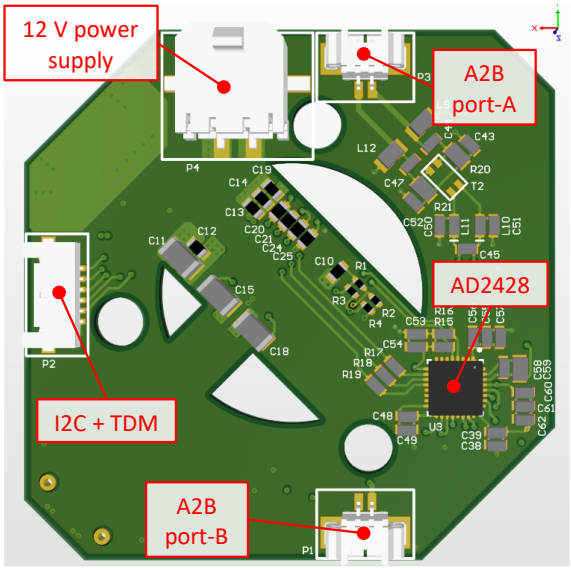


Figure 48. 3D model of the A²B node (bottom view) with highlighted the main components.

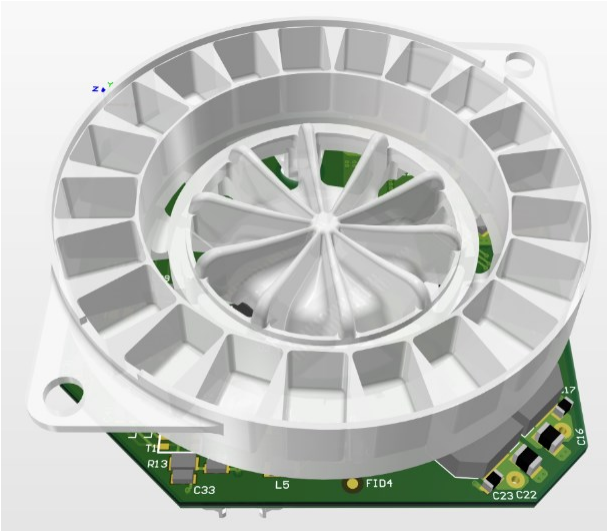


Figure 49. 3D model of the A²B node with the UltraFlat loudspeaker which is supplied by ASK Industries S.p.A. (the diaphragm is hidden).

AD243x architecture

This architecture leverages the new generation of A²B transceivers, i.e., the AD243x family, for the realization of a distributed power amplifier for the ILZ application. The AD243x transceivers allow transferring data together with power on the same UTP cable. The maximum power for each bus is 50 W. In addition, two other improved features are the maximum number of subordinate nodes, which is 16, and the global maximum cable length of each bus, which is 80 m.

The block diagram of the proposed solution is shown in Figure 50, where it is represented one A²B bus composed by the main node and up to 16 subordinate nodes equipped with a 1-channel power amplifier. Even with this solution the amplifiers can be remotely configured by the main node by means of I²C commands over A²B. This solution allows to dramatically reduce the weight and cost of the cables, as a single UTP cable is employed to distribute both data and power. A²B specifications state that the voltage level on the bus must be 24 V to reach the maximum power capability of 50 W. With this solution it is possible to combine up to four A²B busses, thus reaching 64 audio channels. The drawback of this architecture is that if 16 power amplifiers are connected on the same bus, the power available for each of them is 3.125 W, which do not satisfy the initial project requirements. If the final application will require 5 W_{rms} for each amplifier, an A²B bus must have a maximum of 10

subordinate nodes connected, thus reaching a maximum of 40 audio channels when four busses are employed.

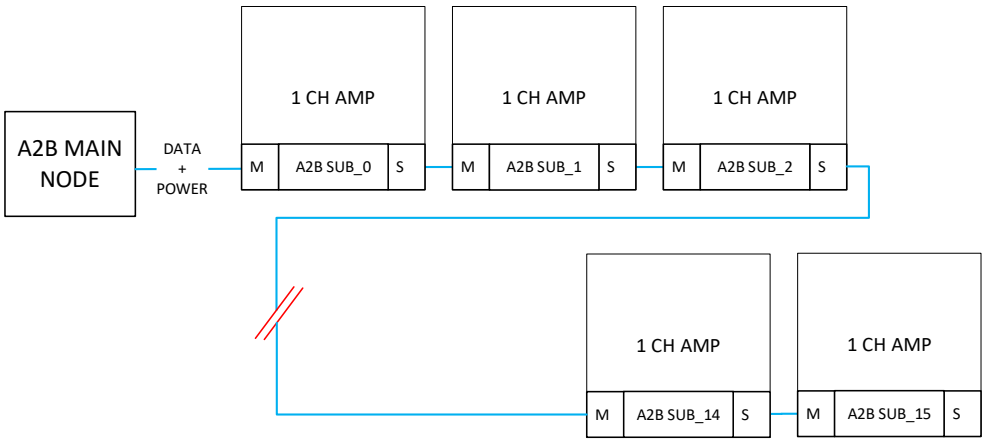


Figure 50. Block diagram of the architecture based on the AD243x A²B transceivers. The blue connections represent the A²B bus, which carries data together with power.

The starting point for the design of this amplifier was the AD242x solution. The loudspeaker, PCB dimension and amplifier were not changed.

The block diagram of the designed amplifier is shown in Figure 51. The supply voltage distributed by the A²B bus is 24 V. This voltage level is excessive for the audio amplifier and the A²B transceiver, whose supply voltage ranges are shown in Table XVI. In particular, the TAS6421-Q1 needs three different supply voltages: PVDD for the output power stage, VBAT for the internal voltage references and the gate driver and VDD for the low voltage circuitry. Hence, it was decided to equip the board with a DC/DC Buck converter that has an output voltage of 8 V. As it can be seen in Figure 51, the 8 V voltage is employed to supply the AD2435 and the VBAT section of the TAS6421-Q1. In addition, to investigate on the audio amplifier efficiency, the board was provided with the possibility to change the PVDD supply voltage of the TAS6421-Q1 between 24 V and 8 V. Both levels allow to reach the desired output power of 5 W_{rms}. Finally, the 3.3 V required for the low voltage circuitry of the amplifier is supplied by the internal linear regulator of the AD2435. The A²B transceiver communicates with the amplifier through an I²C line for the initial configuration and an I²S/TDM connection for audio data. Since in this architecture

each A^2B transceiver drives only one amplifier, it is possible to configure this line as I^2S to reduce electromagnetic emissions.

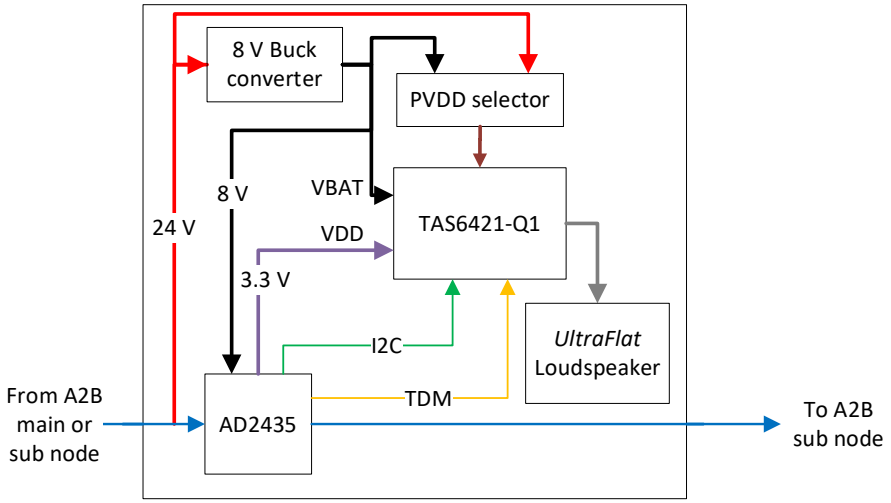


Figure 51. Block diagram of the amplifier board based on the AD2435 A^2B transceiver.

Table XVI - Supply voltage ranges of the TAS6421-Q1 and the AD2435

Parameter	Value	
TAS6421-Q1: PVDD	4.5 ÷ 26.4	[V]
TAS6421-Q1: VBAT	4.5 ÷ 18	[V]
TAS6421-Q1: VDD	3 ÷ 3.5	[V]
AD2435: VIN	3.7 ÷ 9	[V]

Figure 52 and Figure 53 show respectively the top view and the bottom view of the 3D model of the amplifier board. As it can be seen, the PCB shape did not change except for the addition of two screw holes that improve the attachment of the loudspeaker to the PCB. Consequently, air flow gaps were modified to keep the same surface of the AD242x version. In addition, the A^2B lines on port-A and port-B connectors were doubled with respect to the previous version, to accommodate to the higher current. In fact, thin connectors were chosen to limit the height of the board. However, they support low-section wires, so the number of wires were doubled. Finally, Figure 54, Figure 55, and Figure 56 show the realized prototype.

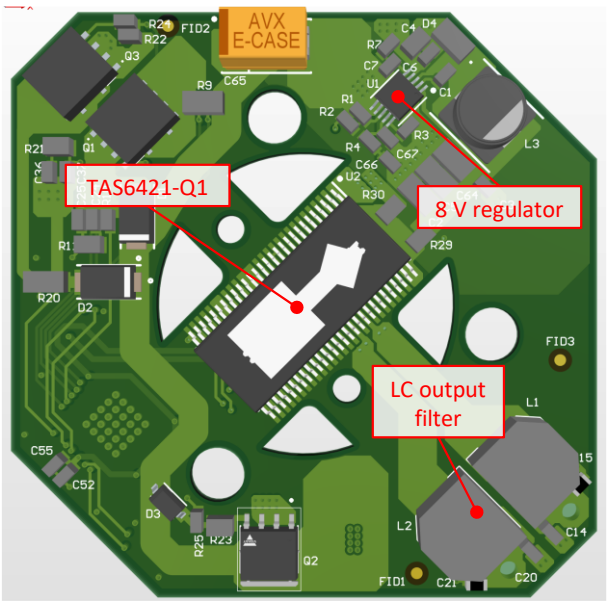


Figure 52: 3D model of the amplifier board (top view) with highlighted the main components.

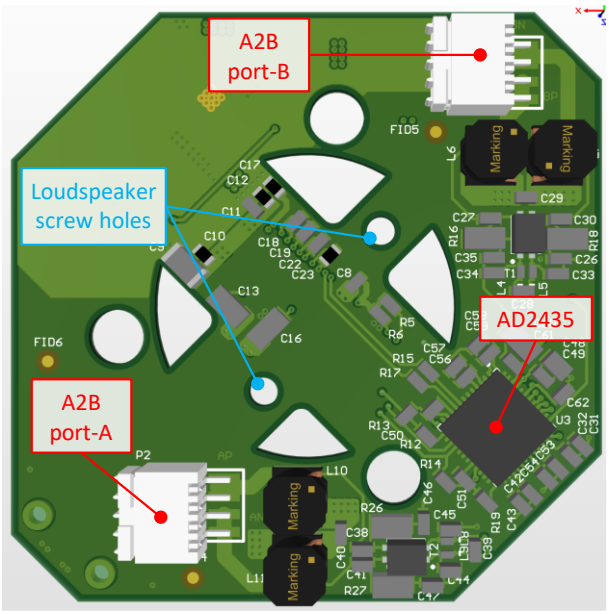


Figure 53. 3D model of the amplifier board (bottom view) with highlighted the main components and mechanical parts.

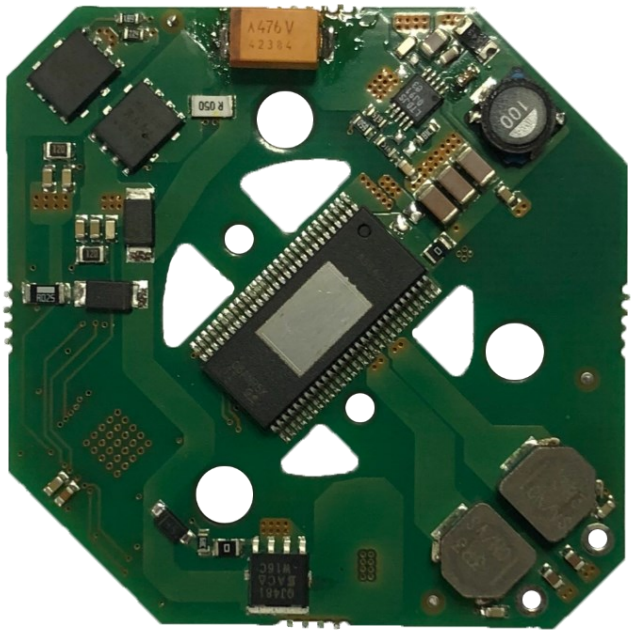


Figure 54. Top view of the realized AD243x amplifier prototype.

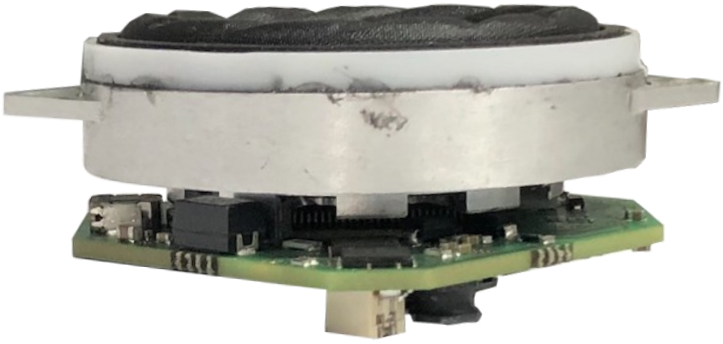


Figure 55. Assembly of the AD243x amplifier and the UltraFlat loudspeaker.

Figure 56. Example of a loudspeaker array with three AD243x amplifiers and UltraFlat loudspeakers in a 3D printed case.

3.1.4 Measurements

This final section will present the experimental results of the characterization of the AD242x and the AD243x solutions. Since a prototype of the AD242x amplifier board was realized first, the electrical characterization was carried out on this board.

AD242x amplifier

The two main parameters that were measured are the Total Harmonic Distortion plus Noise (THD+N) and the amplifier efficiency. The measurement setup that was employed is shown in Figure 57. The AD242x amplifier, namely the Device Under Test (DUT), receives the audio signals from the A^2B main node, which in these tests is the evaluation board EVAL-AD2428WD1BZ (described in Section 3.1.2). The initial configuration of the A^2B nodes is carried out by employing SigmaStudio running on a PC, which is connected to the main node through USB. The measuring instrument is an APx58x by Audio Precision, a recognized standard instrument for audio measurements. To avoid additional A/D and D/A conversions, it was exploited the S/PDIF output of the Audio Precision to send the test signal to the A^2B main node and so to the amplifier. The amplifier is then connected to a 4 Ω resistive load that simulates the loudspeaker. Before the voltage across the resistor is acquired by the balanced input of the Audio Precision, the signal passes through a passive filter, i.e., the AUX-0025, developed by Audio Precision that must be employed when measuring switching amplifiers. The aim of this filter is to reduce the out-of-band switching signal components, which can lead to inaccurate results due to the slew rate limiting of the measuring instrument. Finally, the PC is also connected through

USB to the Audio Precision to acquire the measurement results. During the measurements the power supply voltage of the DUT was 13.8 V.

Firstly, idle power consumption and output noise with zero input were measured. Results are shown in Table XVII.

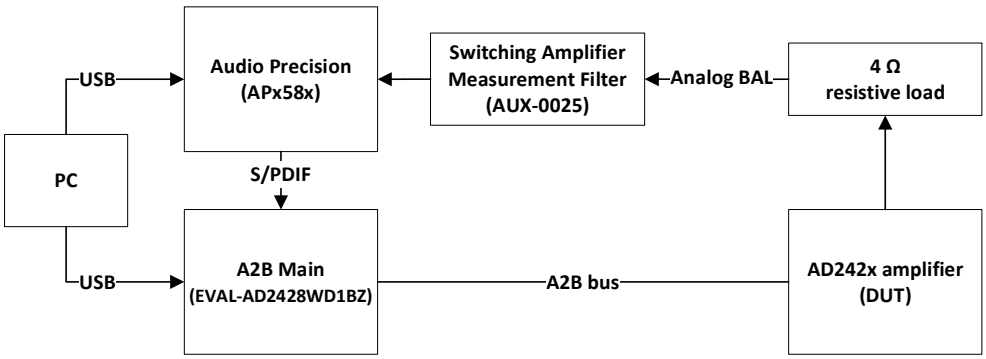


Figure 57. Block diagram of the measurement setup.

Table XVII - Idle output power and output noise measured on the AD242x amplifier board.

Parameter	Value	
Idle power consumption	0.65	[W]
Output noise (zero input)	100	[μV]

Successively, the THD+N ratio was measured as a function of frequency and output power. Initially the Audio Precision was set to output a 1 kHz sinusoidal signal, with an amplitude so that the amplifier output power was 1 W. The THD+N ratio was then measured over the audio bandwidth (i.e., from 20 Hz to 20 kHz) at two switching frequencies of the power amplifier, namely 384 kHz and 2.1 MHz. The results are shown in Figure 58. As it can be seen, the THD+N is almost constant over the audio bandwidth, and it is about 0.01 % for both switching frequencies.

Then, the THD+N ratio was also measured as a function of the output power, again for the two switching frequencies. The output power was calculated as:

$$P_{out} = \frac{V_{out-rms}^2}{R_{nom}} \tag{3.1}$$

where $V_{out\text{-}rms}$ is the value read by the audio precision, and R_{nom} is the nominal values of the resistive load, namely 4 Ω. The results are shown in Figure 59, and as it can be observed the two curves nearly follow the same trend.

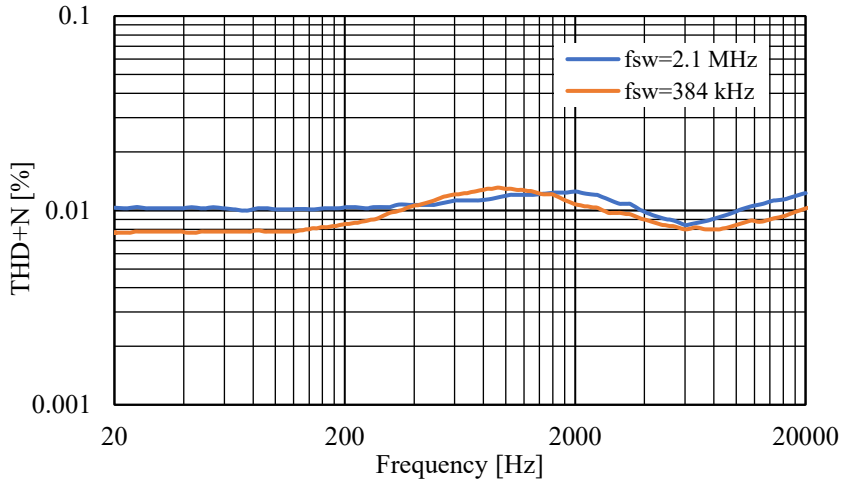


Figure 58. THD+N ratio of the AD242x amplifier as a function of frequency, at two switching frequencies: 384 kHz (orange line) and 2.1 MHz (blue line).

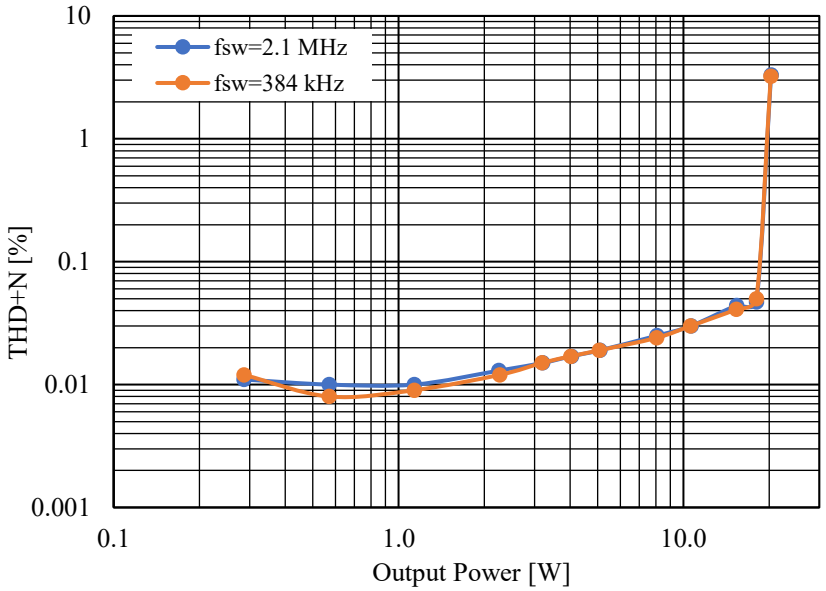


Figure 59. THD+N ratio as a function of output power at two switching frequencies: 384 kHz (orange line) and 2.1 MHz (blue line).

Finally, in Table XVIII is indicated the output power at some typical values of THD+N for the two switching frequencies.

Table XVIII – Output power values at typical THD+N levels for the two switching frequencies.

THD+N [%]	**P_{out} [W] (@ 384 kHz)**	**P_{out} [W] (@ 2.1 MHz)**
0.01	1.49	0.9
0.1	18.49	18.36
1	18.92	19.18
10	22.9	22.85

Afterward, it was measured the amplifier efficiency as a function of the output power and at the two switching frequencies. The test signal was a 1 kHz sinusoidal tone. The efficiency was calculated as:

$$\eta = \frac{P_{out}}{P_{in}} 100 \tag{3.2}$$

where P_{in} is the voltage supply (namely 13.8 V) multiplied by the supply current and P_{out} is obtained as in (3.1).

Finally, it was calculated the power dissipated by the amplifier board as:

$$P_{diss} = P_{in} - P_{out} \tag{3.3}$$

The efficiency and the power dissipated are shown in Figure 60 and in Figure 61 respectively. As it can be seen in Figure 60, the efficiency at the switching frequency of 384 kHz is higher for an output power higher than 2 W. However, since the range of use of the amplifier is up to 5 W, it was decided to keep a switching frequency of 2.1 MHz to reduce the dimension and cost of the output filter.

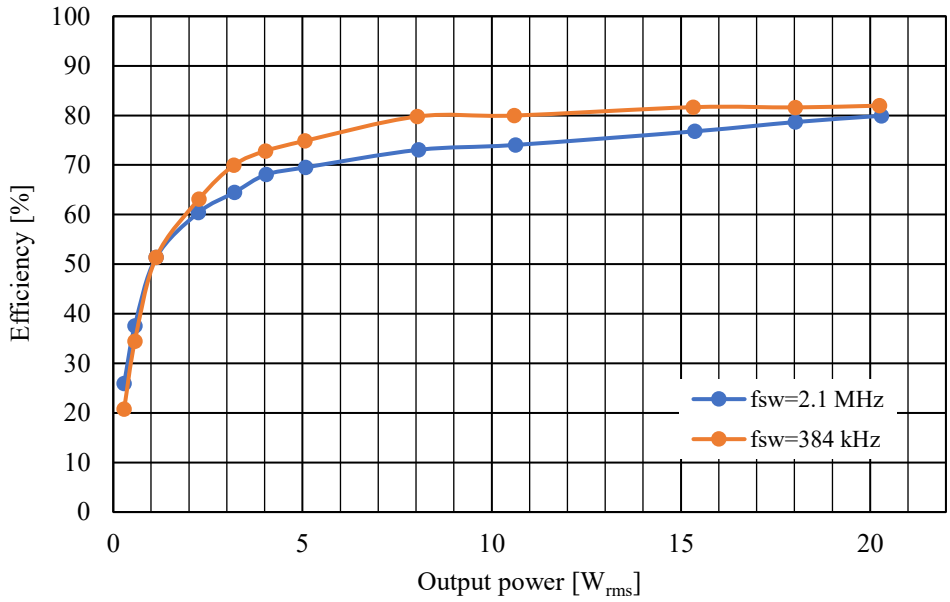


Figure 60. Efficiency of the amplifier board as a function of output power at two switching frequencies.

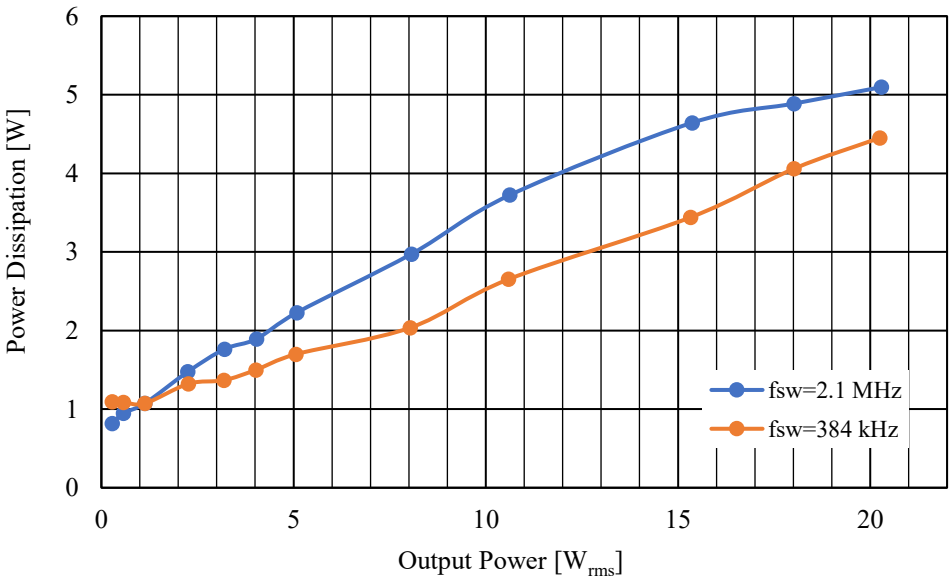


Figure 61. Power dissipation of the amplifier board as a function of output power at two switching frequencies.

The characterization of the AD242x amplifier was concluded with thermal measurements, by employing the thermal camera A320 by Flir, which has a resolution of 320 x 240 pixel and a refresh rate of 30 Hz. Two measures at different output power (2.5 W and 5 W) were carried out. The test signal was white noise high pass filtered at 200 Hz and the switching frequency of the amplifier was 2.1 MHz. Figure 62 shows the images captured with the thermal camera once the thermal steady-state was reached. All the images have temperature markers on key points, namely on the amplifier, on the loudspeaker, and on the environment (*Sp1*, *Sp2*, and *Sp3* respectively).

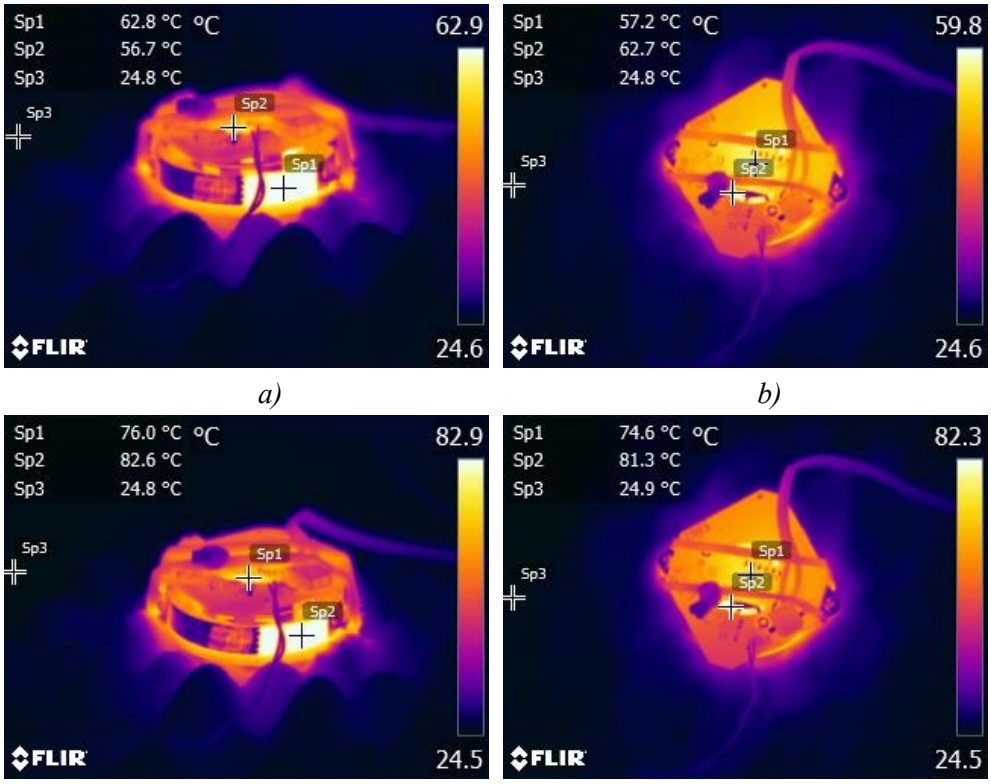


Figure 62. Thermal images of the amplifier and the loudspeaker. a) and b) are the top and bottom view for the test with 2.5 W of output power. c) and d) are the top and bottom view for the test with 5 W of output power.

AD243x amplifier

Since the amplifier section of the AD243x solution is identical to that of the AD242x solution, the THD+N and noise measurements were not repeated. On the contrary, as this solution offers the possibility to select between two power supply voltage levels, i.e., 8 V and 24 V, the efficiency at these two voltage levels was investigated. The same measurement setup shown in Figure 57 was adopted, with the exception that the A^{2}B main node was replaced by the EVAL-AD2435WA3LZ, and the DUT was the AD243x amplifier. The same test signal was employed, namely a 1 kHz sinusoidal tone. The output voltage of the amplifier was measured across the resistive load by the Audio Precision, while the input voltage and current were read on the power supply of the A^{2}B main node.

The efficiency was calculated as:

$$\eta = \frac{P_{out}}{P_{in} - P_{idle}} 100 \tag{3.4}$$

where P_{out} is calculated as in (3.1), P_{in} is the supply voltage of the A^{2}B main node, namely 12 V, multiplied by the current absorbed from the power supply, and P_{idle} is the power consumption of the A^{2}B main node and the amplifier without the input signal.

The efficiency calculated in this way is not related only to the amplifier board, but to the whole A^{2}B bus, namely the main node and the amplifier. However, this parameter is still useful for a comparison of the two power supply voltages since the power consumption of the A^{2}B main node remains constant. The efficiency was measured in three conditions:

1. PVDD 8 V, full-scale of the amplifier 7.5 V
2. PVDD 24 V, full-scale of the amplifier 7.5 V
3. PVDD 24 V, full-scale of the amplifier 21 V

The full-scale value of the amplifier can be configured through I^{2}C, and it represents the peak output voltage level. The results are shown in Figure 63. As it can be seen, when the power supply voltage is 8 V the efficiency is higher of about 5 to 10 % (in the output power range of interest), while as expected the full-scale value does not have any impact. Since the amplifier can provide 5 W of output power with a power supply of 8 V and the efficiency is higher in this condition, it was chosen to maintain this power supply level.

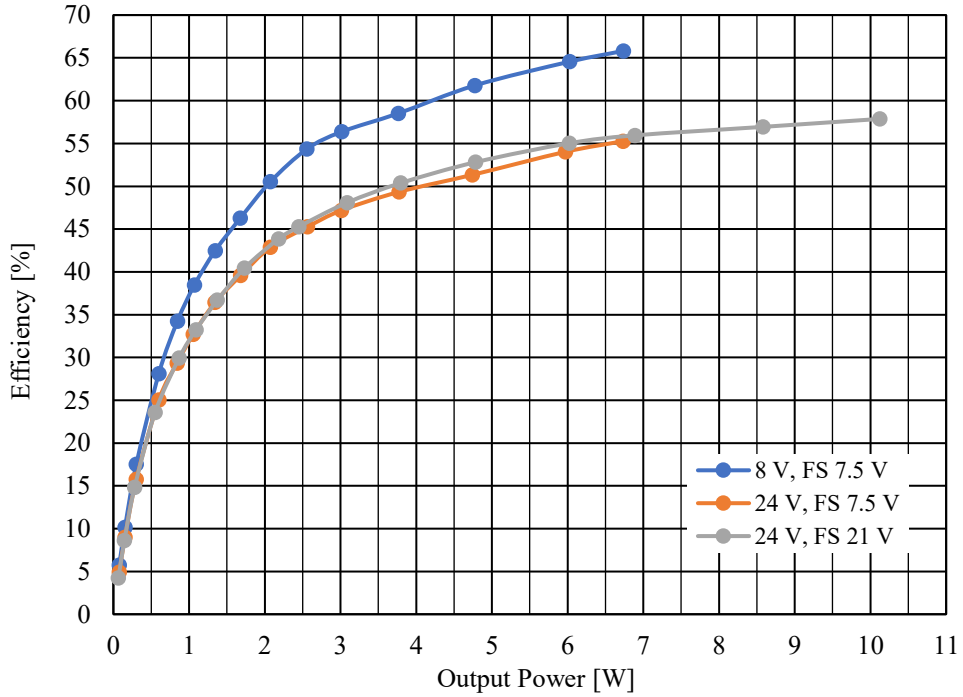


Figure 63. Efficiency of the amplifier board as a function of the output power at different voltage supply levels and full-scale values.

Final considerations

This section presented the ILZ application as a use case of the proposed architecture. Since this application demands to drive a high number of loudspeakers, a typical signal distribution architecture presents several limitations due to the massive wiring and cost. To overcome these limitations, it was designed a distributed amplifier that leverages the A^{2}B bus features for the distribution of data and power. In addition, the amplifier was designed to drive the *UltraFlat* loudspeaker developed by ASK Industries S.p.A.

Two amplifiers based on different A^{2}B transceiver families were designed and realized. The AD242x solution employs A^{2}B only for signal distribution, while power supply is provided externally. Instead, the AD243x solution allows distributing audio data together with power on the same cable. Hence, the wiring is lighter and cheaper than with the AD242x solution, although the A^{2}B main node may be more complicated. In fact, with the AD243x transceivers the maximum power over

the bus is 50 W and if 16 amplifiers are connected in daisy-chain, each one will have only 3.125 W available. If more power is required, fewer amplifier nodes must be connected on the same bus and so the number of main transceivers increases.

Experimental results proved that the audio performance of the amplifier is in line with the datasheet. The THD+N is lower than 0.02 % over the range of interest of output power (up to 5 W). In this range, the efficiency is not high (less than 70%), leading to a greater power dissipation and thus to a higher temperature. Audio measurements were not repeated for the AD243x board, since the amplifier section did not change, but the efficiency for different voltage supply levels was investigated. It was concluded that the highest efficiency is reached when the amplifier has a power supply voltage of 8 V.

3.2 Wave Field Synthesis Listening Room

The proposed architecture is suitable for professional audio applications too. During this thesis work, a Wave Field Synthesis (WFS) listening room was realized taking advantage of the benefits of the proposed A^2B based architecture. The realization of this system involved several researchers of the University of Parma. I mainly focused on the system architecture definition and in writing the firmware for the processing board, in addition to hardware and software debugging and testing. This work was published [63], and here it will be integrated with further details.

WFS is a spatial sound reproduction technique based on the reconstruction of a sound wave front by means of a large number of loudspeakers [64], [65]. Typically, loudspeakers are arranged linearly around the listening area, allowing to control the sound field only in two dimensions. One of main advantages of the WFS with respect to other spatial sound reproduction techniques is that the *sweet spot* is not limited to a small area, but it comprises the whole listening area, thus allowing several people to appreciate the spatialization effect. On the other hand, WFS systems are expensive. In fact, a considerable number of loudspeakers is involved and since they are driven individually, wirings and amplifiers contribute significantly to increase the system cost. Earlier systems employed inconvenient analog wirings [66], while latest exploit audio-over-IP technologies like Dante, Ravenna and AES67 [67], [68], or audio-over-ethernet, namely AVB [69]–[72]. In addition, there are still implementation based on AES10 [73]. Although, ethernet based solutions may simplify and

reduce the cost of the wirings, they require expensive electronic devices, such as ARM based processors or FPGA, to manage the protocols stack.

In the proposed WFS system these issues are overcome by employing A^2B to communicate with the front-end devices, namely the amplifiers and the loudspeakers. To improve the wiring of the system, A^2B active soundbars composed by 8 loudspeakers and an 8-channel amplifier were designed. This has the following advantages:

- The connections between the amplifier and the loudspeakers are minimized since the amplifier is in the soundbar.
- Each soundbar is a subordinate node on the A^2B bus, thus up to 4 soundbars (for a total of 32 channels) can be connected in daisy-chain with a simple UTP cable.

In addition, no FPGA or processor is required in the soundbars since the A^2B is completely managed by dedicated low-cost transceivers. However, A^2B cannot reach distances greater than 40 m, and it is not widespread in professional audio systems, so an interface board, called A^2B-DSP, was designed with the aim to convert A^2B to a common professional audio protocol and to collect signals from different A^2B busses.

In the next sections, the architecture, and the parts of the proposed WFS system will be explained in detail and the first system installation in a WFS listening room will be presented. As shown in Figure 64, the system architecture was first defined. Then, the A^2B-DSP, the amplifier and the soundbar were designed and realized. Next, the soundbar was characterized acoustically and thermally. Finally, the complete system was installed in a listening room.

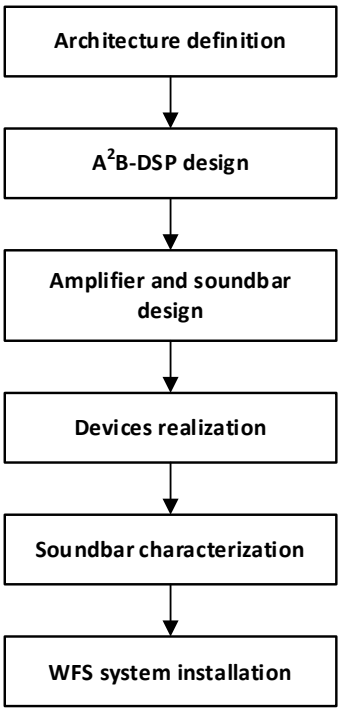


Figure 64. Flowchart of the development of the WFS system.

3.2.1 Architecture Design

The WFS system architecture is based on the architecture described in Section 2.1. The block diagram is shown in Figure 65. The system is composed by three main parts: a commercial audio system (e.g., a PC, a mixer, or an audio interface), a conversion and processing board, namely the A^2B-DSP in Figure 65, and several A^2B networks.

The key part of the whole architecture is the A^2B-DSP board. The principal task of the A^2B-DSP board is the format conversion between A^2B and one of the most used audio protocols, which should be chosen depending on the existing audio system and the number of channels required. Figure 66 and Figure 67 show the available connections of the A^2B-DSP board:

- A gigabit ethernet interface: in addition to DSPs programming and board control, it is employed for AVB communication.

- USB 2.0 which provides an USB-Audio Class 2.0 (UAC-2.0) compliant interface.
- A coaxial BNC and an optical transceiver to implement the AES10 connections.
- AES3/SPDIF input and output.
- Microphone input, with phantom.
- Line input and output.
- Headphones output.
- Expansion connector, which allows connecting external daughter boards to implement other protocols, such as Dante, Ravenna, and AES67.
- Two A²B busses: the A²B-DSP has two AD2428 transceivers that act as main nodes.

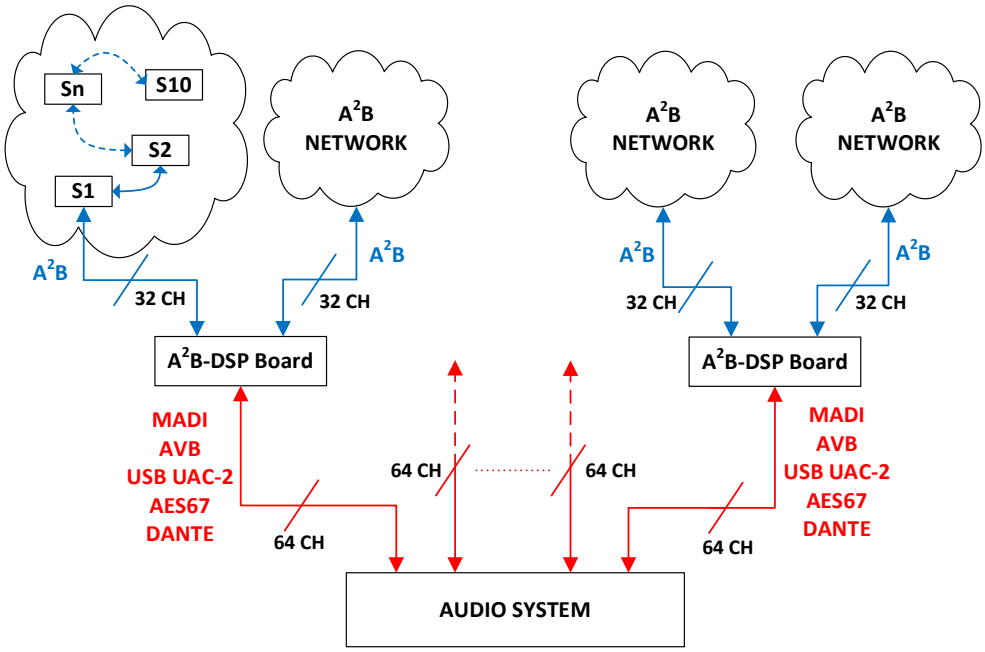


Figure 65. Block diagram of the A²B based WFS system architecture.

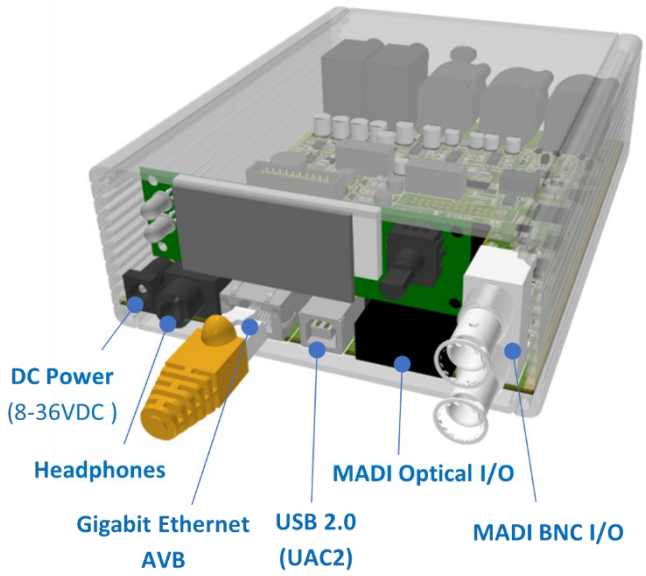


Figure 66. A²B-DSP connections, front view.

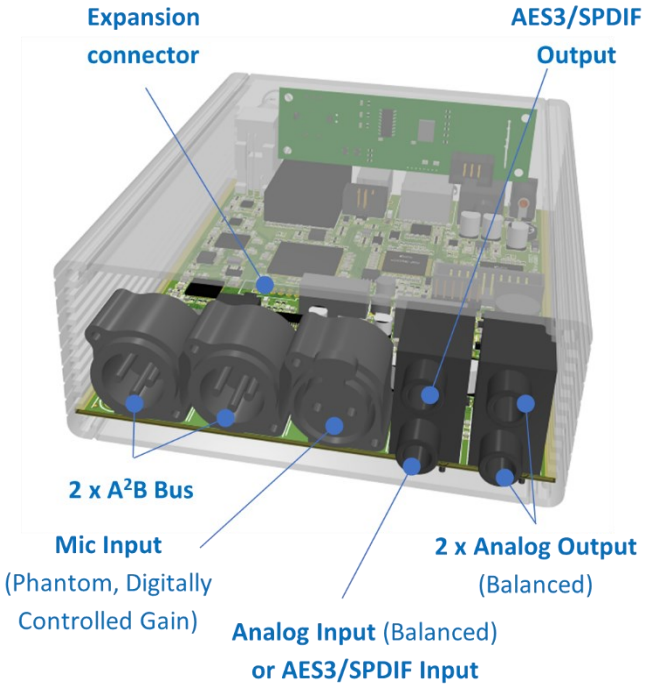


Figure 67. A²B-DSP connections, rear view.

At the A²B side can be connected a huge variety of devices such as amplifiers, processing units, ADCs, DACs, and microphones. These devices (represented in Figure 65 by blocks S1 to S10) must be equipped with an A²B transceiver acting as a subordinate node. Since the A²B-DSP offers the connections to two busses, the maximum number of the available A²B channels is 64.

The design of the A²B-DSP allows connecting several boards to the system, thus increasing the maximum number of channels, and offering system modularity. Furthermore, the A²B-DSP can be employed in different applications thanks to its great connectivity options and the different distance ranges of the supported protocols. Such applications include monitoring systems, where sensors are positioned at great distances from each other, or massive multichannel audio systems, where transducers are placed relatively closed to each other.

A top view of the board is shown in Figure 68, where the principal components are highlighted: XMOS host processors, audio codec, FPGA, DSPs and A²B transceivers. Figure 69 illustrates the internal routing of the board managed by the FPGA, which also implements the AES10 decoder.

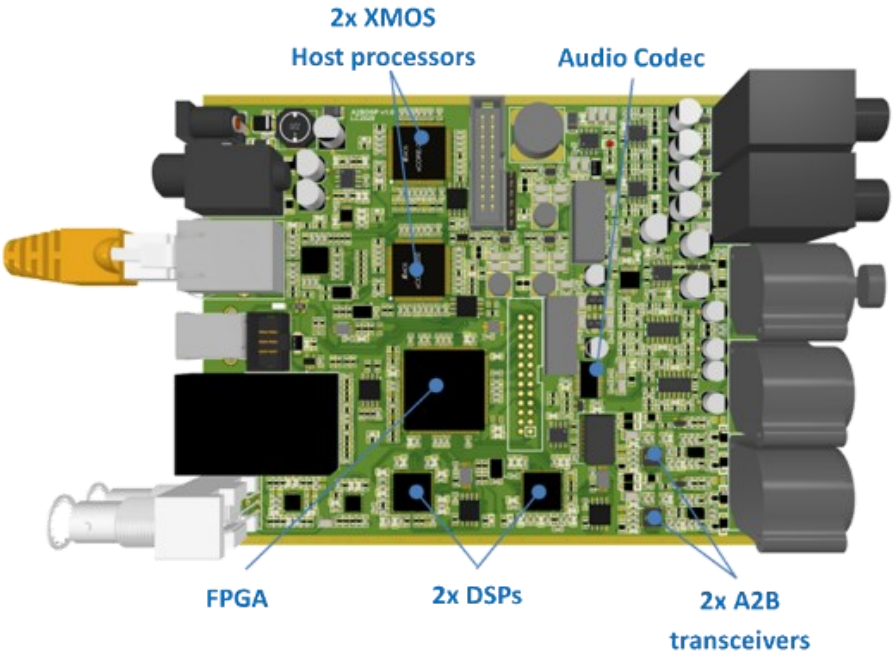


Figure 68. Main components of the A²B-DSP.

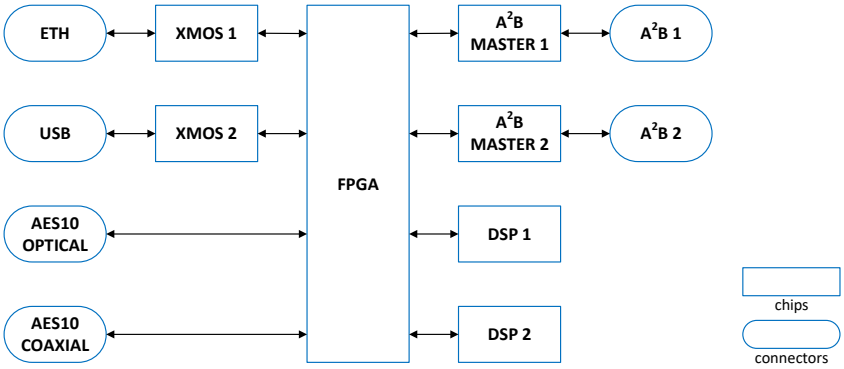


Figure 69. Main components connections.

The A²B-DSP can also execute signal processing algorithms thanks to two DSPs of the SigmaDSP family by Analog Devices. The peculiarity of these DSPs is that they can be programmed by a graphical development tool called SigmaStudio directly through the ethernet port of the A²B-DSP. Each DSP can compute up to 24000 taps at 48 kHz (512 taps per channel, with 64 channels), making them suitable for room correction, loudspeakers pre-equalization or to implement spatial audio algorithms. Finally, the DSPs offer some asynchronous resampling channels, which are particularly useful to solve clocking problems in complex audio networks.

Finally, Figure 70 shows a prototype of the A²B-DSP.

Figure 70. Realization of the A²B-DSP board (without top cover).

3.2.2 Active Soundbar

An A^2B active soundbar was designed to reduce the impact of cables on the overall system cost. The soundbar is composed by 8 loudspeakers and an 8-channel amplifier that acts as a subordinate node, thanks to the equipped A^2B transceiver. The realized prototype can be seen in Figure 71. The loudspeaker housing is made of birch wood, 12 mm thickness, with dimensions (LxWxH) 100x19x14 cm. Once assembled, the total weight is 10.5 kg. The inter-center distance *d* of the loudspeakers is 125 mm, thus the spatial aliasing frequency is about 2.7 kHz on-axis and about 1.8 kHz at 30°, as provided by:

$$f_{alias} = \frac{c}{d(1+\sin(\alpha))} \tag{3.5}$$

where *c* is the speed of sound in air and it is equal to 343 m/s, and α is the angle between the direction normal to the soundbar and the listener. Each loudspeaker is mounted in a closed, isolated volume of 1.1 l (Figure 72), calculated according to Thiele-Small parameters [74]–[77]. For minimizing the standing waves, the rear side is tilted by 10° and each volume is covered in fiberglass on the bottom, rear, and top sides. The model of loudspeaker is a 4.0" full range by RCF-group, having nominal impedance 8 Ω, power rating 40 W_{rms}, sensitivity 92 dB SPL (1 W, 1 m).

Figure 71. Prototype of the A^2B active soundbar.

Figure 72. A^2B active soundbar (front view). Isolated volumes are shown.

In Figure 73 (dashed line), it is possible to see the frequency response of the loudspeaker once mounted in the soundbar, measured in anechoic room at 1 m, 1 W with a calibrated Bruel&Kjaer (B&K) microphone (capsule type 4189, pre-amplifier type 2187). For being more representative, it is shown the measurement relative to the speaker mounted in the center of the soundbar (number 4). Thanks to the two DSPs of the A^{2}B-DSP board, it is possible to introduce an individualized filtering on each channel, employing Finite Impulse Response (FIR) filters. In Figure 73 (dotted line), one can see the frequency response of an inverse filter calculated with the regularized Kirkeby method [78]. Finally, the solid line of Figure 73 shows the equalized response of the loudspeaker, which results almost flat (±1 dB) in the frequency range 200 Hz – 11.2 kHz.

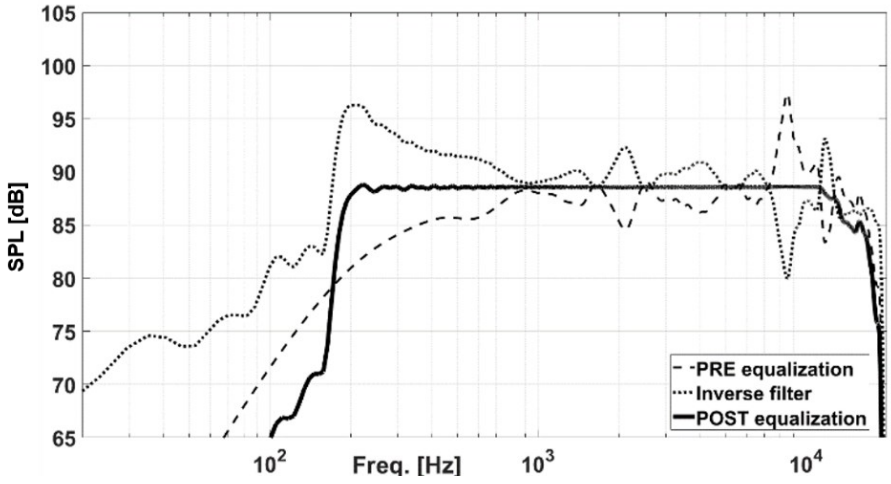


Figure 73. Frequency responses of a not-equalized loudspeaker mounted on the soundbar (dashed line), a FIR equalization filter (dotted line) and the equalized loudspeaker (solid line).

On the rear side, an aluminum frame (Figure 74) houses the electronics (Figure 75). From left to right it can be seen the A^{2}B and power supply connectors for the next soundbar, the custom A^{2}B 8-channel power amplifier, the power supply, and the A^{2}B and power supply connectors from the previous soundbar. The power supply is a commercial AC/DC switching, rated power 300 W, output voltage 36 V. As it can be seen in Figure 75, the amplifier outputs are connected to the loudspeakers through holes which were then sealed with silicone. The aluminum frame is attached to the rear of the soundbar with screws.

On each side there are two input and two output connectors, one for the A²B signal and one for the power supply, which are both delivered in daisy-chain. The power supply connectors are *powercon*, a standard choice in professional audio applications. The A²B bus travels between the soundbars over AES3 (also known as AES/EBU) digital cables, whose physical specifications are defined in the standard IEC 60958 type 1: shielded twisted pair, 110 Ω, XLR connectors.

Figure 74. Electronics housing, rear view.

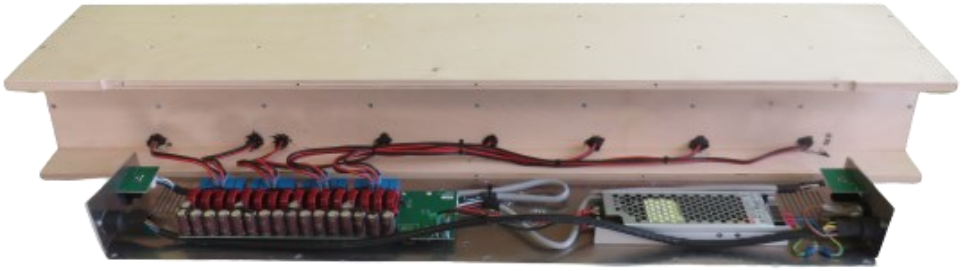


Figure 75. Electronics housing, top view.

The 8-channel A²B amplifier can be seen in Figure 76 (top and bottom views). The board is equipped with 4 class-D amplifiers by Texas Instruments, the TAS3251. It is a 2 channels amplifier, with 75 W/ch output power on 8 Ω load, when operating at 36 V of power supply. Moreover, each class-D amplifier integrates a DSP which provides equalization, crossover, limiter, dynamic range control and volume control for each channel. As it can be seen on the top of the PCB there are the inductors and capacitors of the low-pass output filter of the class-D amplifiers, while all the SMD components are on the bottom.

The A²B soundbar was designed for passive cooling, thanks to the rear aluminum frame that acts as a heatsink. Passive cooling is of great importance in audio systems, especially when several devices are in the same room, as in the WFS systems. However, the amplifier also features a fan control circuit for active cooling for safety reasons. The amplifier cooling system was tested in laboratory during four

hours of continuous operation at 100 W_{rms}. The amplifier temperature was measured on the heatsink, as near as possible to the exposed pad of the amplifier integrated circuit. The result is shown in Figure 77, where it can be seen that the maximum temperature reached is 62 °C (36 °C above the ambient temperature of 26 °C). Once reached the steady state, the fan was turned on to evaluate the effectiveness of an active cooling. This resulted in a temperature drop to 42 °C (a reduction of 20 °C) over the next 1.5 hours. A thermal image of the A^2B amplifier is shown in Figure 78. The hottest points are near the power amplifier integrated circuit (Sp1 and Sp2 in Figure 78).

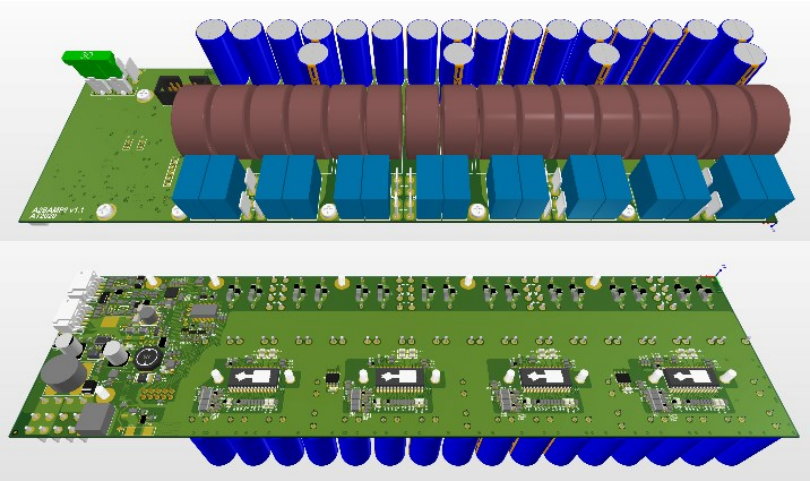


Figure 76. 3D model of the 8-channel A^2B amplifier.

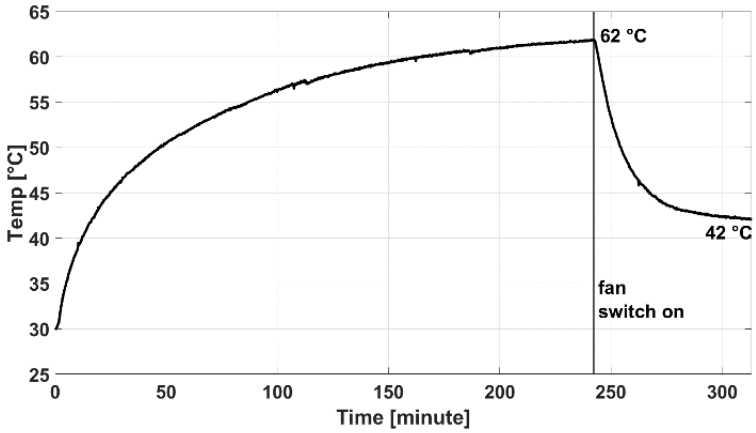


Figure 77. Amplifier temperature as a function of time at 100 W_{rms}.

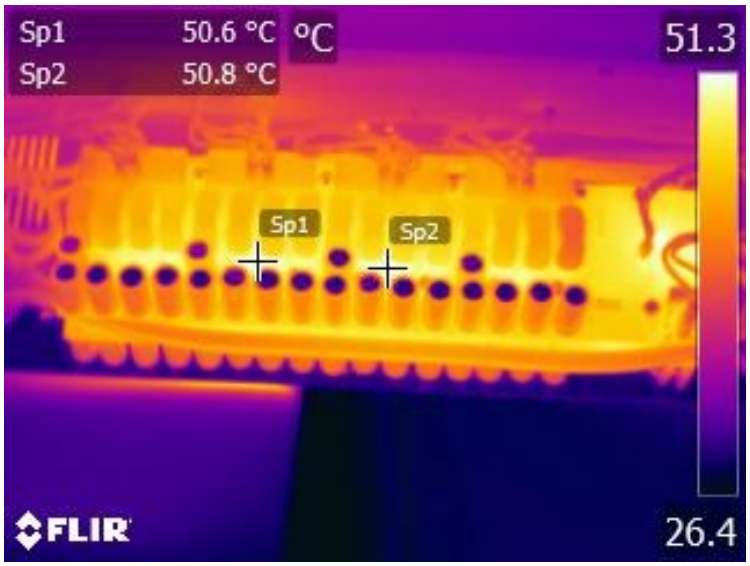


Figure 78. Thermal image of the A^2B amplifier during tests, top view.

The soundbar was then measured in an anechoic room. Firstly, the Sound Pressure Level (SPL) at 1 m was measured with the previously described microphone. White noise was played through the 8 loudspeakers, at the maximum power, and the recorded SPL was 108 dB. Secondly, the soundbar directivity was measured on the horizontal plane by means of a single-axis turntable. The experimental setup is shown in Figure 79. The B&K microphone was positioned at 1 m distance, while the soundbar was positioned so that the loudspeaker number 4 was aligned with the rotation axis. In fact, the two central loudspeakers are more representative, not being affected by border effects. An Exponential Sine Sweep (ESS) [79] of 10 s covering the range 22 Hz – 22 kHz was played through the loudspeaker for each direction. The measurement was repeated 72 times, with an angular resolution of 5°, starting from 0° to 355°.

The deconvolution of the recorded sweeps with the inverse sweep provided the Impulse Responses (IRs). The polar patterns (Figure 80 and Figure 81) were computed by averaging the directivity in octave bands. Since the DSP's crossover on the loudspeakers was set at 100 Hz, the first two octave bands centered at 31.5 Hz and 63 Hz were not computed. One can note that the SPL are well matched at all octaves, except for the one centered at 125 Hz, as already seen in Figure 73.

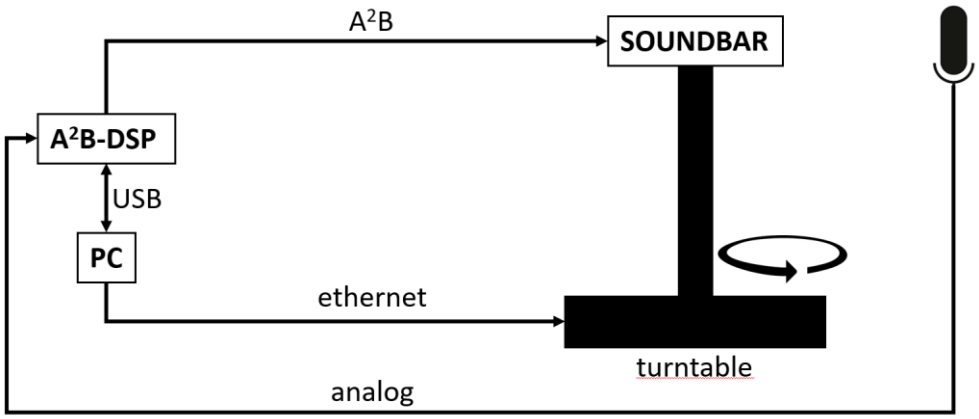


Figure 79. Experimental setup for the measurement of the soundbar directivity.

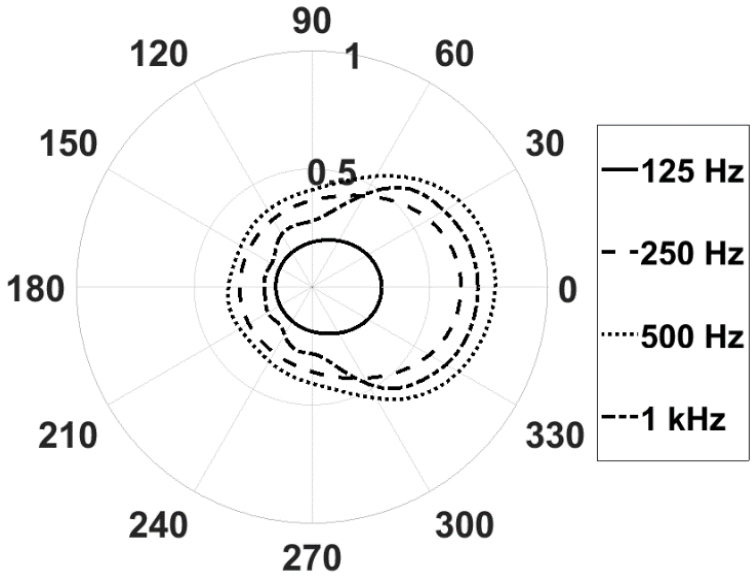


Figure 80. Polar pattern of the loudspeaker 4, octave bands 125 Hz, 250 Hz, 500 Hz, and 1 kHz.

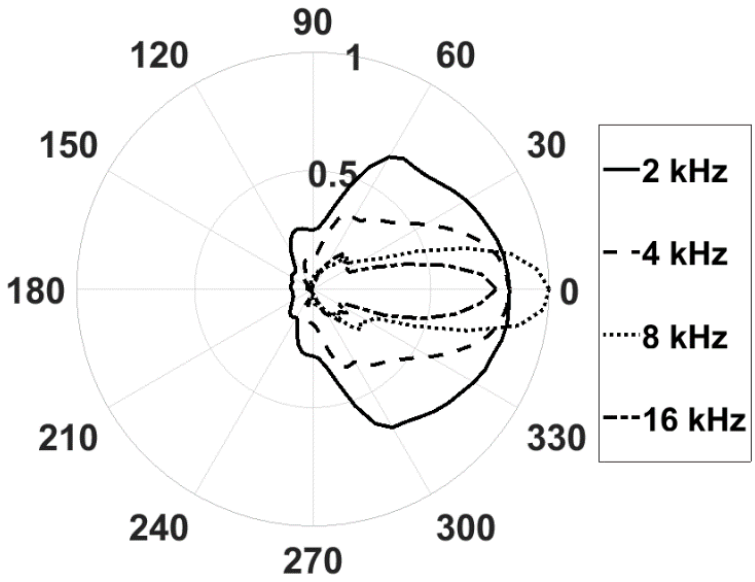


Figure 81. Polar pattern of the loudspeaker 4, octave bands 2 kHz, 4 kHz, 8 kHz, and 16 kHz.

3.2.3 WFS Installation

This section describes the installation of the system in a listening room (Figure 82). The room dimensions allowed the installation of a system with 20 soundbars (160 channels) with dimensions of 5.5 m x 5.5 m. The total number of seats for this configuration is 28. The block diagram of the system is shown in Figure 83. As a project requirement, audio had to be played from a PC, thus an audio interface with MADI capability was chosen. The model is the RME Madiface XT which offers 192 output channels on three different MADI connections (two optical and one coaxial). Each MADI line (64 channels) is connected to an A^2B-DSP, which converts data to two A^2B busses, 32 channels each. Up to 4 active soundbars can be connected on a single A^2B bus and as it can be seen in Figure 83, the power supply is provided by connecting the soundbars in daisy-chain. In addition, the A^2B-DSPs offer two line outputs to drive subwoofers. These additional channels are obtained in the DSPs by summing all the signals of a side of the room (with a low-pass filter at 100 Hz). Finally, Figure 84 and Figure 85 show the installation of the WFS system.

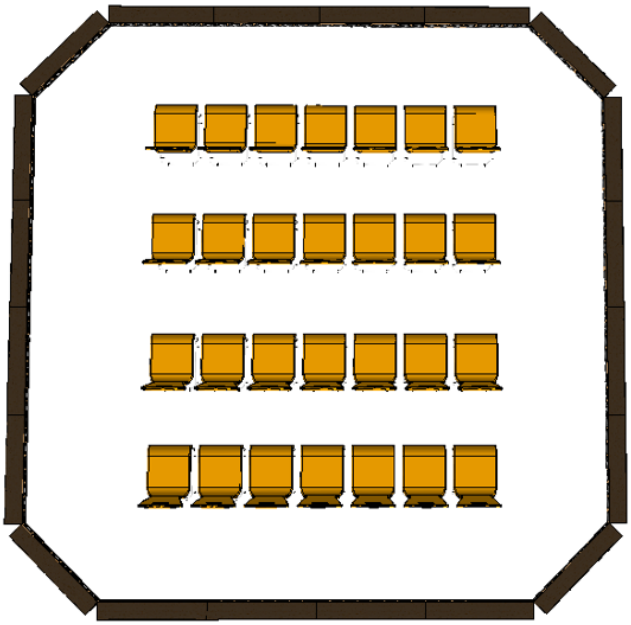


Figure 82. WFS system with 20 soundbars (160 channels) and 28 seats.

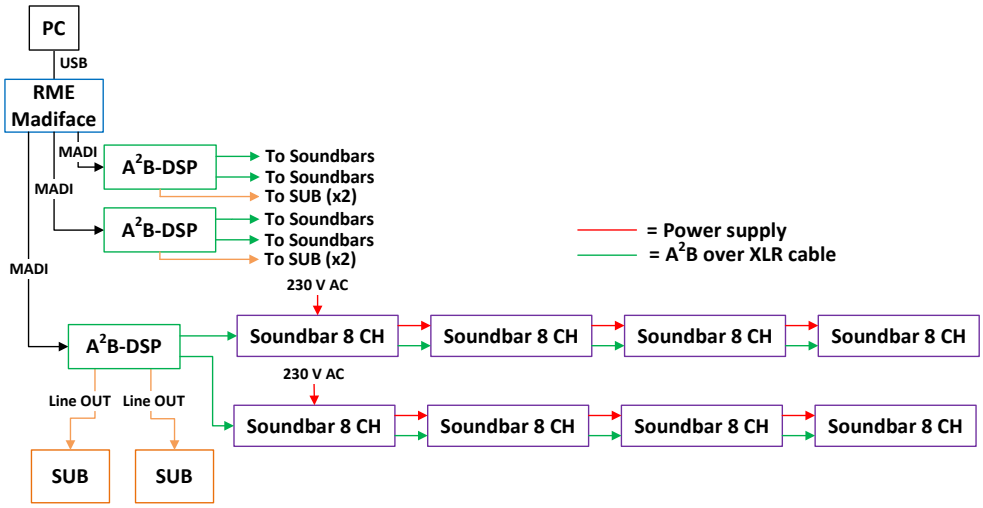


Figure 83. Block diagram of the WFS system installed.

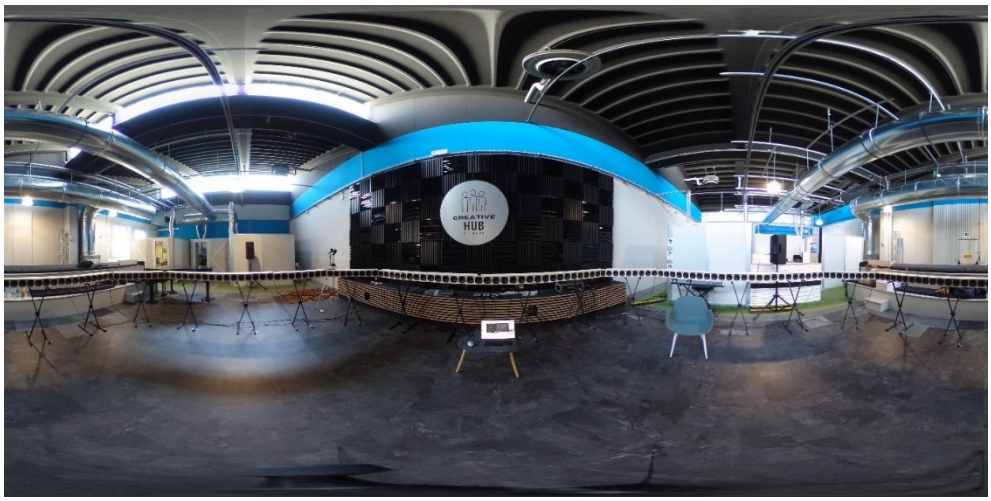


Figure 84. 360° image of the WFS system installation.

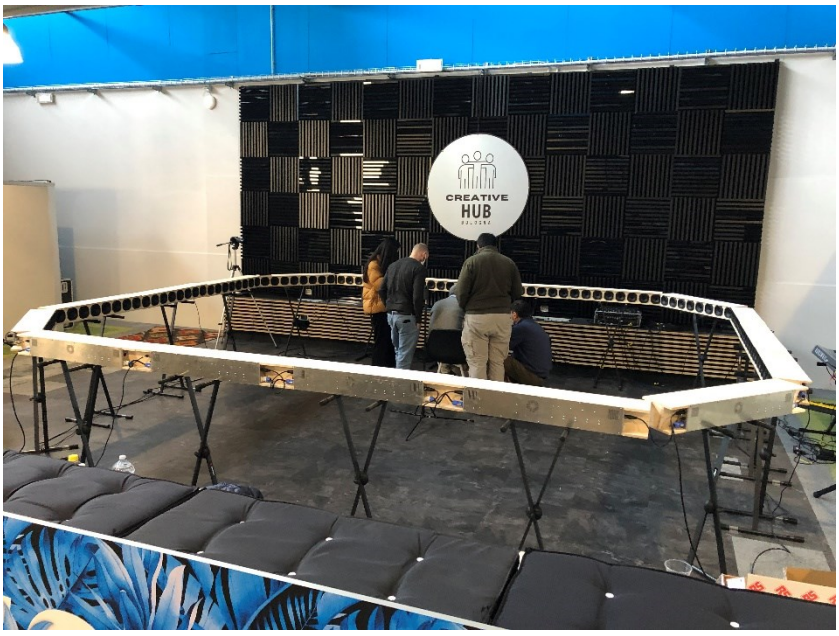


Figure 85. WFS system installation.

3.3 Microphone Arrays

Microphone arrays have become popular in various applications, such as room or environmental measurements, speech recognition, hearing aids, ANC systems, teleconferencing and in-car communication. When designing a microphone array, the first thing to consider is the application. For example, whether the array is employed for environmental measurements or dereverberation and noise reduction in teleconferencing systems dramatically changes the design of the array. The geometry of the array, the number of capsules and their layout typically affect spatial accuracy and the working frequency range, while the type of capsules and data transmission protocol influence audio quality.

In general, spatial accuracy is increased by increasing the number of microphones and by decreasing their distance. Hence, microphone arrays are being designed with an increasing number of capsules [80]–[82]. However, spatial sampling is not only affected by the number of microphones, but the geometry of the array and the microphones arrangement greatly determine the performance. For example, robustness and stability of beamforming filters change according to these parameters.

MEMS microphones are one of the most common technologies employed to build arrays. They can have an analog or a digital output. While electret microphones typically provide higher-quality signals, digital MEMS microphones are more and more preferred, especially in consumer electronics and automotive, thanks to their cost-effectiveness and robustness to electrical noise. Analog capsules may present noise immunity problems, especially with long cables connecting the microphone to the ADC. In addition, analog capsules require pre-amplifiers and ADCs which contribute to increase the system cost and the design complexity. On the other hand, digital MEMS microphones typically offer lower performance in terms of SNR and dynamic range.

MEMS microphones are already employed to build microphone arrays [83]–[85]. However, digital MEMS microphones typically support I^2S or PDM output, which do not allow to have connections longer than a few centimeters [86], [87]. A^2B is a suitable solution to acquire signals from microphone arrays that can be further or even distributed in space. In fact, an A^2B transceiver can acquire up to 4 PDM or I^2S microphones (or more if output is TDM) and transmits to other nodes which can be distributed to cover large areas (remember that the maximum distance between two nodes is 15 m and the global length of an A^2B bus is 40 m). Other

advantages of employing A²B for microphone arrays are the low latency, synchronization, and modularity offered by this technology. A²B introduces just two samples of latency, thus leaving time for processing algorithms (e.g., ANC or echo-cancellation), which require a latency as low as possible to work (usually less than 1 ms). Synchronization is another critical aspect, for example when beamforming techniques are employed. The clock propagation was studied in an A²B bus (see Section 2.2) showing good performance, and in the next section the clock propagation model previously obtained will be used to confirm that jitter and latency introduced by A²B do not affect beamforming performance in terms of directivity. Finally, the modularity of the A²B architecture can be leveraged to build *meta-arrays* (array of arrays) by combining several nodes into a single physical device or by distributing them to cover large areas.

For these reasons, an A²B modular microphone array (triangular array) was designed and realized [88], thanks to a partnership with ASK Industries S.p.A. The array has 4 digital MEMS microphones, and it can be used as a single device or combined with other units to create planar or 3D meta-arrays. As shown in Figure 86, the array was designed starting with the definition of the operating frequency range. Then, some geometries with different number of capsules were compared by simulating their directivity.

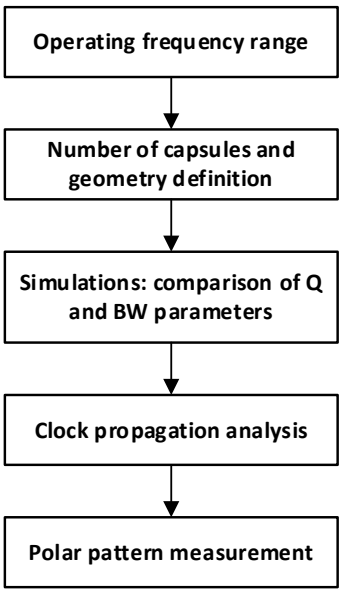


Figure 86. Flowchart of the design process of the triangular microphone array.

Once a design was chosen, a clock propagation analysis was carried out to assess that A^2B clock propagation does not degrade the microphone array performance. Finally, the array was measured to evaluate the beamforming performance.

The next sections will describe in detail the design process of the triangular array and they will present the meta-arrays that were realized, as summarized in Figure 87.

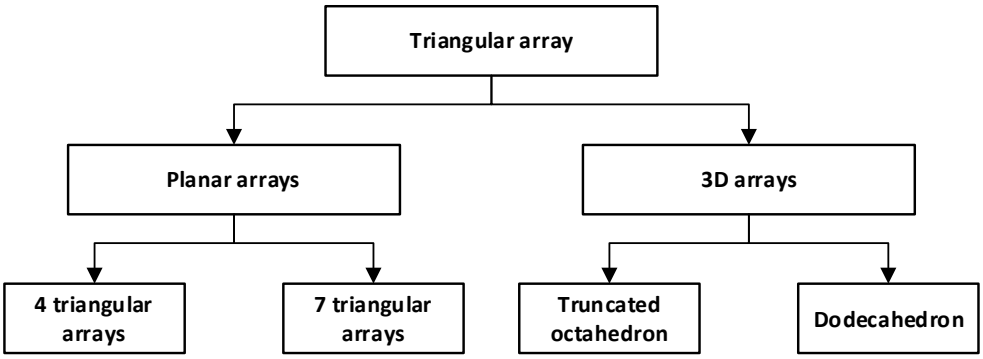


Figure 87. Meta-arrays realized with the triangular microphone array.

3.3.1 Triangular Array

As described in the previous section, A^2B offers several characteristics which can be leveraged to develop microphone arrays. In addition, A^2B easily allows creating modular arrays, just by adding new nodes to the bus. To keep the array a full-digital solution and cost-effective, MEMS digital microphones were chosen.

In literature, there exists many microphone arrays geometries, which can be spherical and cylindrical [89]–[94] or planar [95]–[98], with circular, radial, or random capsules arrangement. To develop the triangular array, numerical simulations were carried out to identify the best geometry and the capsules arrangement with the aim of optimizing the beamforming accuracy and the number of microphones. Planar regular geometries like triangular, square, pentagonal, and hexagonal were preferred as they can be easily employed to create meta-arrays by attaching them in daisy-chain on the A^2B bus. Capsules, ranging from 4 to 7, were positioned on the vertices and as suggested in [99], one was kept in the center to improve the spatial accuracy of the beamforming.

Before going into details of the design of the array, basic beamforming theory and the definition of the parameters employed to evaluate the array performance will be provided. Beamforming consists in combining raw signals captured by the

capsules, to create arbitrary directive beams, called virtual microphones. Here, such operation is performed with a linear processing, which makes use of a matrix of FIR filters, computed with the regularized Kirkeby [100] inversion:

$$H_{m,v,k} = [C^*_{m,d,k} \cdot C_{m,d,k} + \beta_k \cdot I_{m,m}]^{-1} \cdot [C^*_{m,d,k} \cdot A_{d,v} \cdot e^{-j\pi k}] \tag{3.6}$$

where m = [1,…,M] is the capsule index, v = [1,…,V] is the virtual microphone index, k is the frequency index, d = [1,…,D] is the index of the Directions-of-Arrival (DoA) of the sound wave; the matrix C is the complex response of each capsule m for each direction d, the matrix A defines the frequency independent amplitude of the target directivity patterns, $e^{-j\pi k}$ introduces a latency that ensures filters causality, the dot (·) is the scalar product, I is the identity matrix, []* denotes the conjugate transpose, $[]^{-1}$ denotes the pseudo-inverse, β is a frequency-dependent regularization parameter [101], and it represents the most significant improvement of the Kirkeby method over the traditional Tikhonov regularization, where instead β is constant. The grid of DoA employed for simulating and measuring the array responses is a spherical t-design geometry [102], [103], of order t = 21, consisting in a total of D = 240 directions uniformly distributed over a unit-radius sphere. To assess the array performance, a single virtual microphone (V = 1) was encoded with a target directivity A of a fourth order cardioid defined as follow:

$$A(\vartheta) = [0.5 + 0.5\cos(\vartheta)]^4 \tag{3.7}$$

The virtual microphone is obtained by multiplying, in the frequency domain, the response of the array $C_{m,d,k}$ with the beamforming filter $H_{m,v,k}$, as follow:

$$V_{d,k} = C_{m,d,k} \cdot H_{m,v,k} \tag{3.8}$$

The two-dimensional and three-dimensional polar pattern of the virtual microphone are shown in Figure 88.

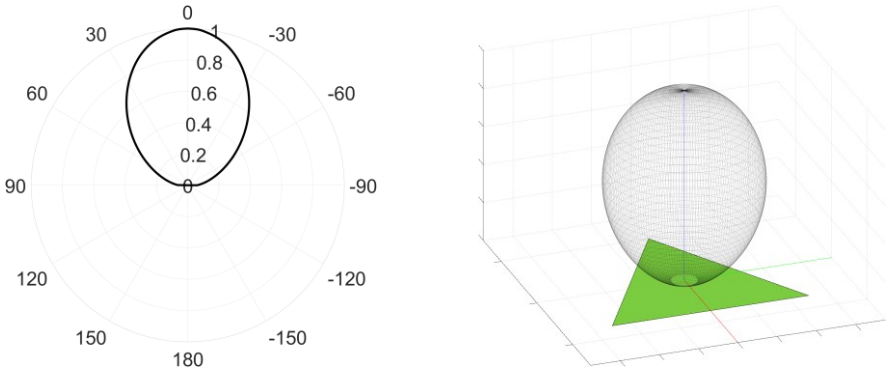


Figure 88. Two-dimensional (left) and three-dimensional (right) ideal polar pattern of a fourth order cardioid.

The directivity of the array was then evaluated by means of two parameters: the directivity factor Q, and the half-power beamwidth BW [104]. The directivity factor Q is defined as:

$$Q_{v,k} = \frac{I_{v,k}^{\max}}{I_{v,k}^{0}} \tag{3.9}$$

where I^{max} is the magnitude of the sound intensity vector in the direction of maximum emission, I^{0} is the average of the magnitude of sound intensity over the whole sphere, v is the virtual microphone, and k is the frequency. The parameter BW is equal to twice the angle of the beam between the direction of maximum sensitivity and the direction at -3 dB below the maximum, hence:

$$BW_{v,k} = 2\angle(\vec{S}_{v,k}^{\max}, \vec{S}_{v,k}^{\max-3dB}) \tag{3.10}$$

where $\vec{S}_{v,k}^{\max}$ is the direction of maximum directivity, $\vec{S}_{v,k}^{\max-3dB}$ is the direction where directivity is reduced by 3 dB respect to the maximum, $\angle$ denotes the angle, v is the virtual microphone, and k is the frequency.

Array Design and Simulation

In the design of this microphone array the first parameter that was involved is the distance between capsules. This parameter is crucial to set the operating frequency range of the array.

In fact, beamforming is physically constrained by the ratio between the distances of the capsules and the wavelength, as:

$$s = \frac{1}{4}\frac{c}{f} \tag{3.11}$$

where c = 343 m/s is the speed of sound and f is frequency. The array was designed to work in the vocal band, i.e., 300 Hz - 3.4 kHz in telecommunications [1], [105], thus the minimum distance s was chosen according to it. By substituting f_{max} = 3.4 kHz in (3.11), the minimum distance of the capsules is $s_{min} \approx 25$ mm. The low frequency limit derives from the maximum distance of the capsules s_{max}. Meta-array geometries allow to increase s_{max}, thus improving the low frequency performance without affecting the high frequency performance.

Once s_{min} was set, the geometry and the number of capsules required were investigated by means of numerical simulations. Four regular polygons were considered, namely a triangle, a square, a pentagon, and a hexagon, as shown in Figure 89. One capsule was positioned in the centre of the polygon, while the others were arranged in the vertexes, along a circle of radius $r = s_{min} = 25$ mm.

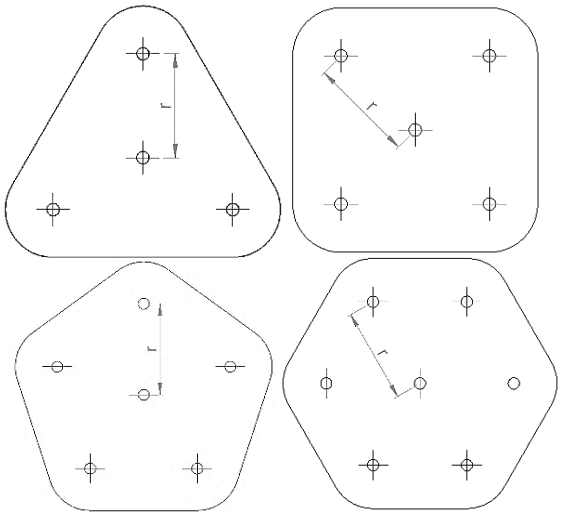


Figure 89. Drawings of the four geometries considered. The minimum distance between the capsules is indicated by r.

The geometries were then simulated with Comsol with the Finite Elements Method (FEM), considering the near field effect: the system is stimulated by a point

source radiating spherical waves of 1 Pa at 1 m distance. The material of the domain is air, while the arrays were modeled as a rigid body. The simulations were calculated for each direction d of the previously described grid ($D = 240$). A three-dimensional modelling was employed, thus discretized with a tetrahedral mesh, featuring six elements per wavelength. The simulations were solved within the frequency range 300 Hz - 3.4 kHz, with a frequency resolution of 5 Hz. For each direction d, the solution is evaluated at the M points corresponding to the position of capsules, considering an ideal frequency response. This provided the matrix $C_{m,d,k}$ required to solve (3.6). The solutions were processed by combining (3.7) in (3.6), for values of M ranging between 4 and 7. Then, the frequency dependent parameters Q and BW were evaluated for each of the four different cases. Results are shown in Figure 90 and Figure 91.

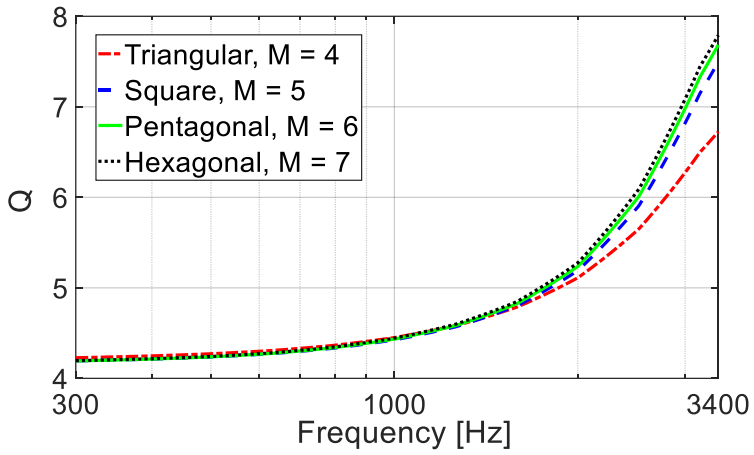


Figure 90. Directivity factor Q for triangular (red), square (blue), pentagonal (green), and hexagonal (black) arrays, with number of capsules ranging between M = 4 and M = 7. Higher Q is better.

The values of Q are in the range 4.2 - 7.8, while the values of BW are comprised between 105° and 69°. As it can be observed by considering BW and Q as functions of M, BW has slight variations over the considered frequency range, while Q only varies significantly for frequencies higher than 2 kHz. At 3.4 kHz, Q increases from 6.7 to 7.8, that is an increment of 16.4%, while the number of capsules increases from 4 to 7, namely an increment of 75%. More clearly, the ratio Q/M was calculated by varying M. This parameter is the directivity factor normalized on the number of

capsules, and it represents the effectiveness of a single capsule. Results are shown in Table XIX, and it can be seen that *Q/M* is maximum when *M* = 4.

Finally, Figure 92 shows the comparison of the KPI of the four proposed geometries. The KPI considered are operating frequency range, cost effectiveness, *Q*, *BW*, and *Q/M*. The operating frequency range is equal for all the geometries, as the minimum and maximum distance between the capsules is the same. Cost effectiveness is better for the microphone with 4 capsules, as the A^2B transceiver can acquire up to 4 PDM signals, while the other arrays require additional devices. As previously discussed, *Q* and *BW* slightly increase as the number of capsules increase, and *Q/M* is better for the array with 4 capsules.

Therefore, the triangular array with four capsules allows maximizing the directivity and minimizing the half-power beamwidth with the minimum number of capsules and the better cost effectiveness.

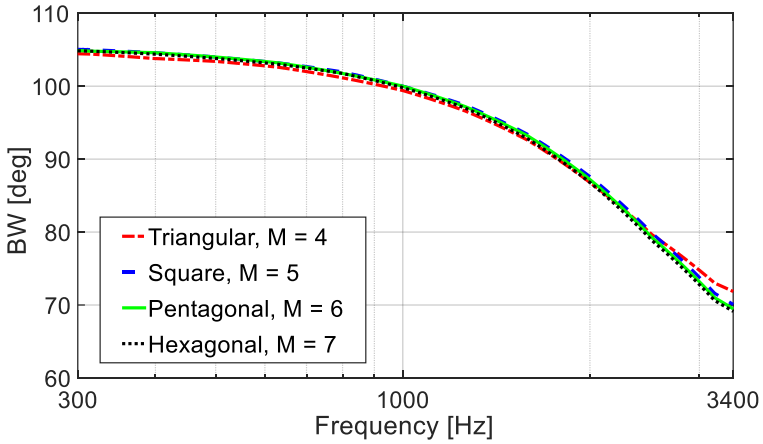


Figure 91. Half-power beamwidth BW for triangular (red), square (blue), pentagonal (green), and hexagonal (black) arrays, with number of capsules ranging between M = 4 and M = 7. Lower BW is better.

Table XIX - Ratio Q/M at 3.4 kHz for M = 4, 5, 6, 7

M	*Q/M*
4	1.68
5	1.5
6	1.28
7	1.11

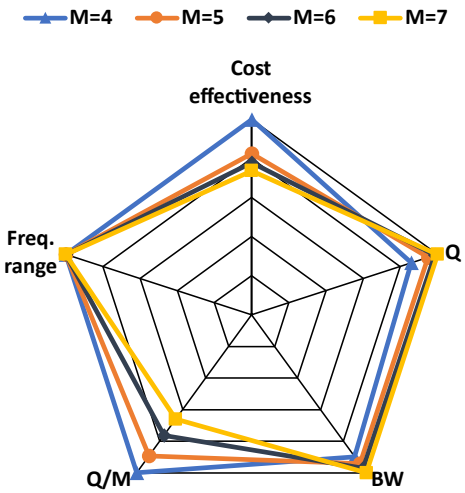


Figure 92. Comparison of the KPI of the four geometries.

A²B Clock Propagation Model

To prove that the clock reconstruction mechanism of A²B did not affect the beamforming performance, the effects of jitter and latency were investigated. The quantities of jitter and latency of each A²B node were measured as explained in Section 2.2, and the effects on beamforming were evaluated by means of an equivalent additive noise [36], [106], [107]. The clock propagation effects were introduced in the post-processing of the numerical simulations by means of fractional delays applied in the frequency domain. This allowed to overcome the limited resolution of the discrete time domain, namely ±1 sample (e.g., 20.83 μs at 48 kHz).

Fractional delays were introduced as:

$$C'_{m,d,k} = C_{m,d,k} \cdot e^{-j\omega t_{delay}} \tag{3.12}$$

where C is the numerical response of the array and $e^{-j\omega t_{delay}}$ accounts for the clock skew contribution. Finally, t_{delay} [s] is:

$$t_{delay} = \mathcal{N}(\mu, \sigma^2, d, m, k) \tag{3.13}$$

where $\mathcal{N}$ is the standard normal distribution, μ is the mean value of $\mathcal{N}$, σ^2 is the variance of $\mathcal{N}$, $d = [1,\ldots,D]$ is the DoA index of the sound waves, $m = [1,\ldots,M]$ is the capsule index, k is the frequency index. The employed values of μ and σ, obtained with measurements of Section 2.2, are summarized in Table XX and they are

represented in Figure 93. This condition represents the worst-case scenario, where each capsule receives a different clock, as it happens in different subordinate nodes. In fact, each A^{2}B transceiver reconstructs a clock signal that is fed to its four microphones. However, it was chosen to model the clock propagation effects in this way because such condition represents what happens when building meta-arrays, where several A^{2}B transceivers are employed.

Table XX - Jitter and latency values for each capsule of the triangular array

Node	Mic	μ [ns]	σ [ns]
Sub 1	1	0	1.6
Sub 2	2	+10	1.8
Sub 3	3	-10	2.0
Sub 4	4	+5	2.1

The central capsule (microphone 1) is assumed as relative reference of the time scale, and it has $\mu = 0$. The latency of the other capsules is modelled with the worst possible combination. The standard deviation values reflect the tendency of the jitter to increase at each reconstruction of the clock on subsequent subordinate nodes. Numerical simulations processed with jitter and latency showed no difference with respect to ideal ones, hence proving that A^{2}B does not affect beamforming directivity even when combining multiple transceivers, which is the case of meta-arrays.

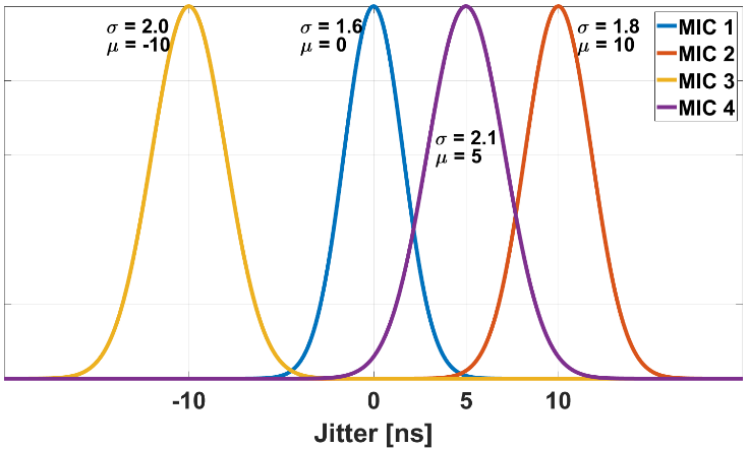


Figure 93. Clock propagation effects applied to four capsules connected to four different A^{2}B transceivers.

Experimental results

The triangular array was built and measured in a semi-anechoic chamber with a two-axis turntable. The protype of the array is shown in Figure 94. The capsules and the other electronic components are placed on the bottom side of the PCB, so that the top side is flat and smooth to minimize diffractions. The specifications of the employed digital MEMS capsule [108] are shown in Table XXI.

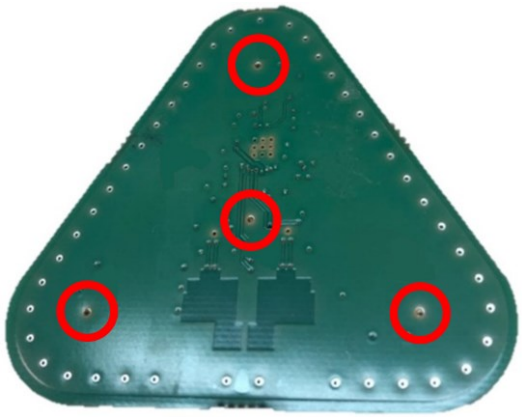


Figure 94. Prototype of the A²B triangular array. Red circles highlight the capsules.

Table XXI - MEMS microphone specifications

Parameter	Value	
AOP	130	[dB SPL]
SNR	69	[dB(A)]
Dynamic range	105	[dB]
Operating voltage range	1.62 - 3.6	[V]

The triangular array is equipped with one A²B transceiver, operating as a bus-powered subordinate node, which acquires four MEMS microphones. A block diagram of the array is shown in Figure 95. Thanks to A²B features it is possible to connect up to 7 microphone arrays on a single A²B bus with only an UTP cable, which transports data together with power. In addition, the triangular arrays can be placed up to 15 m away from each other, thus allowing to cover large areas without

increasing wirings complexity. It should be pointed out that power is supplied by the main node board which acquires data too. For the characterization of the triangular array, it was opted to employ the A^{2}B-DSP board described in Section 3.2.1.

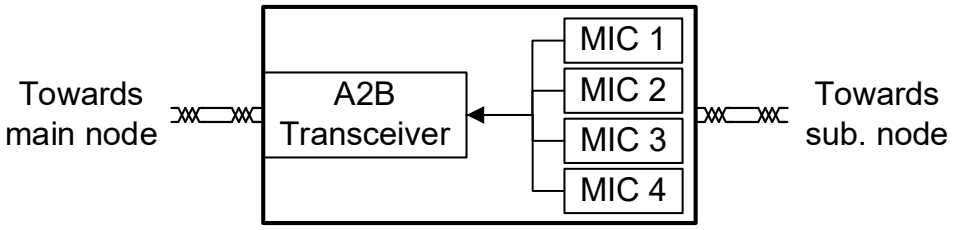


Figure 95. Block diagram of the triangular array.

The schematic of the experimental setup is shown in Figure 96. The measurement consists in obtaining Impulse Responses (IRs) of each capsule of the array for each DoA (D = 240) employed in simulations. For this purpose, the triangular array was mounted on a two-axis turntable and a loudspeaker, placed at 1 m, was employed to reproduce the test signal [91]. A PC controls the turntable, which receives the commands over an ethernet connection. In addition, the same PC is employed to play and record the audio signals, by means of the A^{2}B-DSP board, which is connected to the PC with an USB cable. The A^{2}B-DSP, called interface board in Figure 96, drives the loudspeaker, namely the studio monitor 8351A by Genelec, through an S/PDIF connection. In such a way, the measurement setup is full-digital.

The test signal is an Exponential Sine Sweep (ESS) [109], pre-equalized for flattening the spectrum of the sound source in the range 50 Hz – 18 kHz within ±0.5 dB. For each direction the ESS is played and recorded, and IRs are calculated by convolving the recorded signals with the inverse filter associated with the ESS, namely the inverse sweep. Finally, the $C_{m,d,k}$ matrix of equation (3.6) is obtained by applying a Fast Fourier Transform (FFT) to the IRs. Subsequent processing is unchanged with respect to the simulations. Polar patterns obtained for both simulations and measurements are shown in Figure 97.

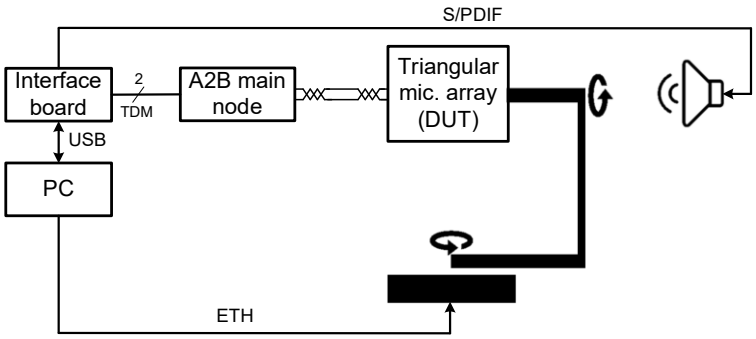


Figure 96. Experimental measurement setup.

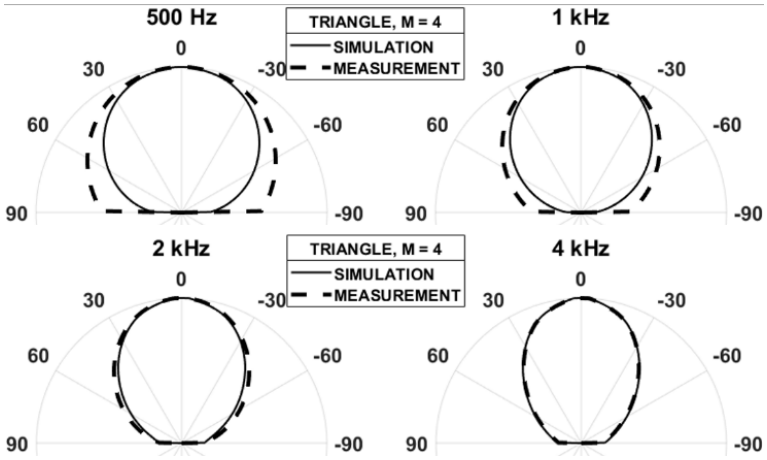


Figure 97. Polar patterns of the triangular array in the four octaves of its working frequency range. Solid lines are simulation results, while dashed lines are measurement results.

It can be observed a good matching between the two methods, particularly at the octaves centered at 2 kHz and 4 kHz. Polar patterns obtained from simulations at 500 Hz and 1 kHz show narrower beams in comparison to the measured ones. In fact, the experimental approach is affected by non-idealities, like geometrical, related to the measurement system, and constructive, related to the MEMS capsules, which are not identical in terms of magnitude and phase response.

Experimental results proved that the A^2B triangular array offers good beamforming performance in the frequency range between 300 Hz and 3.4 kHz. The presented

array can be employed as a single microphone array, or it can be connected to other triangular arrays to increase directivity. In fact, by exploiting the A^2B features, meta-arrays composed by triangular arrays can be easily built in two or three dimensions depending on the application. The next sections will describe in detail the implementation of some meta-arrays.

3.3.2 Planar Array

The easiest microphone array that can be built is a planar array. It was decided to realize a planar microphone array with 7 triangular boards since they saturate the available bandwidth of a single A^2B network. The designed planar meta-array is shown in Figure 98 and the realization of the prototype can be seen in Figure 99.

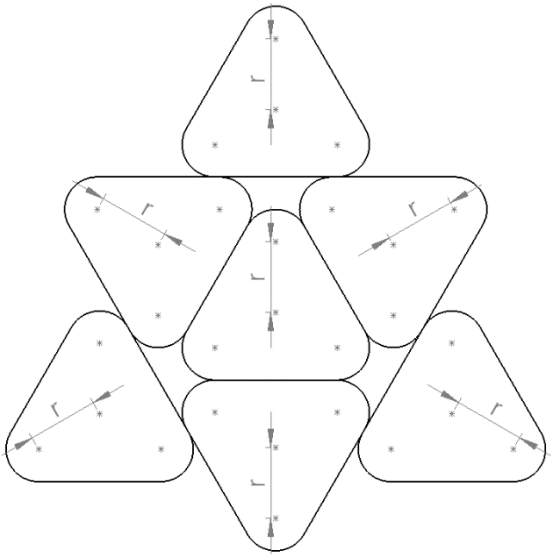


Figure 98. Planar meta-array composed by 7 A^2B triangular arrays.

Figure 99. Prototype of the realized planar microphone array with 7 triangular arrays (28 capsules).

The design procedure and the analysis followed the one described in the previous section. Firstly, simulations were carried out from 20 Hz to 3.4 kHz. Although the upper limit was not modified, the lower limit was decreased to 20 Hz to investigate the performance improvement due to the increase of the maximum distance between the capsules s_{max}.

The target function required by equation (3.6) was modified to produce three virtual microphones, with a polar pattern of a fourth order cardioid described by (3.7) . The virtual microphones were uniformly distributed along the azimuth at angles 0°, 120°, and -120°, with an elevation of 45°, as shown in Figure 100.

Then, to evaluate the array performance, the parameters Q and BW were calculated for the following combinations:

1. Only the central array ($M = 4$)
2. Central array plus the three triangular boards around it ($M = 16$)
3. All the 7 arrays ($M = 28$)

Results are shown in Figure 101 and in Figure 102. Please note that each parameter was averaged among the three virtual microphones to improve the reliability of the results.

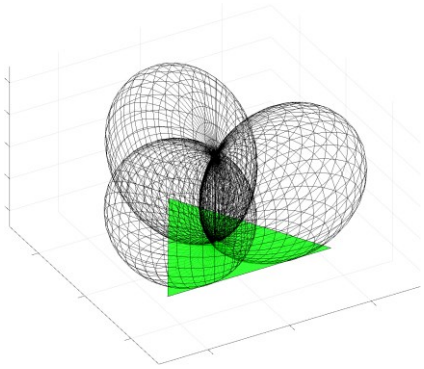


Figure 100. Target function composed by three cardioids of the fourth order.

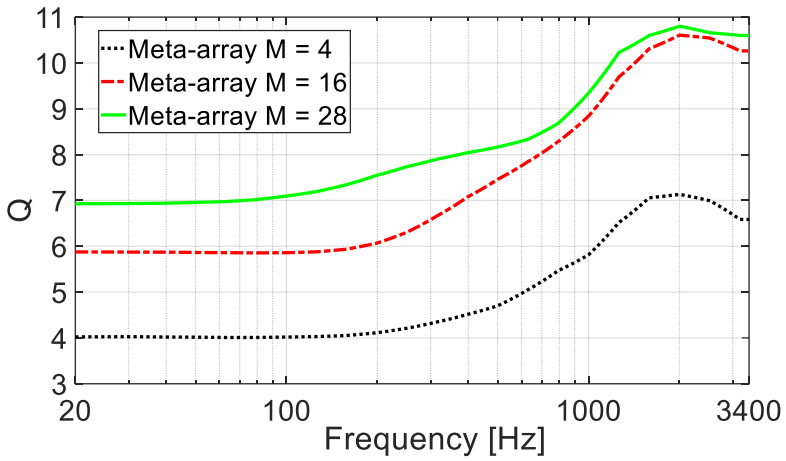


Figure 101. Q for meta-arrays with one triangular array (M = 4, black), four triangular arrays (M = 16, red), and seven triangular arrays (M = 28, green).

Then, the A^{2}B clock propagation effects were applied to simulations to prove that even for meta-arrays the performance are not influenced. A latency value μ between -10 ns and 10 ns was randomly assigned to each node, with a uniform probability distribution. The standard deviation of random jitter σ was linearly increased by 0.2 ns for each node, starting from the minimum measured value of 1.6 ns. By doing this, the four capsules of each triangular array are affected by the same amount

of jitter and latency, which is what happens effectively. By analyzing Q and BW it was concluded that the A^2B clock propagation does not affect beamforming even in meta-arrays. Figure 103 shows the directivity factor Q for the meta-array with 7 triangular boards ($M = 28$). As it can be observed, the green curve, which represents the ideal case, is overlapped with the black curve, which is the case where clock propagation is considered. To observe significant changes of Q, clock propagation effects must be magnified by three orders of magnitude (red curve).

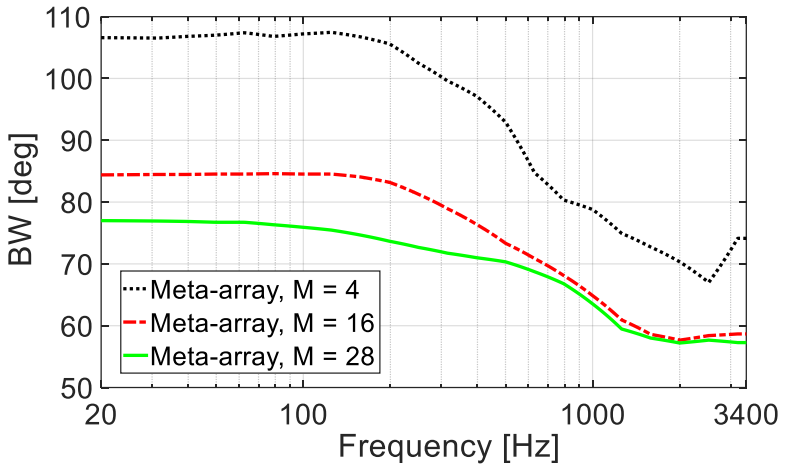


Figure 102. BW for meta-arrays with one triangular array (M = 4, black), four triangular arrays (M = 16, red), and seven triangular arrays (M = 28, green).

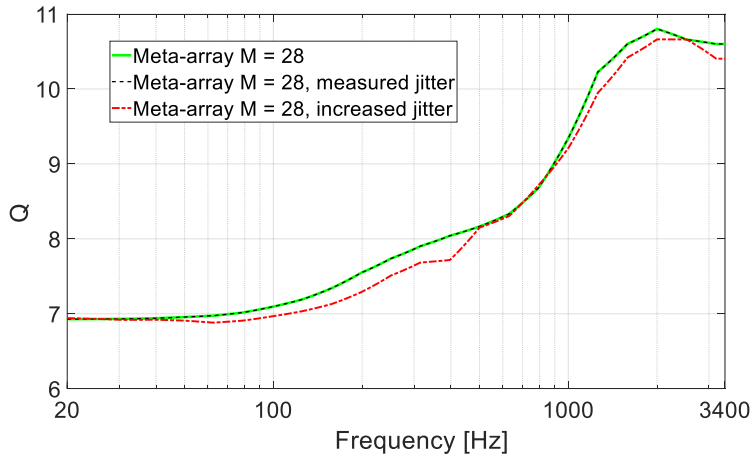


Figure 103. Q for meta-array with 7 triangular arrays, no jitter (green), jitter measured values (black), and jitter increased by a factor of 1000 (red).

3.3.3 Three-dimensional Array

Meta-arrays are not limited to planar geometries, but three-dimensional ones can be realized. In this chapter, it will be explained the design, realization and test of a three-dimensional meta-array based on the A^2B triangular array. This meta-array may be employed for applications such as Ambisonics recordings, room impulse response characterization, and noise source identification.

Design of the meta-array

It was chosen to design a microphone array able to reach a 5^{th} order Ambisonics. Ambisonics is a spatial audio format which is based on the decomposition of the sound field into components called Spherical Harmonics (SH). SH are organized into orders which defines the spatial accuracy. Given the desired Ambisonics order, the number of SH can be found as:

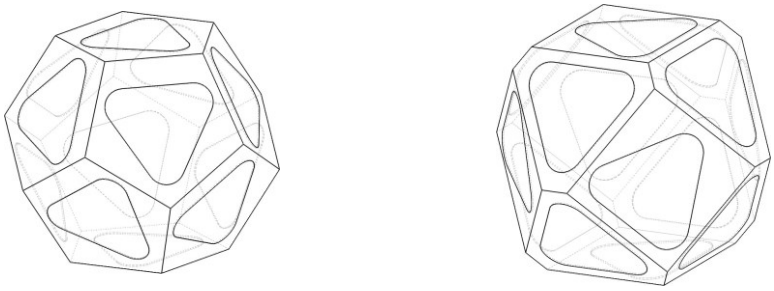
$$o = (n+1)^2 \tag{3.14}$$

where o is the number of SH and n is the Ambisonics order. If n equals to 5, equation (3.14) leads to 36 SH. As pointed out in [110], the number of microphones should be twice the number of SH for a correct spatial sampling over the whole frequency range. For example, for a 5^{th} order Ambisonics the array should have 72 microphones. However, a lower number of microphones can still be employed on a limited frequency range.

Since it was decided to build the meta-array by employing A^2B triangular arrays, 18 units would be needed to reach a total of 72 microphones. These devices should be connected by three A^2B busses, due to bandwidth limitations, thus increasing the complexity of the acquisition board. In addition, most of the existing technologies, like Dante and MADI, support up to 64 channels. For these reasons it was decided to limit the number of microphones to 56, namely 14 triangular arrays on two A^2B busses.

Typically, Ambisonics microphone arrays have spherical geometries to sample as equally as possible the sound field. However, the designed meta-array is based on triangular planar array, so firstly geometries that better approximate a sphere were identified. Two geometries were chosen, namely the dodecahedron (12 triangular arrays) and the truncated octahedron (14 triangular arrays). Successively, FEM simulations were carried out to identify the geometry that guaranteed the better performance in terms of spatial accuracy. For this purpose, two parameters were employed:

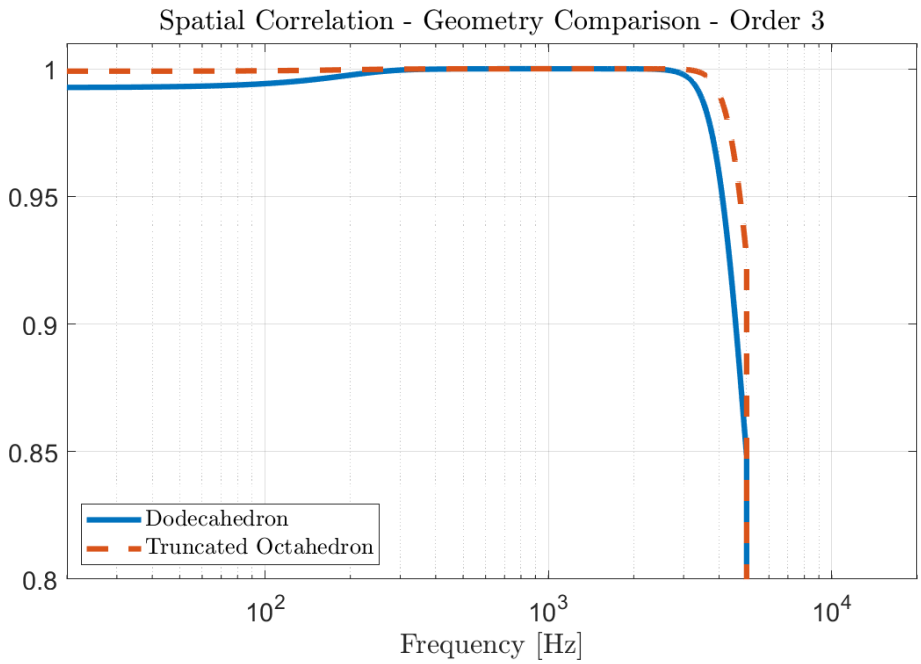
the Level Difference (LD) and the Spatial Correlation (SC) [104]. The LD is expressed in dB, and 0 dB corresponds to the exact level match between the generated and the ideal SH. The SC is a scalar less than or equal to 1, where 1 represents the perfect pattern reconstruction. The two geometries employed for the simulations are shown in Figure 104.



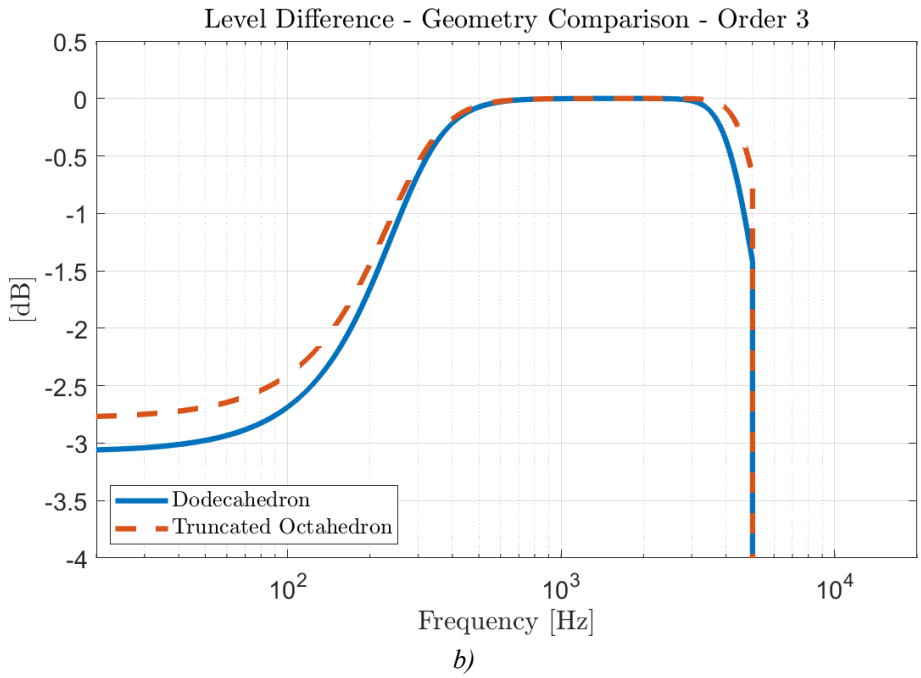
a) *b)*

Figure 104. The two proposed geometries with triangular arrays disposition. a) shows the dodecahedron, while b) show the truncated octahedron.

Simulations were performed to obtain the beamforming matrix *H*, described by equation (3.6), over the frequency range 5.86 Hz ÷ 5004 Hz. From the matrix *H* it was possible to calculate the parameters LD and SC. Results are shown in Figure 105, Figure 106, Figure 107, and Figure 108. In particular, Figure 105 and Figure 106 represent LD and SC of the two geometries for the 3rd and the 5th orders Ambisonics respectively. Figure 107 and Figure 108 shows SC and LD of the 5 Ambisonics orders for the dodecahedron and the truncated octahedron respectively. As it can be seen from all the figures, the truncated octahedron shows better SC and LD over a greater frequency range with respect to the dodecahedron. In fact, this geometry has a greater number of microphones that allows a better sampling of the acoustic field. For these reasons, the truncated octahedron geometry was chosen for the realization of a prototype and from now on it will be referred to as *A2BOcta56*.

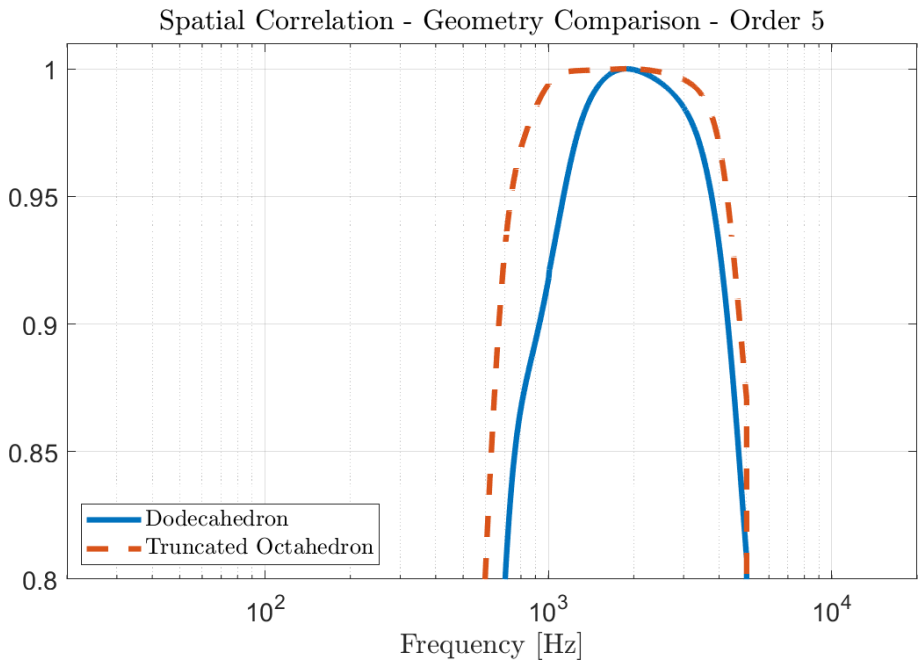


a)

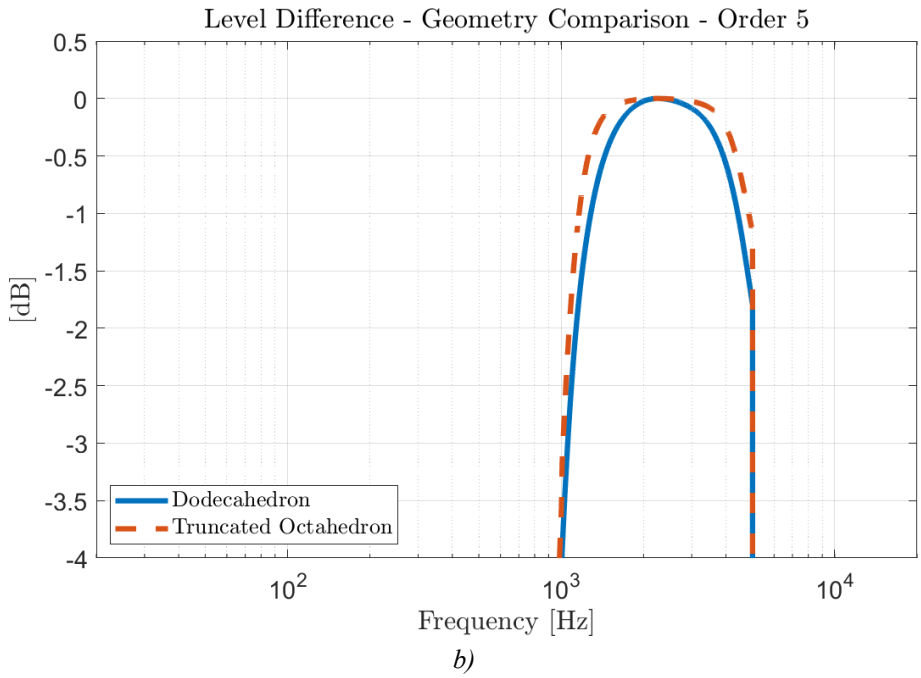


b)

Figure 105. SC (a) and LD (b) of the two geometries at the 3rd order Ambisonics.

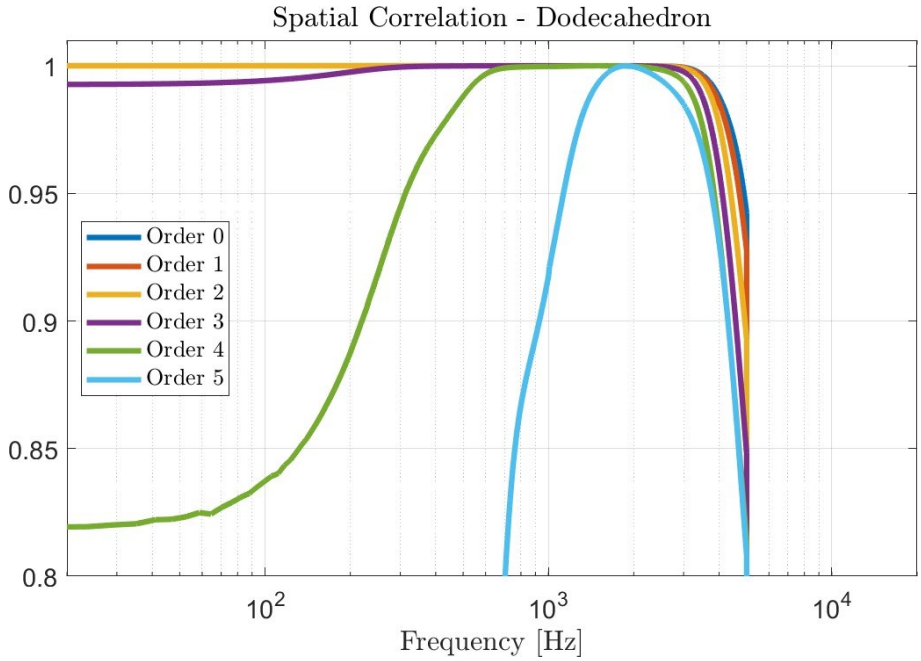


a)

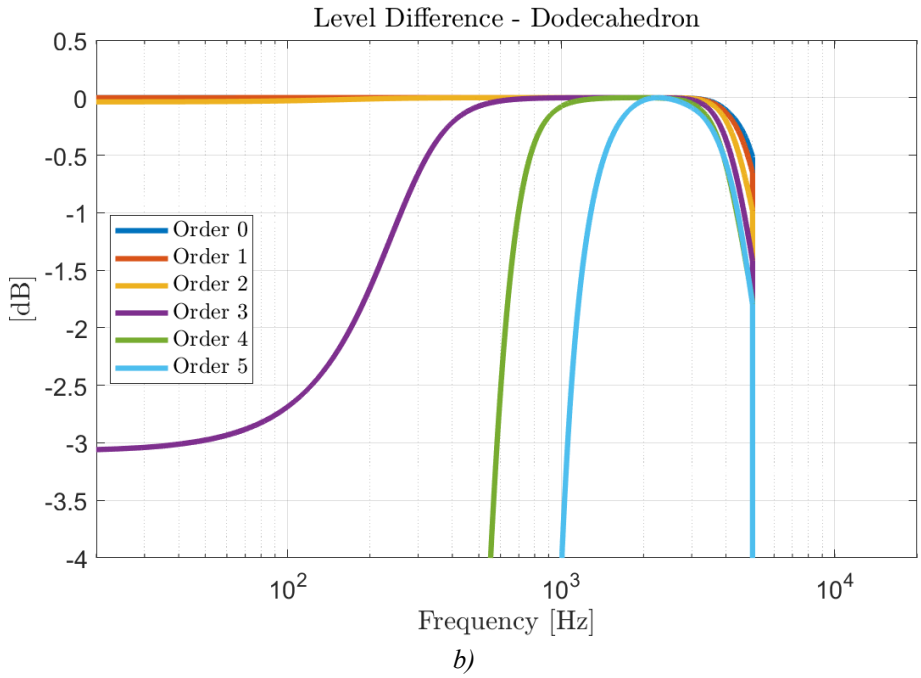


b)

Figure 106. SC (a) and LD (b) of the two geometries at the 5th order Ambisonics.

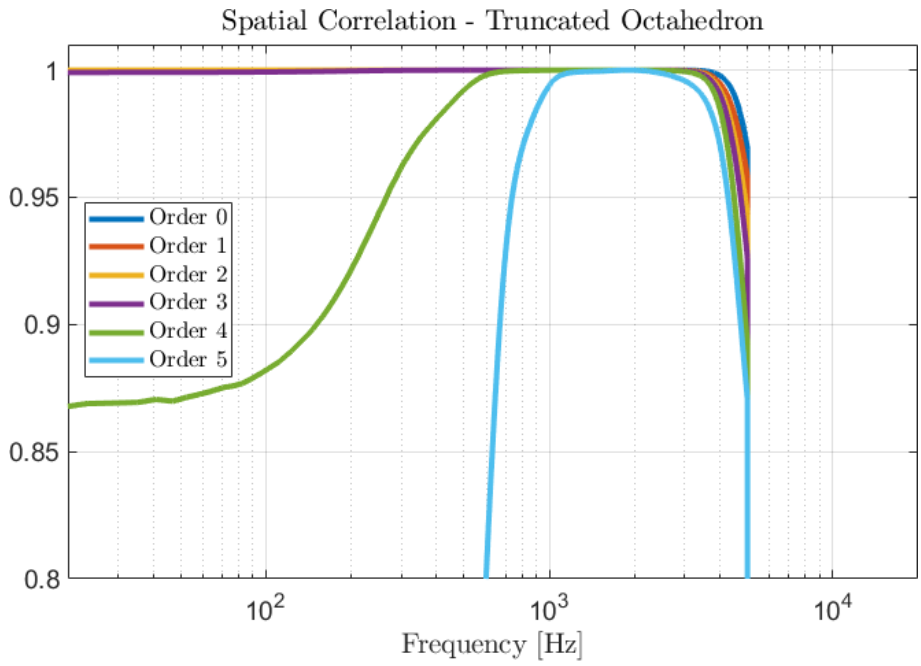


a)

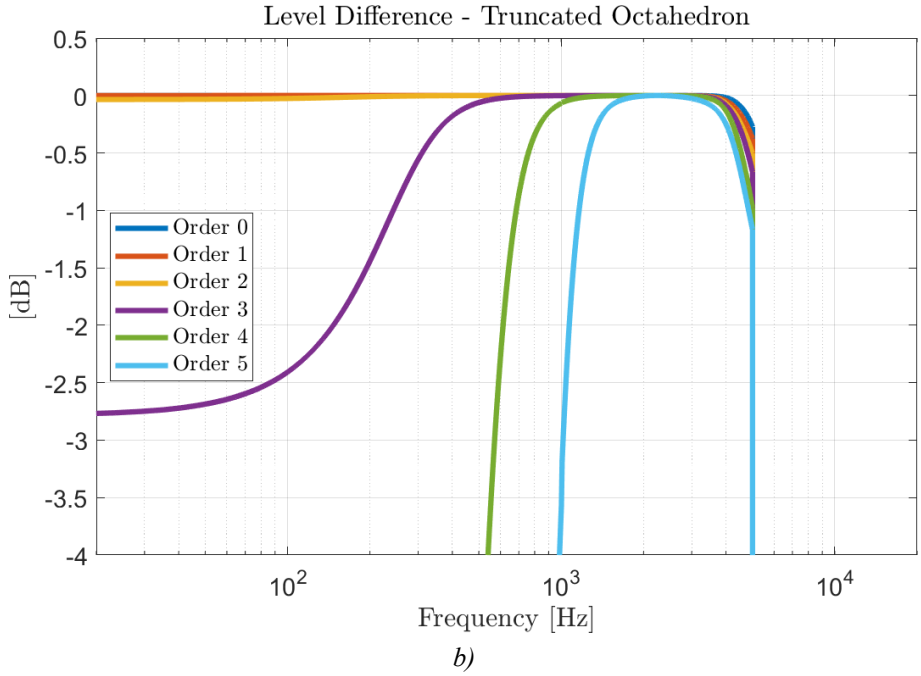


b)

Figure 107. SC (a) and LD (b) of the dodecahedron for the 5 Ambisonics orders.



a)



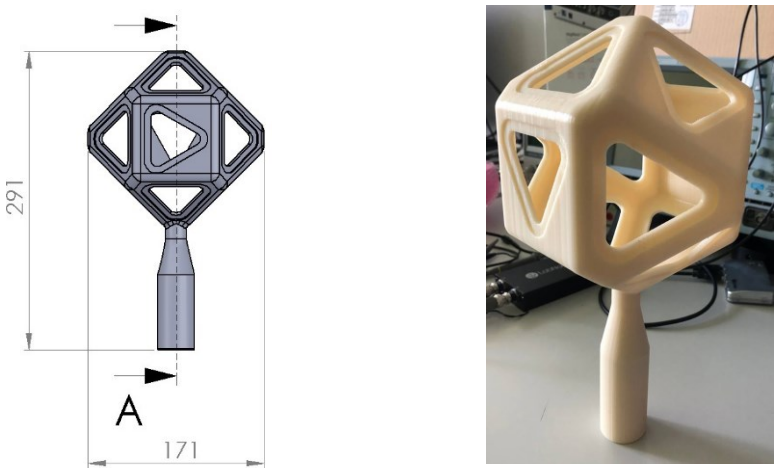
b)

Figure 108. SC (a) and LD (b) of the octahedron for the 5 Ambisonics orders.

Realization

First, the shell of the array was designed by adding a handle and fitting the edge of the model to accommodate to the triangular arrays. Figure 109a shows the 3D model of the shell with quotes, while Figure 109b shows the 3D printed shell.

Successively, since 20 units of the triangular array were available, they were characterized in order to choose the most similar ones. The AOP and the phase response of each unit were measured. The AOP is the sound pressure level at which the microphone exhibits a 10% of THD. This parameter was employed to identify capsules that were potentially damaged during the mounting process. The block diagram of the measurement setup is shown in Figure 110. The A^2B triangular array, namely the DUT, is mounted on a baffle with dimensions 1 mx1 m and it is positioned at 1 m distance from the loudspeaker. The loudspeaker is the Genelec 8351A. Audio signals are played and recorded from a PC through two audio interfaces: the RME MADIface USB and the A^2B-DSP. The former allows the conversion from MADI to USB, which is a technology suitable for the connection to the PC. The A^2B-DSP, which is described in Section 3.2.1, is an audio interface with processing and protocol conversion capabilities. For these tests, it is mainly employed for the conversion from MADI to A^2B and S/PDIF. The A^2B bus is connected to the triangular array, while the S/PDIF link drives the loudspeaker. In addition, it was also employed a reference microphone (4188 by Bruel&Kjaer), which is not depicted in Figure 110. The measurement setup in laboratory is shown in Figure 111.



a) *b)*

Figure 109. a) shows the 3D model of the array with quotes. b) shows the 3D printed shell of the microphone array.

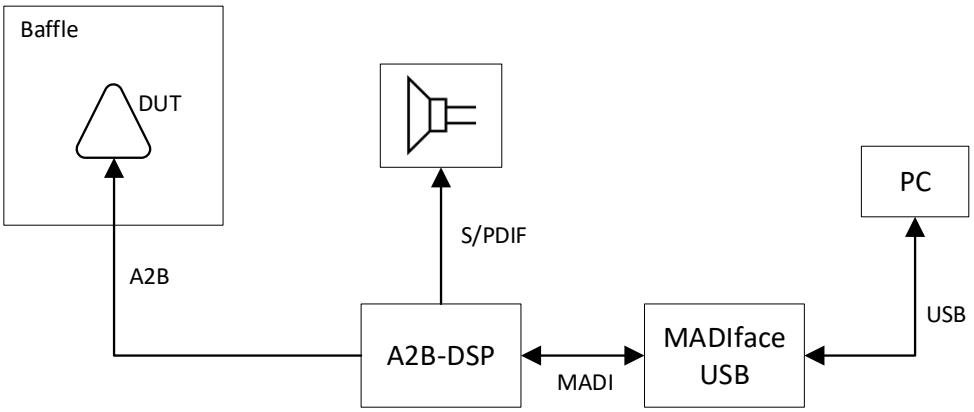


Figure 110. Block diagram of the measurement setup of the A^2B triangular array.

Figure 111. Measurement setup in laboratory.

The test signal that was employed for the AOP measurement is pink noise band-pass filtered in the 1 kHz octave band. The AOP measurement was carried out with the following steps:

1. **Reference microphone calibration.** The B&K reference microphone is employed as a sound level meter to calculate the equivalent sound level (*Leq*). The calibration is carried out with a microphone calibrator, which emits a 1 kHz sinusoidal tone at 94 dB(SPL). The signal of the microphone is recorded with the same I/O interface shown in Figure 110. The recorded signal was analyzed with Aurora plugin *ITUP56 Time History Analysis* to calculate the full-scale value of the reference microphone. This value will be used in the next step.
2. **Pink noise recording with the reference microphone.** Then the loudspeaker was placed at 1 m from the reference microphone and the signal test (pink noise) was recorded. By entering in the plugin *ITUP56 Time History Analysis* the full-scale value of the reference microphone calculated in the previous step, it was found the *Leq* of the reference microphone.
3. **Pink noise recording with the DUT.** The loudspeaker was now positioned at 1 m from the DUT, namely the triangular array mounted on the baffle, and the pink noise was recorded. By analyzing the recording with the plugin *ITUP56 Time History Analysis*, and by entering the *Leq* value of the reference microphone, it was calculated the AOP of the DUT.

This measurement was carried for the 20 triangular array boards that were available. The mean AOP was 128 dB(SPL) and it was employed to choose units with an AOP within ±0.5 dB of the mean value.

To further investigate the performance of the triangular arrays, phase response measurements were carried out with the same setup of Figure 110. The measurement employs the ESS as test signal, which is pre-equalized for flattening the spectrum of the loudspeaker. The procedure for obtaining the IRs of the capsules of the array was described in Section 3.3.1. Then, the IRs were processed in Matlab to obtain the phase responses, which are shown in Figure 112. As it can be noted, the phase response was analyzed in the frequency range from 300 Hz to 600 Hz as it is the most significant range. The lower limit of 300 Hz is caused by the limited length of the IRs (i.e., about 200 samples). In fact, the measurement was not carried out in an anechoic room, thus the first reflection was removed from the IRs. It can be seen that

capsules show a very similar behavior, except for one microphone that deviate from the mean value.

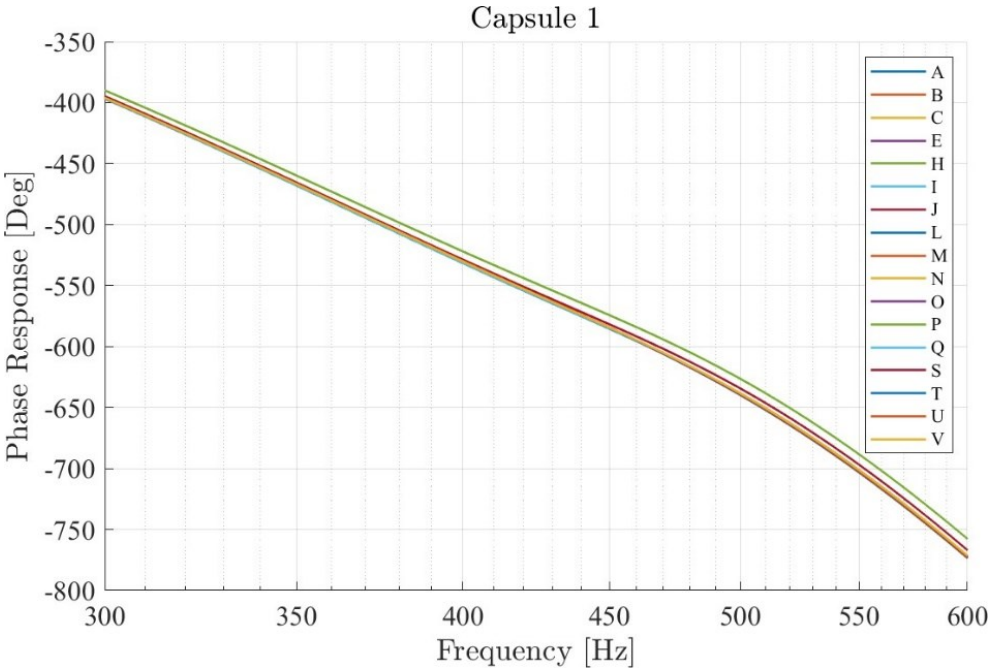


Figure 112. Phase response of the central capsule of each triangular array.

Once the 14 triangular arrays were identified, they were cabled and mounted on the shell of the A2BOcta56. They are connected in daisy-chain in two A^2B busses, 7 boards for each bus. It is not possible to connect more than 7 triangular arrays in a single bus for bandwidth limitations. This can be verified by calculating the RESPCYCS parameter (see Section 2.3) with the desired network configurations. The signals of the two A^2B busses are then synchronously acquired by the A^2B-DSP board. The A2BOcta56 requires two UTP cables, i.e., 4 wires, to connect the two A^2B busses. It was decided to equip the array with a RJ45 connector to exploit two ethernet pairs for the A^2B connections. The connector was inserted in the handle of the array, as shown in Figure 113. Since, the A^2B-DSP offers two XLR connectors for A^2B, it was realized an adapter from RJ45 to XLR.

Finally, before fixing the triangular arrays to the shell with hot glue, sound absorbing material was inserted inside the shell. In fact, since the digital MEMS employed are omnidirectional, they would capture sound coming from their rear side,

namely the inside of the array. Absorbing material helps to reduce this phenomenon. Photos of the assembly of the A2BOcta56 array are shown in Figure 114 and Figure 115.

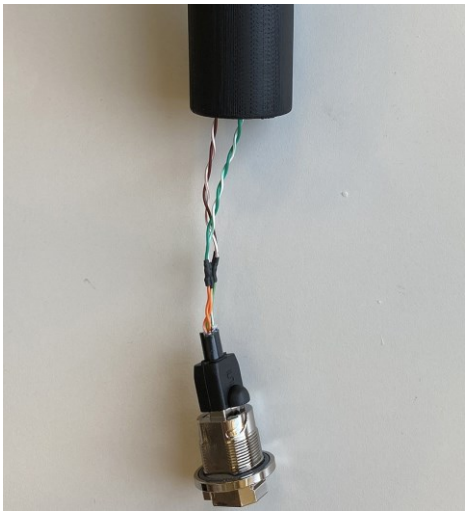
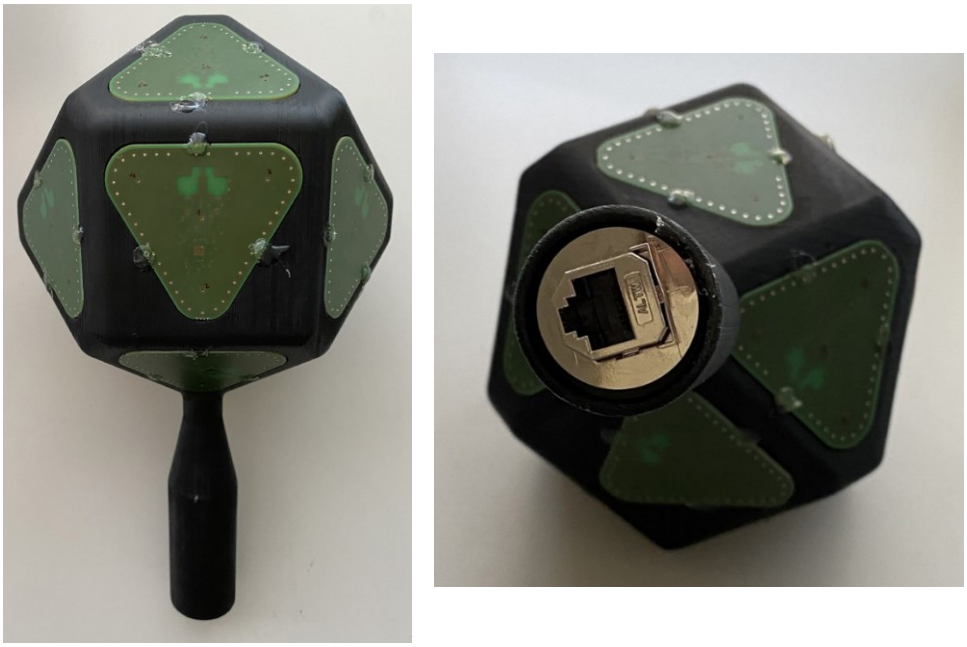


Figure 113. RJ45 connector of the A2BOcta56 microphone array. The connector is then inserted in the handle of the array.

a) *b)*

Figure 114. Photos of the assembly of the A2BOcta56 array.



a) *b)*

Figure 115. Photos of the A2BOcta56 assembled. In a) a lateral view of the array. In b) a particular of the connector inserted in the handle.

Finally, the SigmaDSPs of the A^2B-DSP were configured to reorganize channels. In fact, the A^2B-DSP receives 28 audio channels from each of the A^2B busses, data passes through the DSPs and then audio channels are concatenated and output on the MADI line by the FPGA. Hence, each DSP receives up to 32 channels coming from one of the A^2B main transceivers and it sends them to the FPGA. So DSP1 manages channels from 1 to 32 and DSP2 manages channels from 33 to 64. However, since in this application only 28 audio channels are received from each A^2B bus, the MADI output channels from 29 to 32 would be empty and data would restart from channel 33. For this reason, DSP1 was configured to accept the first four channels from DSP2, while DSP2 was configured to shift its channels so that data would start from its first channel. The block diagram is shown in Figure 116.

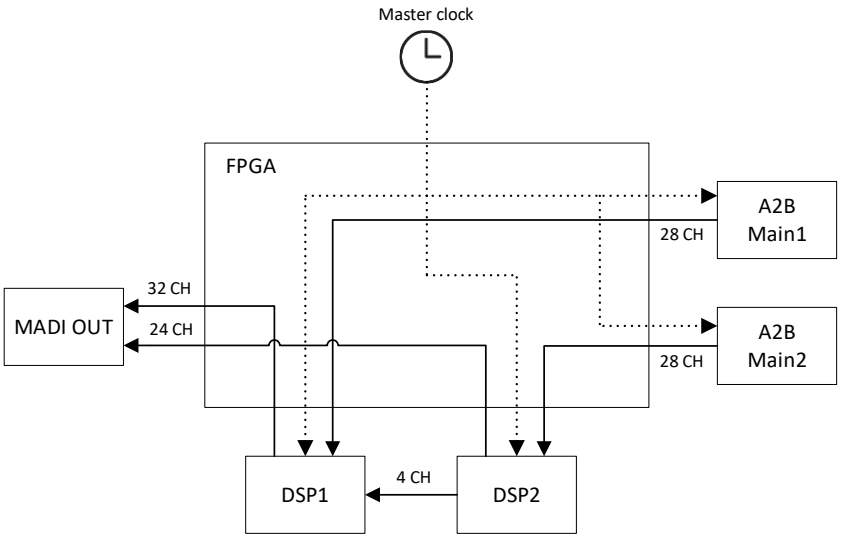


Figure 116. Block diagram of the A²B-DSP configuration to reorganize A2BOcta56 audio channels.

Experimental results

The A2BOcta56 was characterized in order to calculate the *H* matrix described by the equation (3.6) (see Section 3.3.1). Firstly, the *C* matrix must be obtained from measurements with the same technique employed for the triangular array (see Section 3.3.1). The block diagram of the setup used for these measurements is shown in Figure 117.

The A2BOcta56, namely the DUT, is mounted on the two-axis turning table, which is driven by a PC running a Matlab script that controls the measurement. The A2BOcta56 signals are acquired by the A²B-DSP, and then they are converted to MADI and sent to the RME Madiface USB, which is the audio interface that communicates with the PC. The test signal is generated by the PC and it is sent to the A²B-DSP and then converted to S/PDIF format to drive the loudspeaker, i.e., the Genelec 8351A. The measurements were carried out in laboratory and Figure 118 and Figure 119 show the setup. As it can be seen in Figure 119, the rotating axis of the turning table was covered with absorbing material to avoid undesired reflections.

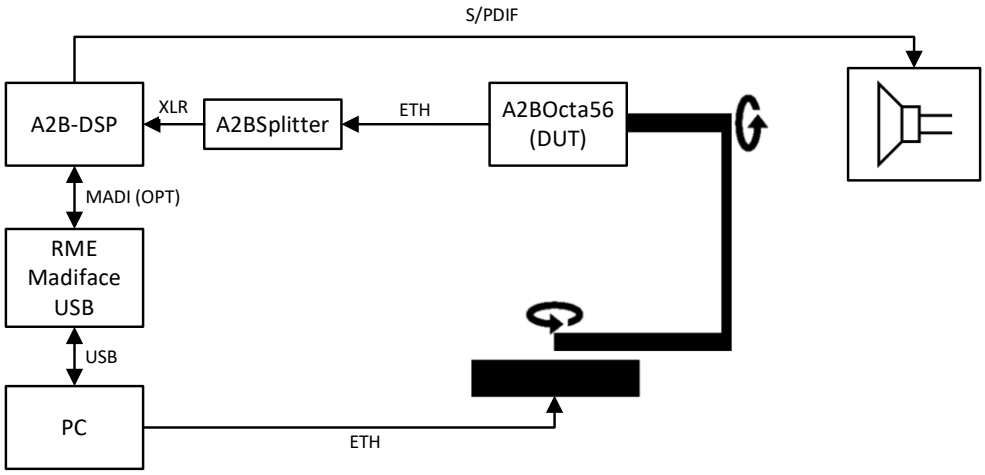


Figure 117. Block diagram of the measurement setup of the A2BOcta56.

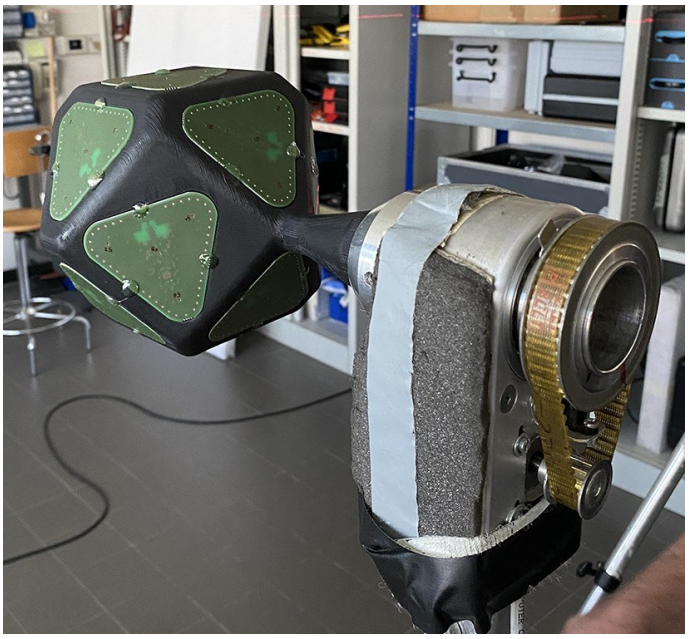


Figure 118. A2BOcta56 mounted on the two-axis turning table.

Figure 119. Measurement setup showing the loudspeaker and the DUT on the turning table.

This measure was repeated at three distances namely 0.5 m, 1 m, and 2 m so that data is obtained for both the near-field and the far-field. From the obtained data it was possible to calculate the IRs. Each IR was cut so to exclude the first sound reflection, thus simulating an anechoic environment down to a certain frequency, which is dependent on the length of the IR. In this case, IRs of about 310 samples were obtained, leading to a low-frequency limit of 155 Hz. Examples of the obtained IRs are shown in Figure 120. As it can be noted in Figure 120a, the IR at 0.5 m shows a reflection around the sample 220. This reflection is caused by the rotating axis on which the A2BOcta56 is mounted, leading to the fact that the absorbing material was not sufficient to completely avoid the reflection. It was decided to keep the reflection in the IRs, so as not to shorten them too much.

To evaluate the spatial performance of the array, the parameters LD and SC were calculated with the measured data, and they were compared with the ones obtained in simulations. The comparison of the measured and simulated LD and SC are shown in figures from Figure 121 to Figure 126 for the distances 0.5 m, 1 m, and 2 m. The LD is more significant at low frequency, while the SC is more significant at high frequency. As it can be observed in these figures, experimental results are in line with the simulated ones, especially at high frequency. At low frequency, a good match can be noted for the distances 1 m and 2 m, while at 0.5 m a greater deviation can be observed. It may be caused by the interference of the rotating axis, as it was observed on the IRs (Figure 120a).

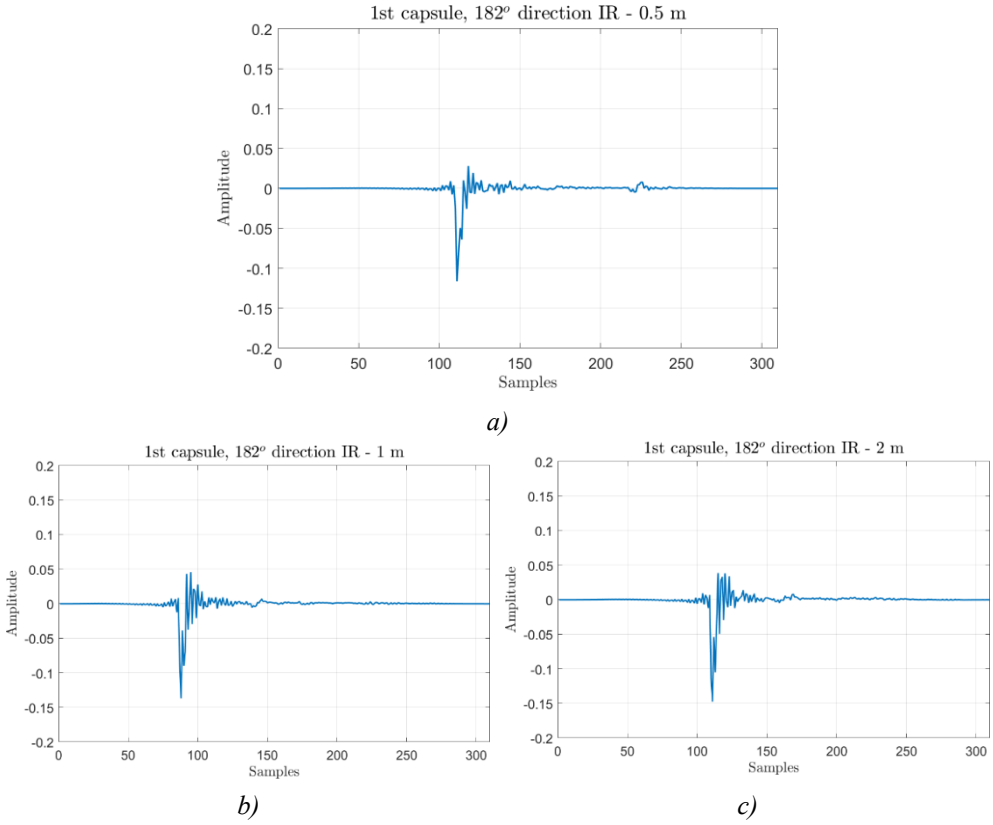
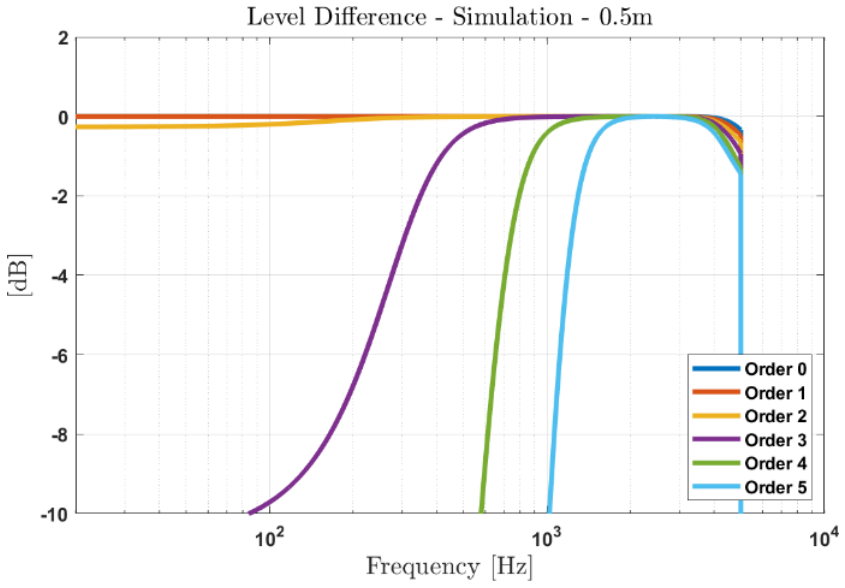
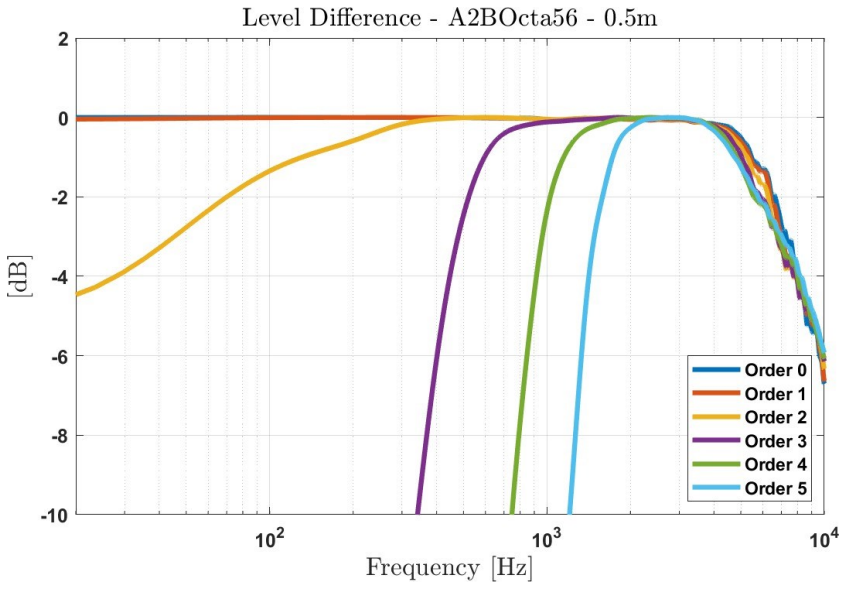


Figure 120. IRs obtained at 0.5 m, 1 m, and 2 m.

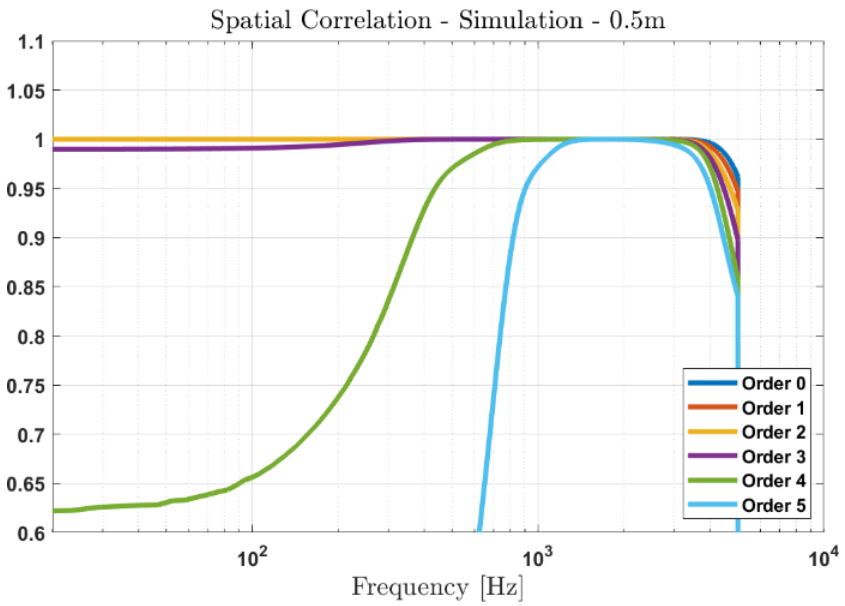


a)

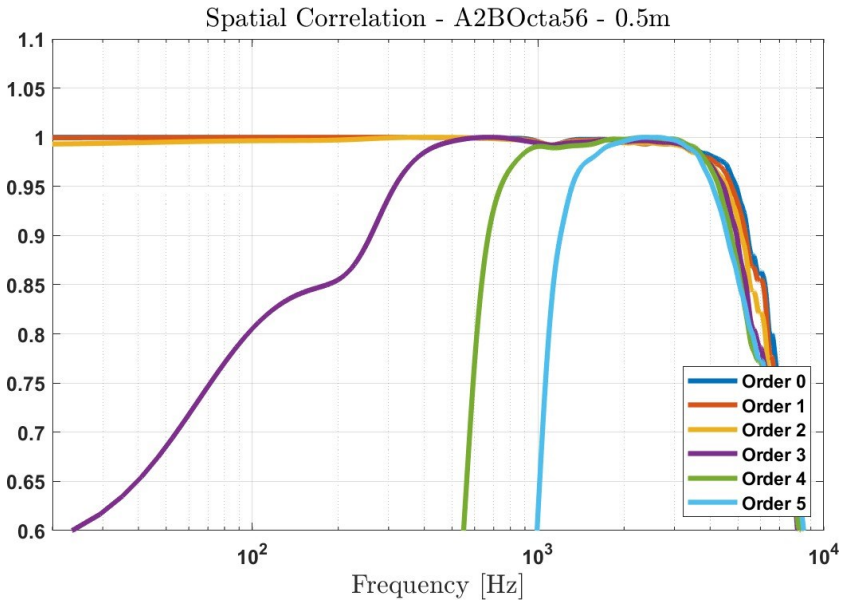


b)

Figure 121. Level Difference calculated from simulations (a) and from measurements (b) for the distance 0.5 m.

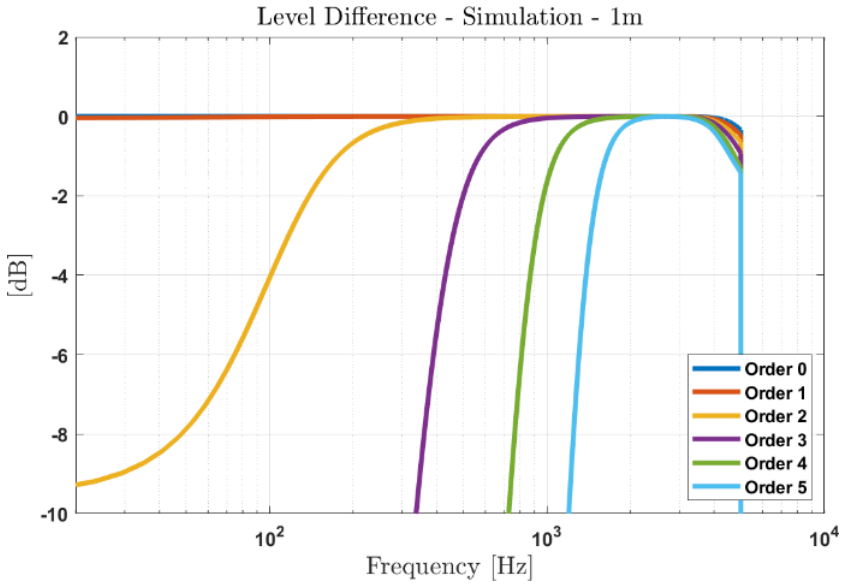


a)

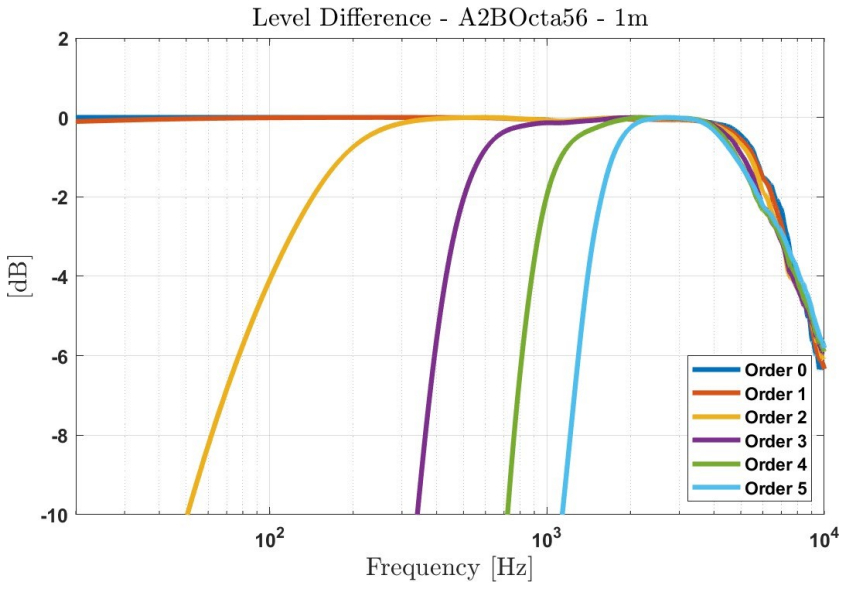


b)

Figure 122. Spatial correlation calculated from simulations (a) and from measurements (b) for the distance 0.5 m.

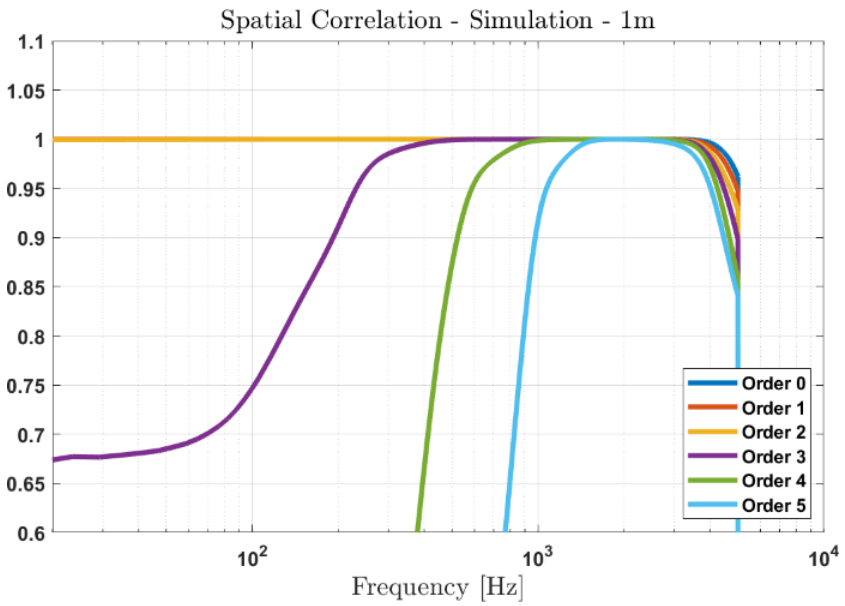


a)

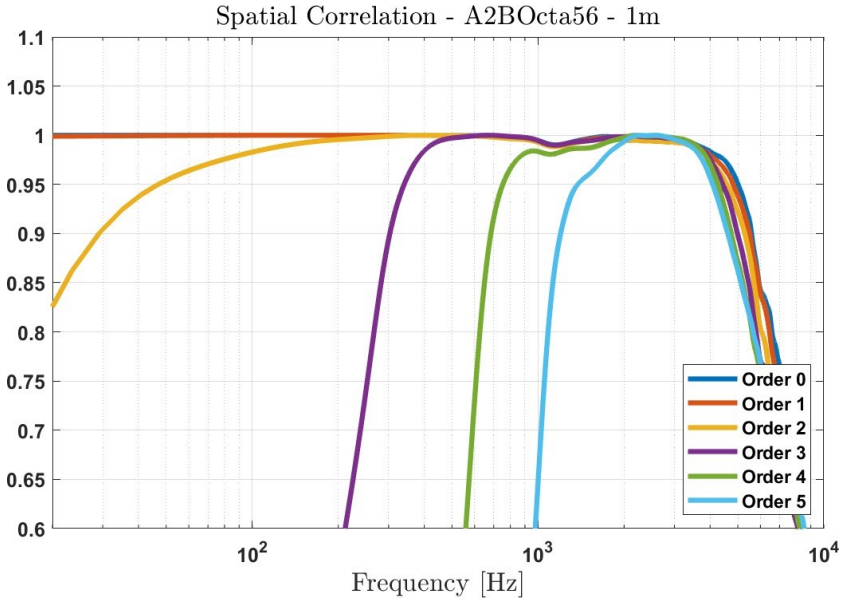


b)

Figure 123. Level Difference calculated from simulations (a) and from measurements (b) for the distance 1 m.

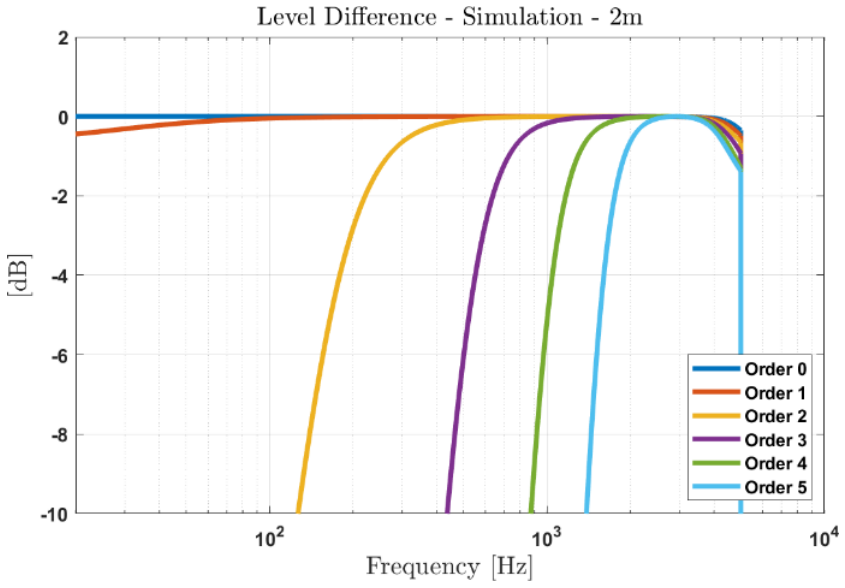


a)

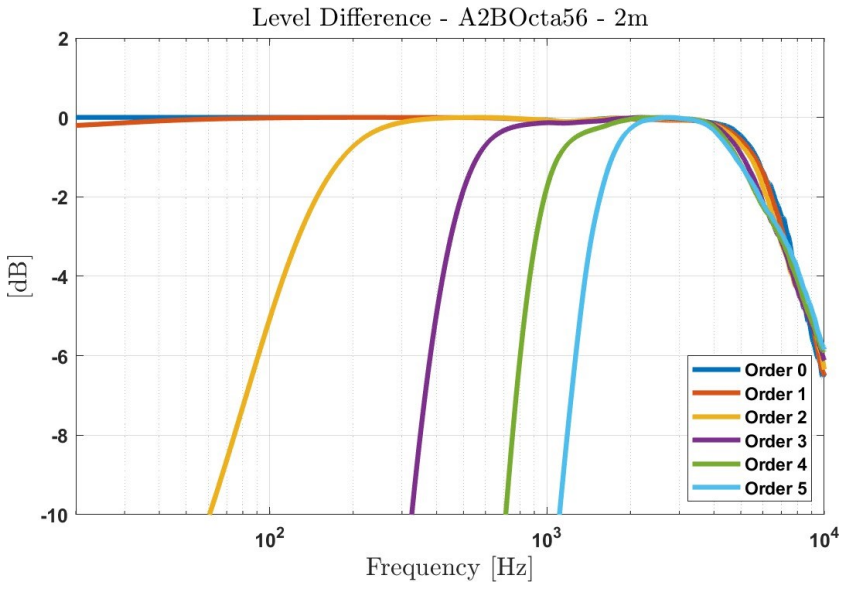


b)

Figure 124. Spatial correlation calculated from simulations (a) and from measurements (b) for the distance 1 m.

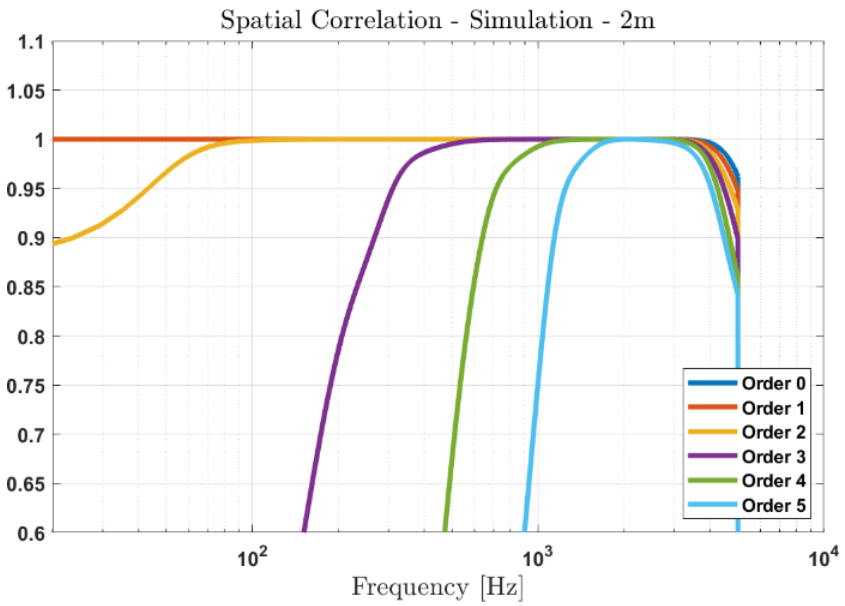


a)

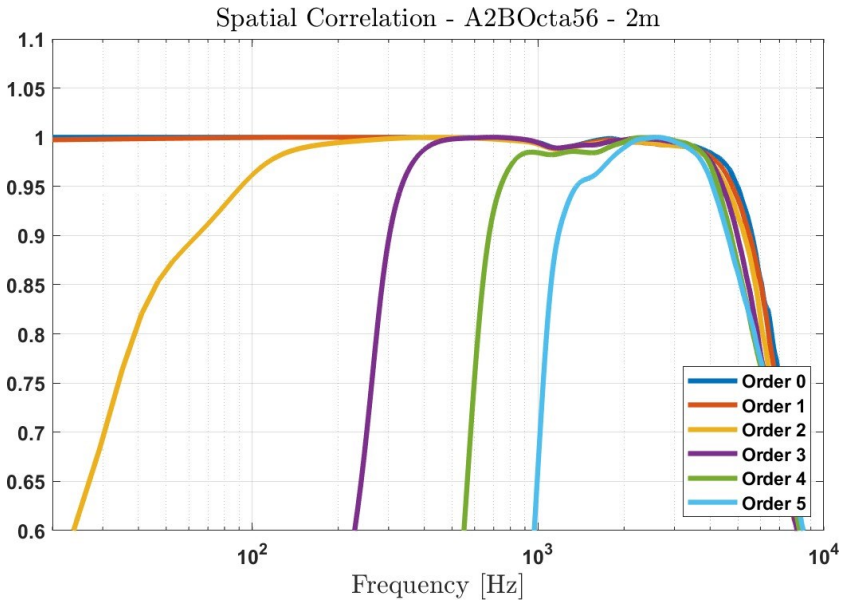


b)

Figure 125. Level Difference calculated from simulations (a) and from measurements (b) for the distance 2 m.



a)



b)

Figure 126. Spatial correlation calculated from simulations (a) and from measurements (b) for the distance 2 m.

Finally, to identify a frequency range in which SH are reconstructed correctly, it was chosen a range of acceptability of the LD and SC parameters. The ranges of acceptability are [-1 dB, 0 dB] for the LD, and [0.95, 1] for the SC. Hence, two frequency limits were obtained for each Ambisonics order and for the three measurement sets at the three distances. The lower limit was chosen according to LD, while the upper limit was chosen according to SC. Results are shown in Table XXII, Table XXIII, and Table XXIV. In conclusion, the array is able to reconstruct SH up to the 5th order Ambisonics in the frequency range from 1720 Hz to 4090 Hz. If a lower Ambisonics order is required, the frequency range increases, e.g., for a 1st order Ambisonics it is from 20 Hz to 4789 Hz.

Table XXII - Acceptability frequency ranges for the measure at 0.5 m

Ambisonics order	**Lower limit [Hz]**	**Upper limit [Hz]**
1	20	4754
2	131	4567
3	595	4457
4	1148	4287
5	1723	4082

Table XXIII - Acceptability frequency ranges for the measure at 1 m

Ambisonics order	**Lower limit [Hz]**	**Upper limit [Hz]**
1	20	4818
2	183	4577
3	575	4428
4	1122	4084
5	1725	4084

Table XXIV - Acceptability frequency ranges for the measure at 2 m

Ambisonics order	**Lower limit [Hz]**	**Upper limit [Hz]**
1	20	4789
2	183	4569
3	561	4426
4	1099	4264
5	1720	4090

Conclusions

In this work, a new architecture for audio signal distribution was presented. It aims to satisfy the new requirements and to overcome the issues of audio transport in automotive and professional or consumer audio applications. In fact, recent research topics such as immersive audio, 3D audio or spatial audio led to the development of innovative recording and reproduction systems, which demand more and more channels. Consequently, the cost and the complexity of these systems must be carefully considered during the design process.

The goal of this work was to present an audio distribution architecture that allows simplifying the cabling and the complexity of multichannel systems composed by a great number of transducers. After a brief description of A^2B, in this dissertation several professional audio protocols were presented and compared, namely AES10, AVB, Dante, Ravenna and AES67.

Even though, most of today's systems adopt audio-over-ethernet or audio-over-IP technologies, the A^2B bus is a promising solution to implement the front-end section of the data transport architecture. In fact, it is a low-cost and simple technology to acquire and distribute signals between the increasing number of transducers of current audio systems. The A^2B nodes are connected in daisy-chain, and on the same bus audio data, configuration commands and power are transported together. In addition, access to the bus is managed by dedicated transceivers that do not require programming, but only an initial configuration, thus reducing the design cost. Moreover, A^2B offers great performance: low-jitter, and just 2 samples of latency. However, it also presents some limitations due to the static configuration of the bus and to the limited cable length.

The proposed architecture is composed by three main parts: the peripheral devices, the transport layer, and the processing and protocol conversion section. The peripheral devices are A^2B enabled systems of any kind (e.g., amplifiers,

microphones, processing units, etc.). Some examples were presented in the case studies section (Chapter 3). The processing and protocol conversion section is implemented by a custom board called A^{2}B-DSP. This device offers the connection to two A^{2}B busses and it can convert data between A^{2}B and some of the most common audio protocols, namely AVB, AES10, and USB UAC-2.0. In addition, it can implement signal processing thanks to the two on-board SigmaDSPs. The protocol conversion section was added for compatibility reasons with the existing systems, to support longer range technologies, and to collect signals with a higher bandwidth protocol. Furthermore, a new A^{2}B software feature called A^{2}B Plug-and-Play was introduced, thus allowing to connect and disconnect different devices on the A^{2}B bus, without the need to reconfiguring it. This feature is necessary for applications where the bus configuration may change dynamically, such as the listening room or the microphone arrays presented in this work. In fact, by default the configuration of an A^{2}B bus is static, as the microcontroller on the main board must know in advance the configuration of the subordinate nodes, their peripheral and the channel routing. A^{2}B Plug-And-Play allows saving the configuration of a subordinate node in a local EEPROM accessible to the main node to configure it.

The principal characteristics of the proposed architecture are:

- Scalability and modularity
- Distributed design
- Simple design of the front-end devices
- Simple and low-cost wiring
- Full-digital data path

In the last chapter of this dissertation, three case studies that leverage the proposed architecture were presented, namely the audio distribution in a car (ILZ), a WFS listening room and three types of microphone arrays.

An A^{2}B distributed amplifier was developed for the ILZ application, which is a proprietary technology of ASK Industries S.p.A. The amplifier board mounts an A^{2}B transceiver and a 1-channel class-D amplifier and it was designed to fit to the *Ultra-Flat* loudspeaker. The ILZ application benefits from the proposed architecture as it allows to greatly simplify the wiring between the numerous active loudspeakers (up to 75% compared with traditional solutions [53]), while keeping low the cost of a single node. During this work, it was also realized a device, namely the *A^{2}B Streamer*, to drive the amplifiers from a PC by employing an USB connection.

Professional audio applications may exploit the proposed architecture too. In this dissertation, it was presented the realization of a WFS listening room. 8-channel active soundbars were developed, each one equipped with an A^2B transceiver. The daisy-chain topology of the A^2B helps to reduce the cabling complexity, while the design of a single device is simplified thanks to the dedicated A^2B transceivers, which allow avoiding powerful processors or FPGA to manage the communication.

Finally, several A^2B microphone arrays were presented and the concept of meta-array, namely array of arrays, was introduced. First, the A^2B triangular microphone array was developed. It is equipped with an A^2B transceiver which directly acquires four PDM MEMS microphones, thus avoiding the design of preamplifiers and ADCs. Thanks to the A^2B characteristics, it is possible to easily distribute the triangular arrays, for example in a meeting room, to realize a cost-effective distributed microphone array. Alternatively, several triangular arrays can be combined to realized planar or three-dimensional microphone arrays. In this work, two designs were proposed: a planar array (with 7 triangular arrays) and a three-dimensional octahedral array (with 14 triangular arrays). These arrays improve the low frequency spatial performance, with respect to the triangular array, thus allowing to increase the directivity in this frequency range.

List of Publications

[A1] N. Rocchi, A. Toscani, G. Chiorboli, D. Pinardi, M. Binelli and A. Farina, "Transducer Arrays Over A²B Networks in Industrial and Automotive Applications: Clock Propagation Measurements," IEEE Access, vol. 9, pp. 118232-118241, 2021, doi: 10.1109/ACCESS.2021.3106710.

[A2] N. Rocchi, A. Toscani, D. Pinardi, M. Binelli, L. Chiesi, A. Farina, E. Bonomi, L. Tronchin, "A Modular, Low Latency, A2B-based Architecture for Distributed Multichannel Full-Digital Audio Systems", in 2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA), Sep. 2021, pp. 1-8, doi: 10.1109/I3DA48870.2021.9610947

[A3] A. Bevilacqua, L. Tronchin, M. Binelli, L. Chiesi, N. Rocchi, D. Pinardi, A. Toscani, A. Farina, "Application of a Wave Field Synthesis (WFS) Audio System based on A2B protocol: a case study," in 2nd Symposium: The Acoustics of Ancient Theatres, Jul. 2022

[A4] A. Bevilacqua, L. Tronchin, M. Binelli, L. Chiesi, N. Rocchi, D. Pinardi, A. Toscani, A. Farina, "Design of a multichannel audio system based on A2B architecture," in 2nd Symposium: The Acoustics of Ancient Theatres, Jul. 2022

[A5] D. Pinardi, N. Rocchi, A. Toscani, M. Binelli, G. Chiorboli, A. Farina, L. Cattani, "An Innovative Architecture of Full-Digital Microphone Arrays Over A²B Network for Consumer Electronics", IEEE Transactions on Consumer Electronics, vol. 68, no. 3, pp. 200-208, Aug. 2022, doi: 10.1109/TCE.2022.3187453

[A6] A. Toscani, N. Rocchi, D. Pinardi, M. Binelli, L. Saccenti, A. Farina, S. Pavoni, M. Vanali, "Low-cost Structural Health Monitoring System for Smart Buildings", in 2nd International Conference on Sustainable Mobility Applications, Renewables and Technology (SMART), Nov. 2022, pp. 1-7

Bibliography

[1] J. Ahonen, V. Pulkki, and T. Lokki, "Teleconference Application and B-format Microphone Array for Directional Audio Coding," in *Proceedings of the AES 30th International Conference*, Mar. 2007, pp. 1–10.

[2] K. A. A. and F. Jason, "Audio Source Localization as an Input to Virtual Reality Environments," in *144th AES Convention*, 2018.

[3] A. Farina, D. Pinardi, M. Binelli, M. Ebri, and L. Ebri, "Virtual Reality for Subjective Assessment of Sound Quality in cars," in *144th Audio Engineering Society Convention*, 2018, pp. 1–9.

[4] D. Pinardi, L. Ebri, C. Belicchi, A. Farina, and M. Binelli, "Direction Specific Analysis of Psychoacoustics Parameters inside Car Cockpit: A Novel Tool for NVH and Sound Quality," *SAE Technical Paper 2020-01-1547*. pp. 1–8, 2020. doi: 10.4271/2020-01-1547.

[5] S. M. Kim, C. J. Chun, and H. K. Kim, "Multi-channel audio recording based on superdirective beamforming for portable multimedia recording devices," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 3, pp. 429–435, Aug. 2014, doi: 10.1109/TCE.2014.6937327.

[6] M. Kessler, "Introducing the Automotive Audio Bus (A2B)," in *2017 AES International Conference on Automotive Audio*, Sep. 2017, pp. 1–5.

[7] "AES3-2009 (r2019): AES standard for digital audio engineering - Serial transmission format for two-channel linearly represented digital audio data." Audio Engineering Society, Inc., 2009.

[8] "AES10-2020: AES Recommended Practice for Digital Audio Engineering - Serial Multichannel Audio Digital Interface (MADI)." Audio Engineering Society, Inc., pp. 1–20, 2020.

[9] "IEEE Standard for Local and Metropolitan Area Networks--Timing and Synchronization for Time-Sensitive Applications," *IEEE Std 802.1AS-2020*

(Revision of IEEE Std 802.1AS-2011), pp. 1–421, 2020, doi: 10.1109/IEEESTD.2020.9121845.

[10] "IEEE Standard for Local and metropolitan area networks--Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pp. 1–119, 2010, doi: 10.1109/IEEESTD.2010.5594972.

[11] "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams," *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. C1-72, 2010, doi: 10.1109/IEEESTD.2009.5375704.

[12] "AVB Endpoint Design Guide." XMOS, 2014.

[13] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)*, pp. 1–499, 2020, doi: 10.1109/IEEESTD.2020.9120376.

[14] G. M. Garner and H. Ryu, "Synchronization of audio/video bridging networks using IEEE 802.1AS," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 140–147, Feb. 2011, doi: 10.1109/MCOM.2011.5706322.

[15] E. Mann *et al.*, "AVB Software Interfaces and Endpoint Architecture Guidelines." AVnu Alliance, pp. 1–84, 2015.

[16] "IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks," *IEEE Std 1722-2016 (Revision of IEEE Std 1722-2011)*, pp. 1–233, 2016, doi: 10.1109/IEEESTD.2016.7782716.

[17] "IEEE Standard for Device Discovery, Connection Management, and Control Protocol for IEEE 1722(TM) Based Devices," *IEEE Std 1722.1-2013*, pp. 1–366, 2013, doi: 10.1109/IEEESTD.2013.6627916.

[18] "Avnu Alliance." https://avnu.org/

[19] A. Gothard, R. Kreifeldt, and C. Turner, "AVB for Automotive Use." pp. 1–9, 2014.

[20] "Audinate." https://www.audinate.com/

[21] Audinate, "The Relationship Between Dante, AES67 and SMPTE ST 2110." pp. 1–21, Dec. 2019.

[22] S. Heinzmann, R. Michl, and A. Hildebrand, "RAVENNA - Operating Principles." ALC NetworX, pp. 1–25, 2011.

[23] H. Weibel and S. Heinzmann, “Media Clock Synchronization Based On PTP,” in *44th International Conference: Audio Networking*, Nov. 2011, pp. 1–8.

[24] A. Holzinger and A. Hildebrand, “Realtime Linear Audio Distribution Over Networks: A Comparison of Layer 2 and 3 Solutions Using the Example of Ethernet AVB and RAVENNA,” in *44th International Conference: Audio Networking*, Nov. 2011, pp. 1–9.

[25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” Jul. 2003. doi: 10.17487/rfc3550.

[26] A. Hildebrand, “RAVENNA & AES67.” ALC NetworX, pp. 1–22, 2014.

[27] “AES standard for audio applications of networks - High-performance streaming audio-over-IP interoperability,” *AES67-2018*, pp. 1–72, 2018.

[28] F. Rumsey, “Audio Over IP: A Guide From the Summit Through the Jungle,” *J. Audio Eng. Soc.*, vol. 69, no. 4, pp. 276–280, Apr. 2021.

[29] D. Scheirman, “Large-scale Loudspeaker Arrays: Past, Present and Future (Part One - Computer Control, User Interface and Networked Audio Considerations),” in *AES 59th International Conference*, Jul. 2015, pp. 1–12.

[30] N. Bouillot *et al.*, “Best Practices In Network Audio,” *J. Audio Eng. Soc.*, vol. 57, no. 9, pp. 729–741, Sep. 2009.

[31] J. Triggs, “The A to Z of A2B Applications,” *Analog Dialogue*, vol. 54, no. 3. pp. 1–4, 2020.

[32] N. Rocchi, A. Toscani, G. Chiorboli, D. Pinardi, M. Binelli, and A. Farina, “Transducer Arrays Over A^2B Networks in Industrial and Automotive Applications: Clock Propagation Measurements,” *IEEE Access*, vol. 9, pp. 118232–118241, 2021, doi: 10.1109/ACCESS.2021.3106710.

[33] W. Maichen, *Digital timing measurements. From scopes and probes to timing and jitter*. Springer, 2006. [Online]. Available: www.springer.com

[34] N. da Dalt and A. Sheikholeslami, *Understanding Jitter and Phase Noise*. Cambridge University Press, 2018. doi: 10.1017/9781316981238.

[35] ISO/IEC, *ISO/IEC TR 14165-117 : 2007(R2018), Information Technology - Fibre Channel - Part 117: Methodologies for Jitter and Signal Quality (MJSQ)*. 2018.

[36] C. Travis and P. Lesso, “Specifying the Jitter Performance of Audio Components,” in *AES 117th Convention*, 2004, pp. 1–15. [Online]. Available: https://www.aes.org/e-lib/online/browse.cfm?elib=12950

[37] J. N. Tripathi, V. K. Sharma, and H. Shrimali, “A Review on Power Supply Induced Jitter,” *IEEE Trans Compon Packaging Manuf Technol*, vol. 9, no. 3, pp. 511–524, Mar. 2019, doi: 10.1109/TCPMT.2018.2872608.

[38] J. Buckwalter, B. Analui, and A. Hajimiri, “Predicting Data-Dependent Jitter,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 51, no. 9, pp. 453–457, Sep. 2004, doi: 10.1109/TCSII.2004.834537.

[39] M. Miller and M. Schnecker, “A Comparison of Methods for Estimating Total Jitter Concerning Precision, Accuracy and Robustness,” in *DesignCon*, 2007, pp. 1–24.

[40] G. Soliman, “Improved Jitter Distribution Tail-Fitting Algorithm for Decomposition of Random and Deterministic Jitter,” *IEEE Trans Electromagn Compat*, vol. 62, no. 5, pp. 1852–1858, Oct. 2020, doi: 10.1109/TEMC.2019.2947222.

[41] B. A. Blesser, “Digitization of Audio: A Comprehensive Examination of Theory, Implementation, and Current Practice,” *Journal of the Audio Engineering Society*, vol. 26, 2012.

[42] Tektronix, “Understanding and Characterizing Timing Jitter.” pp. 1–20, 2017.

[43] Clockworks, “Technote 008 A2B I2S Clock Jitter.” pp. 1–31, 2020.

[44] J.-W. Lee, H.-J. Kim, and C. Yoo, “Spread spectrum clock generation for reduced electro-magnetic interference in consumer electronics devices,” *IEEE Transactions on Consumer Electronics*, vol. 56, no. 2, pp. 844–847, May 2010, doi: 10.1109/TCE.2010.5506010.

[45] AnalogDevices, “AD2420(W)-AD2426(W)-AD2427(W)-AD2428(W)-AD2429(W) Datasheet.” 2020.

[46] AnalogDevices, “AD2420(W)/6(W)/7(W)/8(W)/9(W) Automotive Audio Bus A2B Transceiver Technical Reference,” 2019.

[47] J. Cheer, “Active Sound Control in the Automotive Interior,” 2021, pp. 53–69. doi: 10.1007/978-3-030-51044-2_3.

[48] F. Rumsey, “Automotive audio drives forward,” *AES: Journal of the Audio Engineering Society*, vol. 66, no. 1–2, pp. 83–87, 2018.

[49] J. Boehm *et al.*, “A Distributed Audio System for Automotive Applications,” in *144th Audio Engineering Society Convention*, May 2018, pp. 1–6.

[50] S. Poferl, M. Becht, and P. de Pauw, “150 Mbit/s MOST, the Next Generation automotive infotainment system,” in *International Conference on Transparent Optical Networks*, 2010, vol. 150, no. 1, pp. 1–2.

[51] H. Lim, D. Herrscher, and F. Chaari, “Performance Comparison of IEEE 802 .1Q and IEEE 802.1 AVB in an Ethernet-based In-Vehicle Network,” in *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, 2012, pp. 1–6.

[52] L. lo Bello, “Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging,” in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Sep. 2014, pp. 1–8. doi: 10.1109/ETFA.2014.7005251.

[53] M. Crosio, “A2B and Ethernet in Automotive Applications: What, When, and How.” Analog Devices, pp. 1–4, 2019.

[54] J. Cheer, S. J. Elliott, and M. F. S. Gálvez, “Design and implementation of a car cabin personal audio system,” *AES: Journal of the Audio Engineering Society*, vol. 61, no. 6, pp. 412–424, 2013.

[55] X. Liao, J. Cheer, S. J. Elliott, and S. Zheng, “Design of a loudspeaker array for personal audio in a car cabin,” *AES: Journal of the Audio Engineering Society*, vol. 65, no. 3, pp. 226–238, 2017, doi: 10.17743/jaes.2016.0065.

[56] S. Goose, L. Riddle, C. Fuller, T. Gupta, and A. Marcus, “PAZ: In-Vehicle Personalized Audio Zones,” *IEEE Multimedia*, vol. 23, no. 4, pp. 32–41, 2016, doi: 10.1109/MMUL.2015.94.

[57] G. Pepe, L. Gabrielli, S. Squartini, L. Cattani, and C. Tripodi, “Deep Learning for Individual Listening Zone,” in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2020, pp. 1–6. doi: 10.1109/MMSP48831.2020.9287161.

[58] W. Gallian, F. M. Fazi, C. Tripodi, N. Strozzi, and A. Costalunga, “Optimisation of the target sound fields for the generation of independent listening zones in a reverberant environment,” in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, Sep. 2021, pp. 1–10. doi: 10.1109/I3DA48870.2021.9610888.

[59] A. Borroni, M. Martalo, C. Tripodi, and R. Raheli, “Experimental Analysis of Individual Listening Zone Algorithms in Controlled Environments,” in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, Sep. 2021, no. 1, pp. 1–7. doi: 10.1109/I3DA48870.2021.9610882.

[60] L. Strong, “Why do you need USB Audio Class 2?” XMOS, pp. 1–9, 2015. Accessed: Aug. 23, 2022. [Online]. Available: https://www.xmos.ai/download/Why-do-you-need-USB-Audio-Class-2(1.0).pdf

[61] XMOS, “USB Audio 2.0 Software,” 2016. https://www.xmos.ai/file/usbaudiodevice-software (accessed Aug. 31, 2022).

[62] XMOS, “USB Audio Design Guide,” 2016. https://www.xmos.ai/download/sw_usb_audio:-sw_usb_audio-[design-guide](6.16.1alpha1).pdf (accessed Sep. 01, 2022).

[63] N. Rocchi *et al.*, “A Modular, Low Latency, A2B-based Architecture for Distributed Multichannel Full-Digital Audio Systems,” in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, Sep. 2021, pp. 1–8. doi: 10.1109/I3DA48870.2021.9610947.

[64] A. J. Berkhout, “Holographic approach to acoustic control,” *AES: Journal of the Audio Engineering Society*, vol. 36, no. 12, pp. 977–995, 1988.

[65] A. J. Berkhoul, D. de Vries, and P. Vogel, “Acoustic control by wave field synthesis,” *Journal of the Acoustical Society of America*, vol. 93, no. 5, pp. 2764–2778, 1993, doi: 10.1121/1.405852.

[66] “The Game of Life.” https://gameoflife.nl/en

[67] S. Kaneko *et al.*, “Development of a 64-channel spherical microphone array and a 122-channel loudspeaker array system for 3D sound field capturing and reproduction technology research,” *144th Audio Engineering Society Convention 2018*, pp. 1–6, 2018.

[68] M. Pezzoli *et al.*, “A Dante Powered Modular Microphone Array System,” in *145th Audio Engineering Society Convention*, Oct. 2018, pp. 1–5.

[69] T. Cohrs and L. Treybig, “Array microphones and signal processing within an ethernet-based AVB network,” in *2016 IEEE 6th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Sep. 2016, pp. 145–148. doi: 10.1109/ICCE-Berlin.2016.7684741.

[70] P. Dodds, S. v. Amengual Gari, O. Brimijoin, and P. W. Robinson, “Auralization systems for simulation of augmented reality experiences in virtual environments,” in *5th International Conference on Spatial Audio*, Sep. 2019, pp. 29–34. doi: 10.22032/dbt.

[71] S. Devonport and R. Foss, “The Distribution of Ambisonic and Point Source Rendering to Ethernet AVB Speakers,” in *5th International Conference on Spatial Audio*, Sep. 2019, pp. 39–46. doi: 10.22032/dbt.

[72] R. Foss and S. Devonpoor, “An Immersive Audio Control System Using Mobile Devices and Ethernet AVB-Capable Speakers,” *Journal of the Audio Engineering Society*, vol. 66, no. 9, pp. 724–733, Sep. 2018, doi: 10.17743/jaes.2018.0037.

[73] T. Reussner, A. E. S. Member, C. Sladeczek, M. Rath, and S. Brix, “Audio Network-Based Massive Multichannel Spatial Audio Research,” *Journal of the Audio Engineering Society*, vol. 61, no. 4, pp. 235–245, 2013.

[74] N. Thiele, “Loudspeakers in Vented Boxes: Part 1,” *J. Audio Eng. Soc.*, vol. 19, no. 5, pp. 382–392, May 1971.

[75] N. Thiele, “Loudspeakers in Vented Boxes: Part 2,” *J. Audio Eng. Soc.*, vol. 19, no. 6, pp. 471–483, Jun. 1971.

[76] R. H. Small, “Closed-Box Loudspeaker Systems- Part 1: Analysis,” *J. Audio Eng. Soc.*, vol. 20, no. 10, pp. 798–808, Dec. 1972.

[77] R. H. Small, “Closed-Box Loudspeaker Systems- Part 2: Synthesis,” *J. Audio Eng. Soc.*, vol. 21, no. 1, pp. 11–18, Feb. 1973.

[78] H. Tokuno, O. Kirkeby, P. A. Nelson, and H. Hamada, “Inverse Filter of Sound Reproduction Systems Using Regularization,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E80-A, no. 5, pp. 809–820, May 1997.

[79] Angelo Farina, “Simultaneous measurement of impulse response and distortion with a swept-sine technique,” in *108th AES Convention*, 2000.

[80] R. Kerstens, D. Laurijssen, and J. Steckel, “Low-cost one-bit MEMS microphone arrays for in-air acoustic imaging using FPGA’s,” in *2017 IEEE SENSORS*, Oct. 2017, pp. 1–3. doi: 10.1109/ICSENS.2017.8234087.

[81] M. Turqueti, J. Saniie, and E. Oruklu, “Scalable acoustic imaging platform using MEMS array,” in *2010 IEEE International Conference on Electro/Information Technology*, May 2010, pp. 1–4. doi: 10.1109/EIT.2010.5612131.

[82] H. A. Sanchez-Hevia, R. Gil-Pita, and M. Rosa-Zurera, “FPGA-based real-time acoustic camera using pdm mems microphones with a custom demodulation filter,” in *2014 IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, Jun. 2014, pp. 181–184. doi: 10.1109/SAM.2014.6882370.

[83] A. Farina and S. Fontana, “Microphone Assembly Having a Reconfigurable Geometry,” US Patent 0379969 A1, 2019

[84] B. Li and T. Ju, “An Improved Sound Source Location Method for MEMS Microphone Array,” in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, Oct. 2019, pp. 469–472. doi: 10.1109/ICCT46805.2019.8947027.

[85] S. Chowdhury, M. Ahmadi, and W. C. Miller, “Design of a MEMS acoustical beamforming sensor microarray,” *IEEE Sens J*, vol. 2, no. 6, pp. 617–627, Dec. 2002, doi: 10.1109/JSEN.2002.807773.

[86] L. Li, K. Lian, J. Fu, P. Zhu, Z. Hu, and C. Guo, “Acoustic Enhanced Camera Tracking System Based on Small-Aperture MEMS Microphone Array,” *IEEE Access*, vol. 8, pp. 215827–215839, 2020, doi: 10.1109/ACCESS.2020.3041445.

[87] E. Zwyssig, F. Faubel, S. Renals, and M. Lincoln, “Recognition of overlapping speech using digital MEMS microphone arrays,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 7068–7072. doi: 10.1109/ICASSP.2013.6639033.

[88] D. Pinardi *et al.*, “An Innovative Architecture of Full-Digital Microphone Arrays Over A^2B Network for Consumer Electronics,” *IEEE Transactions on Consumer Electronics*, vol. 68, no. 3, pp. 200–208, Aug. 2022, doi: 10.1109/TCE.2022.3187453.

[89] J. S. Abel *et al.*, “A configurable microphone array with acoustically transparent omnidirectional elements,” in *127th Audio Engineering Society Convention 2009*, 2009, pp. 3–12.

[90] D. Ayllón, V. Benito-Olivares, C. Llerena-Aguilar, R. Gil-Pita, and M. Rosa-Zurera, “Three-dimensional microphone array for speech enhancement in hands-free systems for cars,” in *Proceedings of the AES International Conference*, 2012, pp. 1–7.

[91] A. Farina, S. Campanini, L. Chiesi, A. Amendola, and L. Ebri, “Spatial sound recording with dense microphone arrays,” in *AES 55th International Conference*, 2014, pp. 1–8.

[92] R. González, J. Pearce, and T. Lokki, “Modular design for spherical microphone arrays,” in *Proceedings of the AES International Conference*, 2018, pp. 1–7.

[93] J. Meyer and G. Elko, “A highly scalable spherical microphone array based on an orthonormal decomposition of the soundfield,” in *IEEE International*

Conference on Acoustics Speech and Signal Processing, May 2002, vol. 2, pp. II-1781-II–1784. doi: 10.1109/ICASSP.2002.5744968.

[94] S. Sakamoto, S. Hongo, T. Okamoto, Y. Iwaya, and Y. Suzuki, "Sound-space recording and binaural presentation system based on a 252-channel microphone array," *Acoust Sci Technol*, vol. 36, no. 6, pp. 516–526, 2015, doi: 10.1250/ast.36.516.

[95] R. Schultz-Amling, F. Kuech, M. Kallinger, G. del Galdo, J. Ahonen, and V. Pulkki, "Planar microphone array processing for the analysis and reproduction of spatial audio using Directional Audio Coding," in *124th Audio Engineering Society Convention 2008*, 2008, pp. 1–10.

[96] J. Lopez-Ballester, M. Cobos, J. J. Perez-Solano, G. Moreno, and J. Segura, "General purpose modular microphone array for spatial audio acquisition," in *138th Audio Engineering Society Convention*, 2015, pp. 1–6.

[97] I. Andras, P. Dolinsky, L. Michaeli, and J. Saliga, "Beamforming with small diameter microphone array," in *2018 28th International Conference Radioelektronika (RADIOELEKTRONIKA)*, Apr. 2018, pp. 1–5. doi: 10.1109/RADIOELEK.2018.8376368.

[98] Z. I. Skordilis, A. Tsiami, P. Maragos, G. Potamianos, L. Spelgatti, and R. Sannino, "Multichannel speech enhancement using MEMS microphones," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2015, pp. 2729–2733. doi: 10.1109/ICASSP.2015.7178467.

[99] B. Rafaely, "The Spherical-Shell Microphone Array," *IEEE Trans Audio Speech Lang Process*, vol. 16, no. 4, pp. 740–747, May 2008, doi: 10.1109/TASL.2008.920059.

[100] O. Kirkeby, F. Orduna, P. A. Nelson, and H. Hamada, "Inverse Filtering in Sound Reproduction," *Measurement and Control*, vol. 26, no. 9, pp. 261–266, 1993.

[101] H. Tokuno, O. Kirkeby, P. A. Nelson, and H. Hamada, "Inverse Filter of Sound Reproduction Systems Using Regularization," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E80-A, no. 5, pp. 809–820, 1997.

[102] R. H. Hardin and N. J. A. Sloane, "McLaren's improved snub cube and other new spherical designs in three dimensions," *Discrete Comput Geom*, vol. 15, no. 4, pp. 429–441, Apr. 1996, doi: 10.1007/BF02711518.

[103] D. Pinardi, “Spherical t-Design for Characterizing the Spatial Response of Microphone Arrays,” in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, Sep. 2021, pp. 1–8. doi: 10.1109/I3DA48870.2021.9610850.

[104] D. Pinardi and A. Farina, “Metrics for Evaluating the Spatial Accuracy of Microphone Arrays,” in *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, Sep. 2021, pp. 1–9. doi: 10.1109/I3DA48870.2021.9610887.

[105] Qiyue Zou, Xiaoxin Zou, Ming Zhang, and Zhiping Lin, “A Robust Speech Detection Algorithm in a Microphone Array Teleconferencing System,” in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, 2001, vol. 5, pp. 3025–3028. doi: 10.1109/ICASSP.2001.940295.

[106] “IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters,” *IEEE Std 1241-2010 (Revision of IEEE Std 1241-2000)*, no. January. pp. 1–139, 2011. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=5692954

[107] “IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices,” *IEEE Std 1658-2021*, vol. 2012. pp. 1–126, Feb. 2011. doi: 10.1109/IEEESTD.2012.6152113.

[108] InfineonTechnologies, “Im69D130,” 2017. https://www.infineon.com/dgdl/Infineon-IM69D130-DS-v01_00-EN.pdf?fileId=5546d462602a9dc801607a0e46511a2e

[109] A. Farina, “Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique,” in *108th Convention*, 2000, pp. 1–24.

[110] S. Moreau, J. Daniel, and S. Bertet, “3D sound field recording with higher order ambisonics - objective measurements and validation of 4th order spherical microphone,” in *Audio Engineering Society - 120th Convention*, May 2006, pp. 1–24.