Characterisation of Teensy 4.1 Ecosystem for Low-Latency Audio Network Transmission in Networked Music Performances

Leigh Daniel Murray
Department of Musicology
University of Oslo
Oslo, Norway
leighmurray@gmail.com

Stefano Fasciani
Department of Musicology
University of Oslo
Oslo, Norway
stefano.fasciani@imv.uio.no

***Abstract*—This paper evaluates and benchmarks the Teensy 4.1 Ecosystem as a low-cost platform for Networked Music Performances. The study focuses on bidirectional streaming of uncompressed audio and assesses various latency components across the system, including I2S passthrough, buffer passthrough, and Ethernet transmission in unidirectional and roundtrip modes, as well as "My Mouth to My Ear" latency over Local and Wide Area Networks. UDP bandwidth utilisation (both unidirectional and bidirectional) and compatibility with JackTrip software are also examined. Results show that a Teensy 4.1-based audio streaming system achieves one-way source-to-ear latencies as low as 0.51 ms at 96 kHz with an 8-sample buffer and 2.50 ms at 48 kHz with a 32-sample buffer, well below the Ensemble Performance Threshold of 25 ms, which marks the limit of perceivable delay. This low latency accommodates additional network propagation delays. Furthermore, the Teensy 4.1 efficiently utilises Ethernet bandwidth, minimising impacts during simultaneous data transmission. Overall, the findings indicate that microcontroller-based boards like the Teensy 4.1 are ideal for low-latency audio streaming, offering better performance and lower costs than general-purpose and some single-board computers.**

***Index Terms*—Networked Music Performance, Audio Streaming, Low-Latency Audio, Teensy, Latency Measurement**

I. Introduction

In musical performances, achieving synchronicity is straightforward when musicians are in the same physical space. However, geographical separation, financial constraints, and global health crises can hinder collaboration. This has prompted musicians and researchers to seek technological solutions, such as Telematic or Network Music Performance (NMP) technologies. By using modern high-speed WAN, these technologies enable real-time, high-quality, low-latency audio streaming, allowing musicians to perform synchronously from different locations, provided streaming latency remains manageable.

While several NMP software platforms are free and open-source, they often require high-performance or specialised hardware to achieve low latencies and high audio quality, leading to increased costs. Moreover, performance bottlenecks can arise from general-purpose operating systems, which are not suited for ultra-low-latency audio I/O. This paper evaluates the Teensy 4.1[1] as a microcontroller-based platform for ultra-low-latency audio streaming in NMP, presenting a compact and cost-effective solution.

Additionally, we review related NMP platforms to establish criteria that define state-of-the-art standards, using these criteria to assess the Teensy 4.1's capabilities. Finally, we provide recommendations for future research and development of microcontroller-based platforms aimed at achieving ultra-low-latency audio streaming in NMP.

II. Related Work

A. Latency

The single most important requirement for NMP is low latency audio streaming [1]–[3], which is typically measured as the time it takes for the audio created by one performer to be heard by the other(s), and has been referred to as the Over-all One-way Source-to-Ear delay (OOSE) [1] or Mouth to Ear (M2E) delay [4]. This is illustrated in Fig.1.

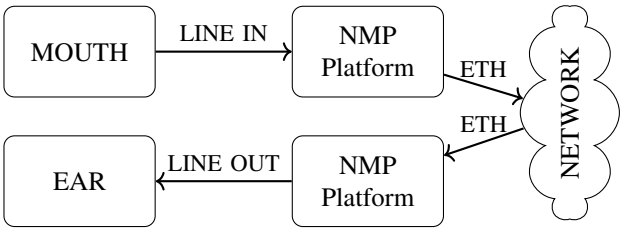

Fig. 1. Audio/data flow for M2E latency

The requirement for low latency audio is due to the necessity for musicians to synchronise their performances such that they are able to play in time with one another and maintain a steady tempo. In order to achieve this synchronicity, OOSE typically needs to be less than 10-25 ms, referred to as the Ensemble Performance Threshold (EPT) [5] or Latency Tolerance Threshold [1]. Certain genres, slower tempos and greater experience can help musicians to maintain steady

[1]https://www.pjrc.com/teensy/

tempo even at higher latencies [6], with no impact on the quality of experience up to 40 ms [7]. However, 25 ms is generally considered the maximum acceptable limit, as it is the threshold beyond which performers begin to perceive delays [1], [5], [8].

A similar method of latency measurement is known as My Mouth to My Ear (MM2ME) and has been presented as a more appropriate measurement for NMP as it aims to replicate the latency between a musician creating a sound and then hearing a remote musician's response to that sound [4]. This is illustrated in Fig.2.

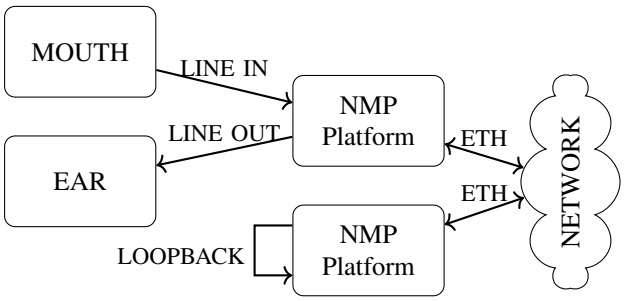

Fig. 2. Audio/data flow for MM2ME latency

To measure latency in a MM2ME test setup, a sound is captured on the LINE IN of the local NMP platform and sent to the remote NMP platform. The remote NMP platform then loops the audio from its LINE OUT to its LINE IN before sending the audio back to the local NMP platform, which finally sends the audio to its own LINE OUT. The delay between capturing a signal on the local LINE IN and producing the same returned signal on the local LINE OUT is the MM2ME latency. In an ideal setup, MM2ME latency will simply be double the OOSE/M2E latency. The main advantage of measuring MM2ME is based on real world NMP setups, where the local and remote devices are potentially several thousand kilometres apart, making it very difficult to accurately measure the delay between generating a sound locally and hearing the sound remotely, as is required by M2E.

When performing together physically, the largest contributor to audio delay is the speed of sound in air as sound waves traverse the air between performers at approximately 343 m/s, or 3 ms per meter. When performing virtually, there are more factors that have a non-negligible impact on latency. These additional latency factors with corresponding latency targets for high-performant NMP devices [9] are outlined in Table I. Significant delays reside in the domain of several milliseconds. Slight delays typically range between approximately 100 µs and a maximum of 1 ms. Values below 100 µs are considered insignificant [9].

B. Audio Quality

Audio quality has been reported as the most important factor contributing to the audience's level of enjoyment of an NMP [10]. The two primary factors that contribute to the quality of an uncompressed digital audio signal are sample rate and bit depth. Together they also determine the bandwidth required for the audio stream and required processing power.

TABLE I
SIGNAL PATH STAGE LATENCY TARGETS

Signal Path Stage	Latency Target
Input Anti-aliasing Filtering	100 µs – 1 ms
Input Blocking & Driver Buffering	2–4 ms
Transmission (Total)	2–4 ms
Sending	100 µs – 1 ms
Routing	2–4 ms
Receiving	100 µs – 1 ms
Jitter Buffering	2–4 ms
Output Blocking & Driver Buffering	2–4 ms
Output Reconstruction Filtering	100 µs – 1 ms

Common audio sample rates for NMP are 44.1 kHz, 48 kHz and occasionally 96 kHz, and common bit depths are 16-bit, 24-bit and 32-bit. In order to provide an NMP experience equivalent to musicians being co-located in the same physical space, a sample rate of 48 kHz and a bit depth of at least 16-bit are generally required [5]. However, a recent study found that reducing sample rates to as low as 8 kHz did not affect the quality of experience for the musicians, with sensitivity varying by instrument [7].

While audio analogue-to-digital and digital-to-analogue converter (ADC and DAC) circuits process audio samples one at a time, the high frequency at which they are produced and consumed cannot be maintained by general purpose computer and operating systems [1], as well as by wide area networks (WAN) like the internet. Therefore, to reduce the processing frequency, the samples are handled in batches, often called buffers.

C. Sample Buffers

Audio samples are transferred between hardware and software applications both locally and over network connections using buffers, though when transferring over a network the buffer is encapsulated inside a packet with additional header data required for network protocols. Although handling each sample individually would result in the lowest latency, this is generally not achievable due to scheduling and interrupt overheads of general purpose operating systems, and the overheads and inconsistent routing within WAN architectures. When a buffer size is too small, audio dropouts can occur as the application is unable to both retrieve and send samples to and from the audio converters at the required frequency. Compromises must be made to ensure a consistent stream of audio, which is easier with larger buffers, while achieving an appropriately low level of latency, which requires smaller buffers. In modern general-purpose operating systems, working with small buffer sizes is challenging since most soundcard drivers support a minimum of 64 samples. Specialised hardware and optimised drivers can reach sizes as low as 32 samples. The time to fill buffers of common sizes and sample rates is shown in Table II.

Although audio converters are configured to operate at predefined sample rates, minor inaccuracies arise because converters are clocked with local oscillators. Consequently,

TABLE II
BUFFER FILL TIME (MS)

Sample Rate	Buffer Size (samples)				
	128	**64**	**32**	**16**	**8**
44.1 kHz	2.90	1.45	0.73	0.36	0.18
48 kHz	2.67	1.33	0.67	0.33	0.17
96 kHz	1.33	0.67	0.33	0.17	0.08

one device may fill its input buffer and play its output buffer slightly faster than the other. This is a prevalent issue and often remains unresolved in current NMP platforms [1]. However, a tunable oscillator circuit design for microcontrollers has been demonstrated to be effective in addressing this problem [11].

D. NMP Hardware

The minimum hardware requirements for an NMP audio platform are:

- Audio converters for audio signal input/output conversion between analogue and digital.
- Ethernet controller for transmitting/receiving digital audio over a network.
- Hardware to run NMP software, managing system configuration and providing connectivity between the audio converters and Ethernet connection for buffering and transferring audio samples, generally featuring at least a CPU, dynamic memory, I/O interfacing and non-volatile memory.

Personal computers (PC) running Windows, macOS, and Linux distributions generally meet these requirements and are the most common platforms for NMP. However, their features and specifications often vastly exceed the bare minimum needed for NMP applications. External audio interfaces or internal sound cards can be added to a PC in order to provide low-latency, high quality audio that is not typically provided by on-board audio converters.

Single board computers (SBC) such as the Raspberry Pi[2] are a suitable lower-cost alternative that also meet these requirements. Though the onboard audio converters of the Raspberry Pi are not suitable for NMP, USB audio interfaces and audio expansion boards that attach to the exposed header pins such as those developed by HiFiBerry[3] can be added which provide ultra-low-latency high quality audio at up to 192 kHz sample rate and minimum sample buffer size of 64 [12] at a total cost of around 125 USD. The BeagleBone Black[4] is also capable of meeting these requirements utilising the audio expansion board Bela, which works at a sample rate of 44.1 kHz with a total cost of around 200 USD.

OpenRemjam[5] is a Teensy-based platform developed for NMP applications. Experiments compared its performance with a Windows PC running standard NMP software, concluding that OpenRemjam was "clearly superior under all test conditions and performs extremely stably" [13]. Using a single sample rate and buffer size, their results showed significantly lower end-to-end latency when compared to the alternatives running on a Windows computer warranting a more in depth investigation into sample rates, buffer sizes and the individual components/processes that contribute to the overall latency for further optimisation.

The covid pandemic saw a surge in the need for and interest around NMP devices, leading to the development of hardware specifically designed for NMP such as the JackTrip Bridge range, but have since ceased production. The Raspberry Pi and HiFiBerry hats are still available for assembling a JackTrip Bridge manually, with relevant benchmarks showing sufficient performance for use as an NMP device [14].

E. NMP Software

NMP software ranges from free command-line executables to cloud-based monthly subscription services offered through web interfaces. A review of 18 NMP software frameworks [1] found that the different technologies varied very little in their implementations [9]. To provide a better understanding of a software NMP architecture, JackTrip[6] was investigated. JackTrip was chosen as it is well established, open-source and has been utilised in a significant number of telematic performances and research works.

JackTrip is an open-source audio streaming software and transmission protocol for low latency streaming developed by the Center for Computer Research in Music and Acoustics (CCRMA) in 2007 [15]. Originally developed for Linux, it has also been ported to Windows, macOS and FreeBSD. The JackTrip protocol uses UDP which is preferable over TCP for low bandwidth and latency-critical applications but does not guarantee packet delivery or correct packet sequencing at the receiving end. To overcome this limitation, JackTrip provides a configurable buffer to mitigate network jitter and adds its own 16 byte header to UDP packets that includes the timestamp and sequence number of the packet [16].

A review of four different low-latency streaming platforms, JackTrip was shown to have the lowest latency but also the highest number of audio glitches [17]. Aretousa, a competitive audio streaming software for NMP [4] was evaluated against JackTrip, highlighting JackTrip's acceptance and ongoing use as state of the art NMP software. However, JackTrip was found to be harder to set up than more recent and more user friendly NMP software, such as Sonobus, but the sound quality was clearer [18].

III. SPECIFICATIONS & IMPLEMENTATION

The Teensy 4.1[7] with an audio shield and Ethernet port was selected for its favourable price/performance ratio, proven reliability in digital musical instruments, and bare-metal programmability within the Arduino framework, eliminating the need for an operating system. It features robust open-source libraries for common hardware components, facilitating future

[2]https://www.raspberrypi.com/
[3]https://www.hifiberry.com/
[4]https://www.beagleboard.org/boards/beaglebone-black
[5]https://github.com/cwerner77/OpenRemjam
[6]https://jacktrip.github.io/jacktrip/
[7]https://www.pjrc.com/store/teensy41.html

development. Widely available and backed by an active online community, the Teensy is a trusted choice in the DIY sector. It has been used in low-latency devices like the open-source TYMPAN hearing aid[8] and in networked spatial audio systems [19]. A JackTrip client has also been implemented for the Teensy in client/server mode [20]. Our aim is not to create a user-friendly NMP platform but to benchmark the hardware components' latency and bandwidth performance as an NMP platform. This foundational work could support researchers and artists in further development, potentially evolving into a complete NMP platform.

A. Teensy 4.1 with Ethernet Chip

Teensy is a range of Arduino compatible development boards which are popular due to their low cost, compact form factor and high performance. With a high-speed 32 bit ARM Cortex-M7 processor running at 600 MHz and an on-board 10/100 Mbps Ethernet PHY transceiver, Teensy 4.1 has been shown to be a capable NMP streaming device [11] in a small form factor, as visible in in Fig. 3. The diagram of the system architecture for this hardware setup can be seen in Fig. 4.

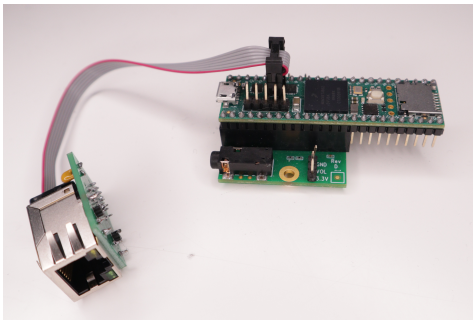

Fig. 3. A Teensy 4.1 development board with audio shield and Ethernet kit.

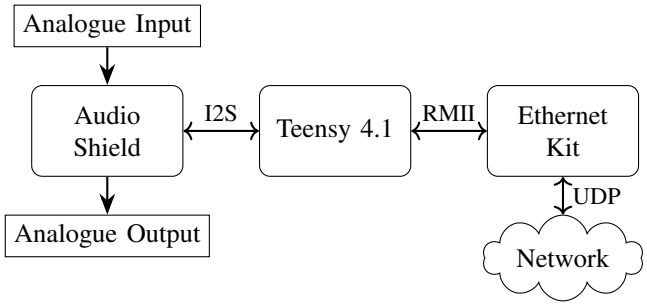

Fig. 4. Architecture and dataflow of the Teensy audio streaming platform.

In addition, the Teensy 4.1 has been proven to be able to successfully address the issue of clock drift in the development of the UNISON audio streaming system [11].

The current total hardware cost for the Teensy 4.1 NMP system is approximately 45.25 USD, including the 31.50 USD for the Teensy 4.1 development board, 9.80 USD for the audio shield and 3.95 USD for the Ethernet kit.

[8]https://shop.tympan.org/

B. Teensy Audio Shield

The Teensy Audio Shield has an NXP SGTL5000 audio codec which is capable of sample rates from 8 kHz to 96 kHz[9]. Our tests and those reported by users on the Teensy forum[10] found that the line output on the SGTL5000 causes an inverting of the signal's phase due to an undocumented inverting preamp which may cause unexpected behaviour in specific applications. However, this can be fixed by using a mixer with a -1 value in the Teensy Audio GUI or by inverting the signal values in code.

C. Libraries

The Ethernet library that is included with the Teeny Arduino IDE installer was found to be very inefficient, so the QNEthernet library[11] is used for tests involving Ethernet connectivity. Similarly, a more efficient library was needed when reading analogue pin values on the Teensy measurement device, so the ADC library[12] is utilised for this purpose. The Teensy audio library[13] provides tools for using the Teensy audio shield's features, including input, output, mixing, filtering, and effects. Developers can adjust global constants for sample rate and buffer size[14]. The Teensy Audio GUI is a browser-based tool for configuring and visualising internal audio routing, offering control over gain, patching, mixing, and documentation of the library's components.

IV. Performance Evaluation

We selected a total of ten tests to evaluate the performance of the Teensy Audio Streaming Platform (TASP). These tests were designed to evaluate the latency and bandwidth against the audio quality of the TASP. Additionally, we conducted a WAN MM2ME test and an integration test using JackTrip. In the following subsections, we present the methodology for each test, noting that the specific setup may vary across them. The latency targets for the processes are presented in Table I for reference. The code developed for these tests and details of their implementation are available on GitHub[15]. Unless specified, each test was automatically repeated 100 times at 1 second intervals, and the average is presented in the results.

A. Methodology

For tests requiring the measurement of the latency between sending and receiving a signal, a second Teensy 4.1 was used as the measurement device. We leverage the internal timer of the measurement Teensy to determine the latency. The measurement device produces a signal by changing output pin 33 from LOW (0 V) to HIGH (3.3 V) which is attached to the TASP's LINE IN. The timer measures the elapsed time until the HIGH level propagates to input pin 24, which is attached to

[9]https://www.nxp.com/docs/en/data-sheet/SGTL5000.pdf
[10]https://forum.pjrc.com/index.php?threads/audio-shield-inverts-the-input.50752
[11]https://github.com/ssilverman/QNEthernet
[12]https://github.com/pedvide/ADC
[13]https://www.pjrc.com/teensy/td_libs_Audio.html
[14]https://www.pjrc.com/teensy/td_libs_AudioNewObjects.html
[15]https://github.com/leighmurray/Teensy4.1LatencyBenchmarks

the TASP's LINE OUT. Measured latency values are reported via USB serial connection.

1) I2S Passthrough: This test determines the latency of the SGTL5000 codec and presents the minimal latency achievable for the device when inputting and outputting audio. This configuration produces results for the following signal path stages; input anti-aliasing filtering, output reconstruction filtering, input sample blocking and output sample blocking. Since this configuration has no application buffer, this configuration could be used for a direct monitoring loop enabling the local musician to hear their own audio. The physical setup for this test configuration is illustrated in Fig. 5. The Teensy Audio GUI was used for routing the audio data within the TASP.

Fig. 6 shows that the TASP is receiving audio data from the inputDeviceI2S, which is the LINE IN of the audio shield. The left channel of this device is routed directly to the left channel of the outputDeviceI2S, which is the LINE OUT of the same audio shield.

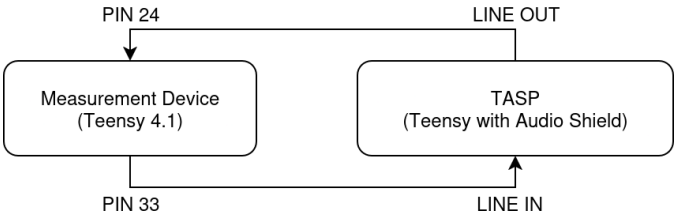

Fig. 5. Signal wiring between measuring device and TASP.

Fig. 6. Teensy Audio GUI configuration for I2S passthrough

2) Buffer Passthrough: This test determines the latency associated with introducing an application audio buffer. This buffer makes the audio data available to the application code. The methodology for this test is based on the same approach described in Section IV-A1 with the following modifications. As shown in Fig. 7, the audio data coming from the audio shield via inputDeviceI2S (audio shield LINE IN) is stored in an input buffer. The application running on the TASP simply copies the buffer samples from inputBuffer to outputBuffer. The outputBuffer is then sent to the outputDeviceI2S (audio shield LINE OUT).

Fig. 7. Teensy Audio GUI Configuration for I2S passthrough with application buffer

3) Ethernet Latency - One Way: This test determines the latency associated with transmitting audio buffers between two TASP devices over the devices' Ethernet ports. The measurement device transmitted data of the appropriate buffer sizes via UDP to the TASP, and the TASP would signal to the measurement device that it had received the packet. This test determines the latency of the sending and receiving stages.

For this test, two identical Teensy 4.1 boards are used, with the respective Ethernet ports directly connected via a LAN patch cable. The device responsible for sending the data and measuring the latency is referred to as the measurement device, while the device responsible for receiving the data and signalling to the measurement device that the data had been received is referred to as TASP.

The measurement device started a timer before transmitting the packet to the TASP. The measurement device then waited for pin 26 to be set to HIGH by the TASP, signalling the UDP packet had been received. The configuration of this test setup is shown in Fig. 8.

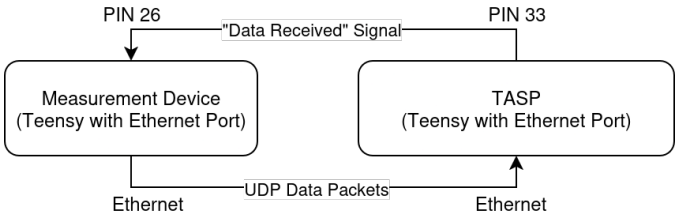

Fig. 8. Data flow for the one way Ethernet latency test

4) Ethernet Latency - Loopback: This test determines the latency involved in Ethernet data transfer and application buffers. While Section IV-A3 measured transmission time, this loopback required that the received data be read into a buffer within the application before being transmitted back to the measurement device via UDP. The latency is thus not only due to transmission of the data but also to processing the audio buffer in the application. The methodology for this test was the same as in Section IV-A3 with the following changes. When the data was received by the TASP it was copied into an application buffer and retransmitted back to the measurement device. When the measurement device registered this data as available, it stopped the measurement timer. This configuration is shown in Fig. 9.

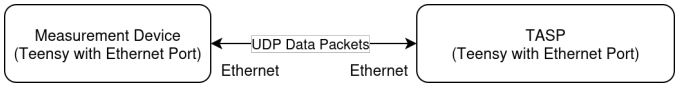

Fig. 9. Data flow for the loopback Ethernet latency test and UDP bandwidth – two way test

5) UDP Bandwidth - Unidirectional: This test determines the effective bandwidth of the TASP's Ethernet connection. The physical setup for this test was the same as described in Section IV-A3. The measurement device continually sent UDP data packets to the TASP. This dataflow is illustrated in Fig. 10.

The TASP would keep track of how many bytes had been received and reported the measured value via serial USB

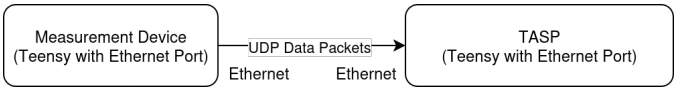

Fig. 10. Data flow for the UDP bandwidth – one way test

connection. The count is reported and reset every second, resulting in a measurement equivalent to bytes per second. This test is conducted for ten seconds, with the resulting average reported in the following results.

6) UDP Bandwidth - Bidirectional: This test determines the bandwidth capabilities of the TASP when sending and receiving data simultaneously. The methodology is the same as that outlined in the previous chapter, except both the measurement device and TASP continually sent UDP packets whilst receiving UDP packets from the other device. This configuration and dataflow can be seen in Fig. 9.

7) MM2ME LAN latency - one channel: This test determines the latency of the device in an NMP setup. The results of this test are compared against the NMP required EPT of 25 ms. The further the latency results are below this threshold, the more time is available for routing and jitter buffering on WANs, resulting in higher quality NMP over larger distances. In this setup, two TASPs were used, referred to as the local TASP and remote TASP. A third Teensy 4.1 was introduced as the measurement device. The local TASP captures audio data and sends it to a remote TASP which performs an audio loopback to itself before sending the audio data back to the local TASP. All the signal path stages in Table I are present in this test with the exception of routing and jitter buffering. The physical connections and data flow can be seen in Fig. 11.

Fig. 11. Data flow for the MM2ME - LAN latency tests

The two TASPs run an identical program. Audio captured on their LINE IN was sent via their Ethernet interface, and data received on their Ethernet interface was transferred to their LINE OUT. The output pin of the measurement device is connected to the LINE IN of the local TASP and the input pin of the measurement device is connected to the LINE OUT of the local TASP.

8) MM2ME LAN latency - two channels: This test determines the additional latency introduced by adding a second audio channel to an NMP, allowing for two mono audio sources (eg. vocals and guitar) or one stereo audio source. The test is identical to the previous single channel test except two audio channels are now used. Similarly, the results of this test can be compared against the EPT for NMPs that require two channel audio.

9) MM2ME WAN latency: The purpose of this test was to take the TASP "out of the lab" and test it in a real-world WAN configuration. This test used identical code to the previous two tests, except the devices were configured to send their audio data to a server which forwarded the UDP packets to the other TASP. The two TASPs were located in a residential apartment in Vällingby, Stockholm, Sweden while the UDP Forwarding Server was hosted at the University of Oslo, Norway. The server in Oslo kept track of all client devices. Whenever it received UDP data packets it would forward them to all devices it had previously received UDP data from, with the exception of the device where the packet originated. The UDP server code can be found on GitHub[16] and the hardware setup and data flow is shown in Fig. 12.

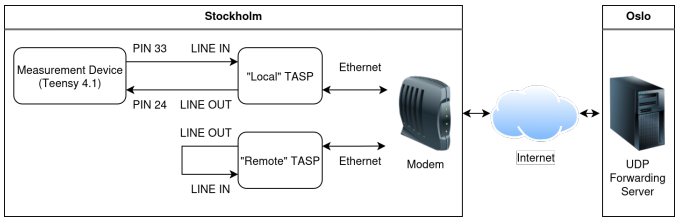

Fig. 12. Data flow for the MM2ME - WAN latency test

The signal path is:

1) Measurement Device Pin 33
2) Local TASP Audio Input
3) Local TASP Ethernet Transmit (Stockholm)
4) UDP Forwarding Server Receive/Transmit (Oslo)
5) Remote TASP Ethernet Receive (Stockholm)
6) Remote TASP Audio Output
7) Remote TASP Audio Input (Loopback)
8) Remote TASP Ethernet Transmit (Stockholm)
9) UDP Forwarding Server Receive/Transmit (Oslo)
10) Local TASP Ethernet Receive (Stockholm)
11) Local TASP Audio Output
12) Measurement Device Pin 24

Oslo and Stockholm are approximately 415 km apart, considering the distance in a straight line. The signal is transmitted back and forth between the two devices in Stockholm via a server in Oslo, resulting in a total of two round trips from Stockholm to Oslo due to the loopback required for MM2ME, for a total distance of at least 1,660 km. Based on straight line distance, this would be comparable to a NMP with a separation of 830 km. A ping test was performed from the network in Stockholm to the forwarding server in Oslo which showed a round trip time (RTT) of 8 ms. This allows for a maximum of 17 ms of additional latencies introduced by the NMP platforms to remain within the 25 ms EPT for OOSE. Since the signal completes this entire round-trip twice for MM2ME, that results in a total of 16 ms of network transmission time on the WAN network utilised for this test and the results are then halved to determine the OOSE values.

10) JackTrip Integration: This test determines the TASP's ability to function as a NMP device by integrating it with a proven NMP technology. This setup consisted of a Windows PC running JackTrip, a TASP and a Linux based UDP forwarding server. The Windows PC and TASP were located

[16]https://github.com/leighmurray/udpforwardergo

in a residential apartment in Vällingby, Stockholm, Sweden while the UDP Forwarding Server was hosted at the University of Oslo, Norway. The Windows PC used an external MOTU M4[17] soundcard running ASIO drivers at 48 kHz with a buffer size of 64. The soundcard was connected to a pair of studio monitor speakers. Voicemeeter Potato[18] software was used to simultaneously route the PC's audio to the MOTU M4 and the JackTrip software running on the PC. The TASP was connected to amplified speakers using the LINE OUT pins. The TASP code was modified to handle the additional JackTrip packet header. The Windows PC and TASP were connected using JackTrip's Peer to Peer mode and implementation details can be found on GitHub[15]. The physical connections and dataflow diagram can be seen in Fig. 13.

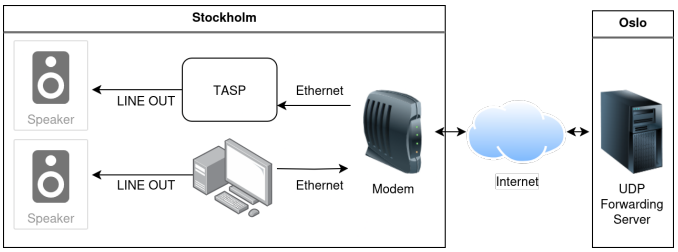

Fig. 13. Physical connections and dataflow diagram for the JackTrip test

Although Fig. 13 only shows the dataflow of the audio stream emanating from the Windows PC, there was a second audio stream containing the LINE IN audio from the TASP that travelled to the Windows PC via the UDP forwarding server as there would be in a regular NMP setup. However, this stream contained no audio during this test. The Windows PC in Stockholm played an audio file that could be heard locally through a speaker connected to the PC's external soundcard. The PC also routed the audio stream to the JackTrip software which sent the stream to the UDP Forwarding Server in Oslo. The UDP forwarding server forwarded the audio data to the TASP in Stockholm where it was played through the external amplified speaker connected to the TASP's LINE OUT.

B. Results

1) I2S Passthrough: Table III and Fig. 14 show the total time between creating the signal on the input of the device and receiving the signal on the output of the device. These results illustrate the TASP's ability to support the 32 sample buffer size used by high-end NMP platforms in addition to supporting smaller buffer sizes of 16 and 8. Also, the TASP is able to operate efficiently at the standard sampling rates of 44.1 kHz, 48 kHz and 96 kHz.

Table IV shows the additional time the TASP introduces in the passthrough of the audio by removing the default durations for input and output blocking delays – equivalent to two times the buffer audio length shown in Table II – from the results in Table III. As illustrated, the I2S latency is only influenced by the sampling rate and remains very consistent between buffer size. These latency values, which include any input and output filtering, as well as possible additional delays from input and output blocking processes are well within the target ranges.

[17]https://motu.com/en-us/products/m-series/m4/
[18]https://vb-audio.com/Voicemeeter/potato.htm

TABLE III
I2S-PASSTHROUGH LATENCY (MS)

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	6.37	3.45	2.00	1.28	0.92
48 kHz	5.84	3.17	1.85	1.17	0.83
96 kHz	2.92	1.58	0.92	0.58	0.42

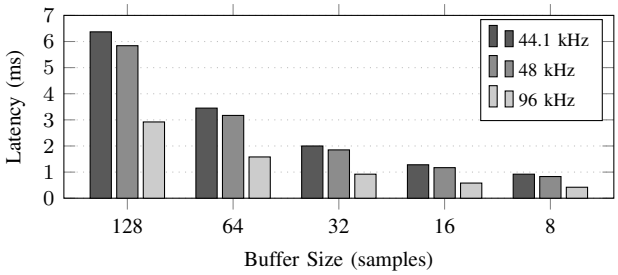

Fig. 14. I2S-passthrough latency

TABLE IV
I2S OVERHEAD TIME (MS)

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	0.56	0.54	0.55	0.55	0.56
48 kHz	0.51	0.50	0.51	0.51	0.50
96 kHz	0.26	0.25	0.25	0.25	0.25

2) Buffer Passthrough: Table V and Fig 15 show that at higher buffer sizes the latency values have increased by over 3 ms compared to the I2S passthrough latency results. To determine the exact increase in latency caused by the application buffer, the default durations for input and output blocking delays – equivalent to two times the audio buffer length shown in Table II – are again removed from the results shown in Table V. The latencies with these blocking delays removed are shown in Table VI.

TABLE V
BUFFER PASSTHROUGH LATENCY (MS)

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	9.38	5.01	2.84	1.75	1.12
48 kHz	9.30	4.65	2.65	1.65	1.03
96 kHz	4.43	2.32	1.31	0.82	0.51

Unlike the I2S passthrough overhead latency, the buffer passthrough overhead latency is influenced by the buffer size. Table VII removes the overhead that was determined in the I2S overhead test as shown in Table IV in order to provide a clearer representation of the additional time associated with the use of application buffers in contrast to direct I2S passthrough.

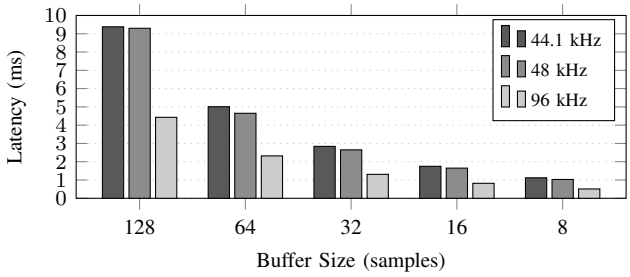

Fig. 15. Buffer passthrough latency

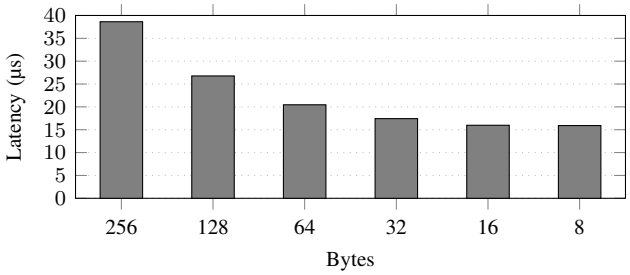

Fig. 16. Unidirectional Ethernet latency for decreasing packet sizes.

TABLE VI
DEVICE OVERHEAD TIME (MS) FOR BUFFER PASSTHROUGH TEST

Sample Rate	Buffer Size (samples)				
	128	**64**	**32**	**16**	**8**
44.1 kHz	3.57	2.11	1.38	1.02	0.75
48 kHz	3.97	1.98	1.32	0.98	0.70
96 kHz	1.76	0.98	0.65	0.48	0.35

As can be seen by comparing Table VII to the default buffer delays in Table II, the introduction of an application buffer has resulted in additional delay approximately equal to the duration of one audio buffer, which is to be expected.

TABLE VII
LATENCY ADDED BY SOFTWARE BUFFER (MS)

Sample Rate	Buffer Size (samples)				
	128	**64**	**32**	**16**	**8**
44.1 kHz	3.01	1.56	0.84	0.47	0.20
48 kHz	3.46	1.49	0.80	0.48	0.21
96 kHz	1.50	0.73	0.39	0.23	0.10

3) Ethernet Latency – Unidirectional: It is important to note that unlike the previous graphs, the results shown in Table VIII and Fig. 16 are in microseconds (µs). The greatest improvement in latency is between 256 and 128 buffer sizes, representing an 11 microsecond reduction, and from there the reduction is halved as the buffer size is halved, resulting in only 1 µs improvements between 32, 16 and 8 byte buffers. This is likely a result of the additional frame headers for the UDP packet and lower OSI layers that are of a consistent size. These processes of "sending" and "receiving" in Table I are categorised as slight delays, equating to delay values between approximately 100 µs and 1 ms. However, these results are all well below 100 µs, which instead classifies them as insignificant delay for the TASP.

TABLE VIII
ETHERNET UNIDIRECTIONAL LATENCY (µS) BY PACKET SIZE (BYTES)

Bytes	256	128	64	32	16	8
Latency (µs)	39	27	20	17	16	16

4) Ethernet Latency – roundtrip: The results in Table IX and Fig. 17 are in microseconds. The loopback latency or RTT for the ethernet packets is double that of the one way latency, within 1 µs. These results are particularly noteworthy, as even after loopback, which involves two sending and two receiving processes, the delay is below 100 µs. This qualifies as insignificant delay for the TASP and marks a possible improvement over existing NMP platforms.

TABLE IX
ETHERNET ROUNDTRIP LATENCY (µS) BY PACKET SIZE

Bytes	256	128	64	32	16	8
Latency (µs)	78	53	41	35	32	31

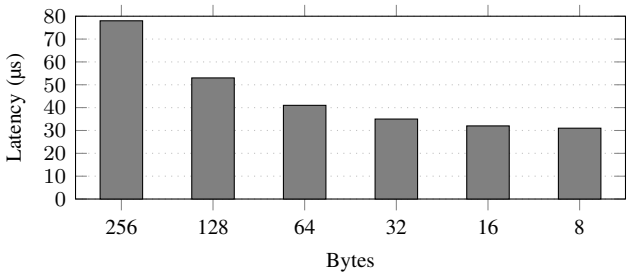

Fig. 17. Roundtrip Ethernet latency for decreasing packet sizes.

5) UDP Bandwidth – Unidirectional: The result in Table X shows the TASP efficiently utilises the bandwidth of its 10/100 Mbps Ethernet chip. A 48 kHz/16 bit audio stream has a bitrate of 0.768 Mbps which is 0.8 percent of the device's available bandwidth, so the bandwidth is sufficient for sending or receiving over 100 audio streams at 48 kHz/16 bit.

TABLE X
UDP ONE-WAY BANDWIDTH PERFORMANCE

Device	**Receive Bandwidth (Mbps)**
TASP 1	95.80

6) UDP Bandwidth – Bidirectional: The results in Table XI show that there is very little negative impact on the device's network bandwidth when both sending and receiving in comparison to only sending or receiving.

7) MM2ME LAN – One Channel: The results shown in Table XII and Fig. 18 are the total latency values for MM2ME so they need to be halved before comparing to the EPT of 25 ms. These halved values, equivalent to the Over-all One-way Source-to-Ear (OOSE) are shown in Table XIII.

TABLE XI
UDP TWO-WAY BANDWIDTH PERFORMANCE

Device	Receive Bandwidth (Mbps)
TASP 1	95.70
TASP 2	93.82

TABLE XII
MM2ME LATENCY (MS) – ONE CHANNEL

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	15.93	8.70	5.01	3.16	2.15
48 kHz	15.05	8.32	4.99	2.81	2.06
96 kHz	7.66	4.44	2.49	1.42	1.02

The results show excellent performance from the TASP, with OOSE latencies as low as half a millisecond at 96 kHz with an 8 sample buffer size. At the NMP standard sample rate of 48 kHz and 32 sample buffer size, the measured OOSE latency is 2.50 ms, allowing for more than 22 ms of routing and jitter buffering. To determine the delay introduced by the TASP devices, we subtracted the time needed to fill the input and output buffers from the results presented in Table XIII and the adjusted results are provided in Table XIV. At the standard NMP sample rate of 48 kHz and above, the TASP introduces latencies of 2.19 ms or less.

8) MM2ME LAN – Two Channels: To determine the latency introduced by including a second audio channel, the resulting latencies from the single channel test shown in Table XII have been removed from the dual channel test results in Table XV and Fig. 19. These results are detailed in Table XVI.

The additional latency introduced by using a second audio channel, transitioning from mono to stereo, is generally well below 1 ms. The results indicate anomalies when utilising a buffer size of 16, as well as with sample rate and buffer size pairs of 44.1 kHz/32 and 96 kHz/64. The cause of these unexpectedly high values remains unclear at present. Some

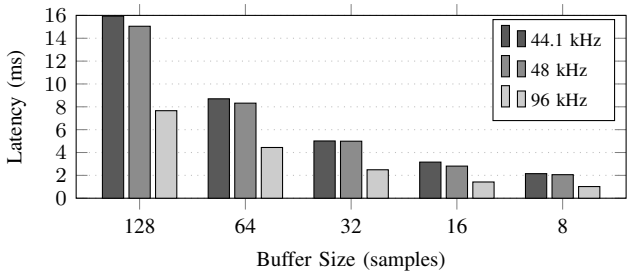

Fig. 18. MM2ME (one channel)

TABLE XIII
OOSE LATENCY (MS)

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	7.97	4.35	2.51	1.58	1.07
48 kHz	7.53	4.16	2.50	1.40	1.03
96 kHz	3.83	2.22	1.24	0.71	0.51

TABLE XIV
LATENCY INTRODUCED BY TASP IN OOSE (MS)

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	2.16	1.45	1.05	0.85	0.71
48 kHz	2.19	1.49	1.16	0.74	0.69
96 kHz	1.16	0.89	0.58	0.38	0.34

TABLE XV
MM2ME LATENCY (MS) – TWO CHANNEL

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	16.02	8.74	6.02	3.65	2.41
48 kHz	15.22	8.38	5.01	3.40	2.22
96 kHz	7.73	5.16	2.90	1.72	1.13

Teensy audio library features require minimum buffer sizes, so further investigation is necessary to determine if such minima, or other libraries, are impacting particular sample rate and buffer size latencies.

9) MM2ME WAN: The results shown in Table XVII and Fig. 20 are MM2ME so they need to be halved before comparing to the EPT of 25 ms. These values are shown in Table XVIII. These results are considerably below the EPT of 25 ms, providing ample time for the introduction of larger sample buffers to mitigate eventual network jitter while remaining below the EPT.

10) JackTrip Integration: The TASP functioned successfully alongside well established NMP systems. From informal listening it was not possible to distinguish the playback of two separate audio streams, after volumes were adjusted to match. However some audio glitches were noticed. Assuming the OOSE latency was consistent with previous tests of around 11 ms, larger sample buffers could have been used to minimise such glitches.

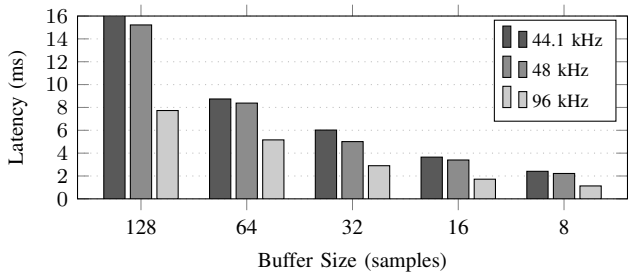

Fig. 19. MM2ME latency (two channel)

TABLE XVI
LATENCY INTRODUCED BY SECOND AUDIO CHANNEL (MS)

Sample Rate	Buffer Size (samples)				
	128	64	32	16	8
44.1 kHz	0.08	0.04	1.01	0.49	0.26
48 kHz	0.17	0.06	0.02	0.59	0.16
96 kHz	0.07	0.72	0.41	0.29	0.11

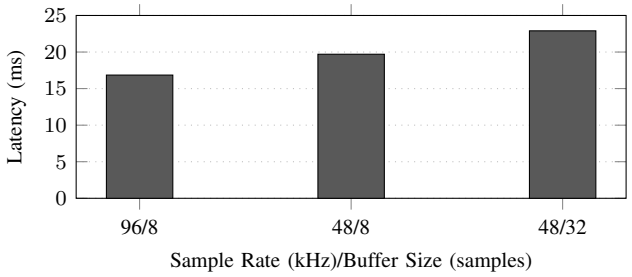

Fig. 20. MM2ME latency measurement results of the TASP over WAN between Stockholm and Oslo

TABLE XVII
MM2ME WAN LATENCY (MS)

Sample Rate	Buffer Size (samples)	
	8	32
96 kHz	16.85	–
48 kHz	19.70	22.90

V. Conclusion & Future Work

This paper explored the Teensy 4.1 ecosystem as a cost-effective NMP endpoint, utilising hardware expansion boards and open-source development environments that are easy to use and procure.

To enhance the limited low-cost NMP options, a TASP was prototyped and characterised. The evaluation benchmarked the Teensy 4.1 with its Ethernet Kit and Audio Shield, assessing performance across various latency metrics, including I2S passthrough, buffer passthrough, Ethernet transmission, UDP bandwidth utilisation, and MM2ME latency over LAN and WAN. Compatibility with established NMP software like JackTrip was also evaluated.

The Teensy 4.1-based TASP achieved OOSE latencies as low as 0.51 ms at 96 kHz with an 8-sample buffer and 2.50 ms at 48 kHz with a 32-sample buffer, well below the EPT of 10-25 ms. It supports small buffer sizes (8 and 16 samples), which are challenging for general-purpose operating systems, and efficiently uses Ethernet bandwidth with minimal impact during simultaneous data transmission. This performance is reached at a significantly lower cost than other embedded NMP platforms. While JackTrip integration suggested larger buffers might reduce glitches, overall performance confirms the Teensy 4.1's viability as a low-cost NMP platform.

The current prototype lacks hardware and software integrations necessary for a standalone NMP system, such as display and input devices. These additions would enable adjustments for buffer size, sample rate, number of channels, IP address, audio streaming control, and performance monitoring. These enhancements could contribute to OpenRemjam, unifying efforts towards a low-latency audio streaming platform based on Teensy 4.1.

TABLE XVIII
OOSE WAN LATENCY (MS)

Sample Rate	Buffer Size (samples)	
	8	32
96 kHz	8.43	–
48 kHz	9.85	11.45

References

[1] C. Rottondi, C. Chafe, C. Allocchio, and A. Sarti, "An overview on networked music performance technologies," *IEEE Access*, vol. 4, pp. 8823–8843, 2016.

[2] M. Rofe and F. Reuben, "Telematic performance and the challenge of latency," *Journal of Music, Technology and Education*, vol. 10, no. 2-3, pp. 167–183, 2017.

[3] K. Tsioutas and G. Xylomenos, "On the impact of audio characteristics to the quality of musicians' experience in network music performance," *Journal of the Audio Engineering Society*, pp. 914–923, 2021.

[4] K. Tsioutas, G. Xylomenos, and I. Doumanis, "Aretousa: A competitive audio streaming software for network music performance," in *Proceedings of the 146th Audio Engineering Society Convention*, Dublin, Ireland, 2019.

[5] A. Carôt, U. Kramer, and G. Schuller, "Network music performance (nmp) in narrow band networks," in *Proceedings of the International Computer Music Conference (ICMC)*, 2006.

[6] N. Bouillot, "njam user experiments enabling remote musical interaction from milliseconds to seconds," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, New York City, NY, 2007, pp. 142–147.

[7] T. Tsioutas, C. Boukis, K. Pastiadis, and G. Papadelis, "The impact of audio quality on the quality of experience in networked music performance," *Future Internet*, vol. 17, no. 3, p. 337, 2025. [Online]. Available: https://www.mdpi.com/1999-5903/17/3/337

[8] A. Carôt, "Musical telepresence: A comprehensive analysis towards new cognitive and technical approaches," Ph.D. dissertation, Universität zu Lübeck, 2009.

[9] A. Carôt, C. Hoene, H. Busse, and C. Kuhr, "Results of the fast-music project—five contributions to the domain of distributed music," *IEEE Access*, vol. 8, pp. 47 925–47 951, 2020.

[10] G. Davies, "The effectiveness of lola (low latency) audiovisual streaming technology for distributed music practice," M. Research diss., Edinburgh Napier University, 2015.

[11] C. Werner and R. Kraneis, "Unison: A novel system for ultra-low latency audio streaming over the internet," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021.

[12] L. Vignati, S. Zambon, and L. Turchet, "A comparison of real-time linux-based architectures for embedded musical applications," *Journal of the Audio Engineering Society*, vol. 70, no. 1/2, pp. 83–93, 2022. [Online]. Available: https://doi.org/10.17743/jaes.2021.0052

[13] R. Moscatelli, K. Stahel, R. Kraneis, and C. Werner, "Why real-time matters: Performance evaluation of recent ultra-low latency audio communication systems," in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, 2024, pp. 77–83.

[14] S. Ubik and J. Melnikov, "High-quality audio network transmissions with raspberry pi," in *2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2022, pp. 1–3.

[15] F. Lopez-Lezcano and C. Wilkerson, "Ccrma studio report," Stanford University, Tech. Rep., 2007.

[16] C. Chafe and J.-P. Caceres, "Jacktrip: Under the hood of an engine for network audio," in *Proc. Int. Comput. Music Conf. (ICMC)*. Montreal, Canada: Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, 2009.

[17] N. Bouillot and J. R. Cooperstock, "Challenges and performance of high-fidelity audio streaming for interactive performances," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2009, pp. 135–140.

[18] D. Rossetti and C. C. Bomfim, "Live electronics, audiovisual compositions, and telematic performance: Collaborations during the pandemic," *Journal of Network Music and Arts*, vol. 3, no. 1, 2021.

[19] T. A. Rushton, R. Michon, S. Serafin, T. Risset, and S. Letz, "Networked microcontrollers for accessible, distributed spatial audio," *Frontiers in Virtual Reality*, vol. 5, 2024.

[20] T. A. Rushton, R. Michon, and S. Letz, "A microcontroller-based network client towards distributed spatial audio," in *Proceedings of the Sound and Music Computing Conference (SMC 2023)*. Stockholm, Sweden: Sound and Music Computing Network, 2023, pp. 170–177.