TECNIACUSTICA MURCIA · 2014

45º CONGRESO ESPAÑOL DE ACÚSTICA
8º CONGRESO IBÉRICO DE ACÚSTICA
EUROPEAN SYMPOSIUM ON SMART CITIES AND ENVIRONMENTAL ACOUSTICS

BINAURAL SYNTHESIS AND REPRODUCTION INCLUDING CROSSTALK CANCELLATION ON A SYSTEM ON A CHIP

PACS: 43.60.+d

Aspöck, Lukas[1]; Pelzer, Sönke[1]; Vorländer, Michael[1]
[1]Institute of Technical Acoustics, RWTH Aachen University, Kopernikusstr. 5, 52074 Aachen, Germany, Phone: +49 241 80 97911; Fax: +49 241 80 92214
las@akustik.rwth-aachen.de, spe@akustik.rwth-aachen.de, mvo@akustik.rwth-aachen.de

ABSTRACT

Credit-card-sized single-board computers with a system on a chip (SoC) became efficient and affordable in recent years increasing their attractiveness for end users. Linux-based operating systems and different interfaces such as USB, HDMI and Ethernet enable the application of audio signal processing. Based on block-based fast convolution, filter operations such as loudspeaker equalization or binaural synthesis can be realized in an efficient way. This article presents and discusses the implementation of loudspeaker equalization as well as of binaural reproduction via two loudspeakers including crosstalk cancellation on small single-board-computers.

1. INTRODUCTION

Nowadays it is nearly impossible to listen to reproduced audio that has not passed through digital conversion. Digital signal processing (DSP) has become indispensable in the domain of recording, processing, storing and transmitting audio signals. Digital recording enabled multitrack recording, digital storage and led to the invention of MP3. However, digital filtering is still mainly being used by experts and has not finally arrived at our homes and everyday life yet.
With the introduction of small single-board computers this might change in the next couple of years. Embedded systems realized by ARM architecture [1] gained great popularity due to low prices, high energy efficiency, flexibility and their space saving dimensions.
Besides their application in smartphones and tablets, ARM based single board computers are emerging, enabling also their application as sophisticated audio signal processors. The computation power is high enough to apply digital filters that are so far only found on expensive professional devices, by performing so-called convolutions. This computational complex method is the only mechanism which can perform arbitrarily complicated filtering and equalization in real-time [2]. For example by applying appropriate equalization filters, a cheap loudspeaker can possibly be improved to sound just like a much more expensive competitor. The reason why such techniques are not yet wide-spread are mainly because DSPs are still specialized hardware, expensive and operable only by professionals.
This paper discusses the applicability of single-board computers for audio signal processing, including high level operations, such as FIR-filtering, binaural synthesis and crosstalk cancellation for binaural reproduction with loudspeakers. The hardware specifications of different single-board computers as well as the software implementation of a convolution engine on ARM-architecture processor systems are described. Finally, some performance benchmarks are presented.

2. SINGLE-BOARD COMPUTERS

Single-board Computers are currently becoming popular, not only for technical experts, but also for home users with less experience in working with such devices. The availability of various preconfigured operating systems and software packages makes working with single-board computers more convenient for these user groups. Single-Board Computers usually include a system-on-a-chip, also called SoC, which contain a general purpose processor, typically based

on ARM-architectures. These CPUs are usually not as powerful as x86-based CPUs running in PCs and laptops. However, also multicore solutions are available, which offer the possibility for running computationally demanding multimedia applications or a large number of complex calculations in real-time. To support these applications, a floating-point unit (FPU) such as VFPv2 or NEON is integrated for acceleration, e.g. for graphic and signal processing [3].
In this paper, two different single-board computers are described: the popular Raspberry Pi and the more powerful chipset Odroid X2. Due to the popularity of the Raspberry pi, the description of the implementation is focused on this device. It was chosen because of the low cost and the diverse and considerable experience of the user community. However, there are other boards available which are also suitable for audio signal processing, e.g., *Cubietech*[1], *Beagle*[2] or *Arduino*[3] boards.

2.1 HARDWARE SPECIFICATIONS

If a SoC fulfills the specifications for performing the planned tasks, it might still not be a applicable solution because the board does not contain the required hardware interfaces. The Raspberry Pi board offers common hardware interfaces such as an HDMI port, an Ethernet socket as well as two USB ports. It also features *General Purpose Input/Output* (GPIO) pins as well as an I²S interface (Inter-ICSound), which offers the possibility to directly connect digital-to-analog converters (DACs) to the SoC. Table 1 shows the boards and some hardware specifications of the Raspberry Pi and the Odroid X2.

	Raspberry Pi	**Odroid X2**
		
SoC type	BCM2835 ARM11x	Exynos441 Cortex-A9
CPU rate	Up to 1 GHz	4 Cores @ 1.7 GHz
Memory	512 MB	2 GB
FPU	VFPv2	NEON
USB ports	2	6
audio	Analog output / HDMI	Analog Input + Output / HDMI
I²S	available	Not avaialbe
Price	~ 35.00 $	~ 130.00 $

Table 1: Two examples for ARM based single-board computers

In comparison the OdroidX2 board is a more powerful device due to its quadcore CPU. Similar hardware is found in in current smartphones or tablet pcs. A disadvantage of the Odroid X2 board is the missing support of I²S DACs. Although there are expansion boards which enable users to connect devices communicating over various protocols, there are no off-the-shelf solutions for connecting I²S sound cards to the Odroid X2. Thus, if the onboard audio chip of the Odroid X2 is not sufficient, a USB sound card has to be connected to the device.

2.2 SOUND CARDS

Because on the Raspberry Pi, the onboard audio chip is not suitable for reasonable audio reproduction and no sound input channel is available, additional hardware is required to process

[1] http://cubieboard.org/
[2] http://beagleboard.org/
[3] http://www.arduino.cc/

audio. Multiple different sound cards have been examined and tested on both boards. Even low cost USB sound cards show an adequate SNR (of around 90 dB), good latency behavior and support at least stereo input and output channels. One example of such a sound card is the *Behringer UCA 202*, containing a *Texas Instrument PCM2900* DAC.
Another attractive solution for connecting high quality sound chips is the I²S protocol. Here, the DAC is directly connected to the hardware infrastructure of the board and no intermediate bus system can interfere or limit the data communication. This keeps the latency low (~5 ms less in comparison to USB sound cards, see Section 5) and avoids interrupt problems. Unfortunately only the Raspberry Pi offers the possibility to connect I²S sound chips. Examples for such DACs are the *Wolfson Audio board*, the *HifiBerry DAC* or the *Audio Codec Shield* (used for the performance analysis). However these sound cards are often adjusted to certain system and hardware configurations and might not work for certain kernels of the operation systems.

2.3 OPERATING SYSTEM

As most single-board computers use ARM-based systems-on-a-chip, Linux distributions are commonly used as the operating system. For the Raspberry Pi, *Raspian,* an adjusted version of the *Debian* distribution, was developed. To adapt to the special needs of the system, e.g., for audio processing, even more specialized distributions exist. *Rasbmc*[4]*, for example,* turns the Raspberry Pi into a media player.
Linux operating systems in general are not suitable for real-time processing as it is designed for high throughput [4]. In common operating systems, the scheduling algorithm of the kernel assigns certain time slices to each running tasks. More important kernel tasks and interrupts of other devices (e.g., reading from the SD-card) might interrupt normal tasks (e.g., the convolution or the audio buffering). This can lead to audio dropouts and glitches in non-modified operation systems, especially if small buffer sizes are used. This problem is solved if pre-emptible kernels are used: Non-kernel processes with assigned real-time priority are allowed to interrupt kernel tasks [5].
For the presented work, the *Rasbian* based distribution *Volumio*[5] was chosen for the Rasberry Pi. It contains a web interface, enabling the user to control music playback and audio processing from another device, a smartphone or a laptop. Several modifications to the operating system have been made to optimize performance for real-time convolution and audio processing. Next to the replacement of the existing web interface (see Section 3.3), the kernel of *Volumio* was exchanged by a real-time kernel (*Linux volumio 3.12.20-rt30_wab #11 PRREEMPT RT*). Additionally several configurations were changed to enable the system to fulfill the high requirements of a real-time system.

3. AUDIO SIGNAL PROCESSING ON A SOC

ALSA (Advanced Linux Sound Architecture) provides an interface to access sound cards on Linux systems. It allows direct audio playback and recording but also the exchange of audio data between applications or higher audio layers. An audio ring buffer, containing multiple periods of frames, can be accessed to send or receive data from the sound card. Various higher level sound libraries exist to facilitate sound processing via the ALSA interface, e.g., PortAudio or JACK (JACK Audio Connection Kit). An overview of Linux audio layers is given in Figure 1. In this project, JACK is used because of its low latency processing by directly mapping available buffers from the audio hardware into the process memory [6].

[4] http://www.raspbmc.com/
[5] http://volumio.org/

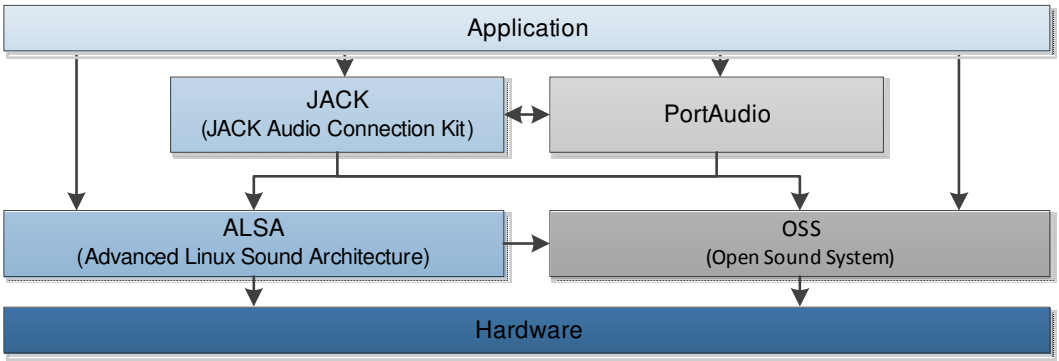

Figure 1: Overview of Linux audio layers. For the presented project, the application uses a JACK server and the ALSA interface. The PortAudio library and OSS interface represent examples for alternatives of unix-based sound processing.

FIR-Filtering was included into the processing chain using a streaming real-time convolution core that has been implemented as a C++ library and was ported to the ARM architecture.
To reduce the latency and increase the performance, the filters are divided into uniformly partitioned blocks, that are adjusted to the size of the sound card's audio buffer. The partitioned convolution can process filters of arbitrary size (e.g. longer than the buffer size) without adding any *additional* latency to the audio processing chain. Therefore it is frequently called zero-latency convolution [7]. At the same time the computation load is reduced due to matched block sizes between input signal and filter blocks. The convolution is executed in frequency domain using the Fast Fourier Transform (FFT) [8] with an Overlap-Save paradigm. Filters can be dynamically exchanged during streaming. To avoid audible artifacts when switching filters a time-domain cross-fade is applied automatically. An aribitrary number of channels can be processed in parallel, limited by the available CPU resources dependent on the filter length and block size. Information about the performance on the introduced SoCs is given in Section 5.
The filter can be either loaded from the SD-card or can be pushed into the convolution core through a TCP/IP network connection. The input audio signal is either loaded as a WAVE-file from the SD-card, transferred through an UDP channel over network, or taken directly from the sound card's analog input jack. The signal flow is illustrated in Figure 2, the interfaces for filter exchange and playback control are described in the following section.

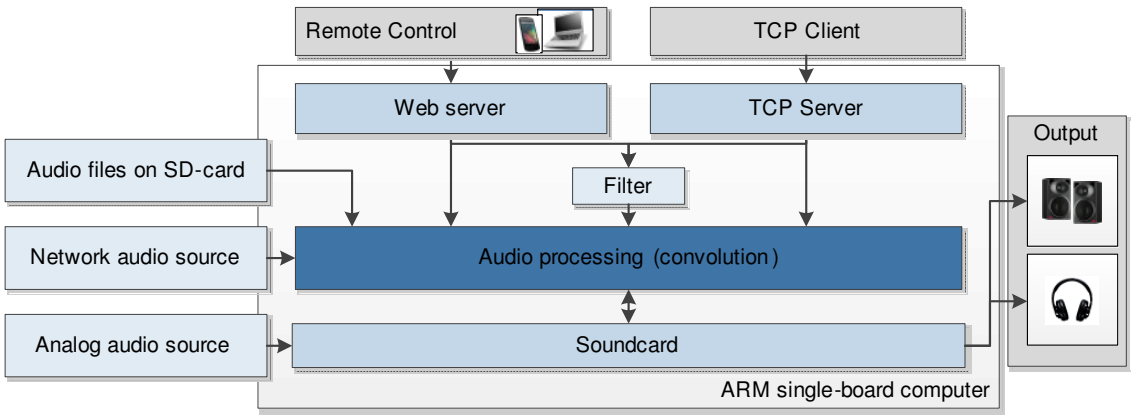

Figure 2: Concept and signal flow of the implemented real-time audio processing on a SoC

3.3 INTERFACES

To support a broad range of possible applications (examples are given in Section 4) different interfaces were implemented to control the SoC. A main focus was set on network-enabled interfaces, so that the single-board computer can function as a stand-alone device, without any non-audio hardware attached, such as a keyboard or display. Therefore several network services were implemented.
The most important service is a TCP/IP server, which enables to control all available functions. This includes the playback, volume and routing control, filter exchange and filter transmission

over network, etc. A TCP client was implemented in MATLAB, which is a convenient environment for preparing and testing filters.
Another network interface allowing graphical user interaction was implemented by setting up a web server. Using a simple web browser, many devices can access an HTML page hosted by the SoC, for example using a mobile phone or tablet computer. The web page shows a control surface with nearly full functionality for various applications, two examples are shown in Figure 3. The sound driver, input signals, processing chain, filters, playback, etc. can be controlled as well as the additional modules binaural encoding and crosstalk cancellation (see Chapter 4). Input files and filter files can be uploaded through the web interface and selected via menu. The system automatically connects to a local network either through Ethernet or by plugging a WiFi-stick into the USB-port. When starting up the SoC it searches for a pre-registered network and automatically connects or opens its own hotspot if the network was not found. This way it is almost in any situation possible to connect to the control surface. In the present study, a couple of applications were implemented and are presented in the next section.

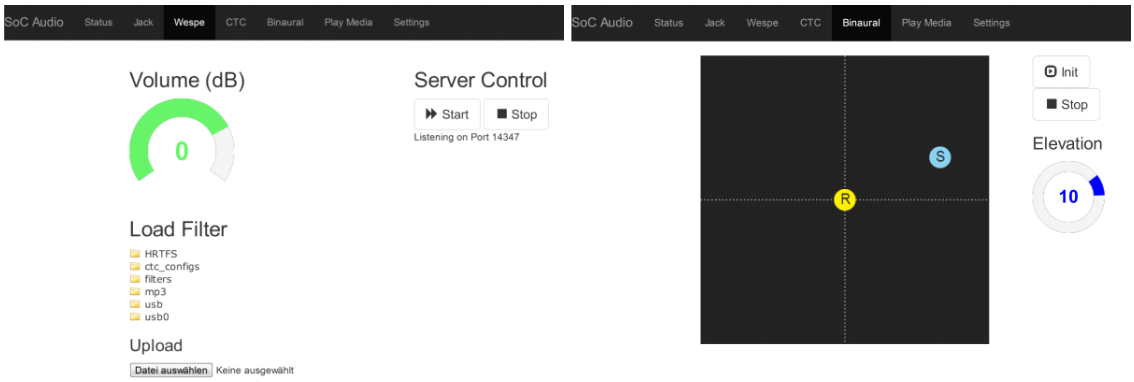

Figure 3: Two examples of the graphical user interface, showing the page to load/upload filters (left) and a binaural panning widget (right).

4. APPLICATIONS

With a properly working convolution application for a small device such as the Raspberry Pi being available, it is an obvious idea to use the system for a loudspeaker equalization. However, there are various other ways to make use of the convolution engine. As the implementation of the real-time convolution was originally dedicated to applications for acoustic virtual reality, e.g., for real-time room acoustics auralization [9] [10], the next step was the implementation of a binaural synthesis and the realization of crosstalk cancellation, enabling binaural reproduction with loudspeakers at little hardware expense.

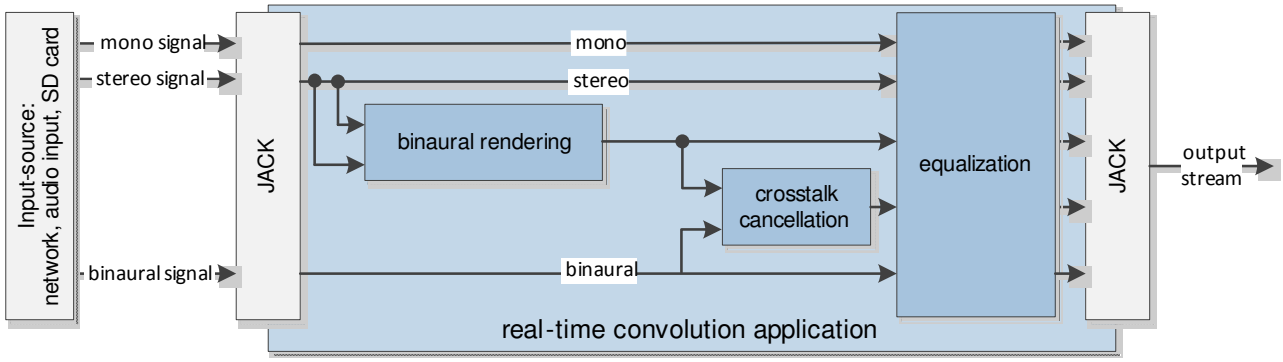

Figure 4: Audiostreaming and processing by three modules using the convolution engine. For all signal types a loudspeaker equalization can be performed (see Section 4.1). Mono as well as stereo signals can be binaurally rendered (see Section 4.2) and can be reproduced via headphones and, if a crosstalk cancellation is applied (see Section 4.3), reproduced by loudspeakers.

4.1 LOUDSPEAKER EQUALIZATION

Especially loudspeakers in the lower price range rarely have a flat frequency response, which makes the application of an equalization a cheap and convenient possibility to improve the reproduction quality of the audio system. If the impulse response or the frequency response of the loudspeaker is known or if it was measured, an equalizing filter can be created. Next to the inversion of the transfer function several other signal processing steps are applied to create the final inverse minimum phase filter. Details of loudspeaker equalization techniques are given in [11]. A big advantage of equalization on a single-board computer is the low cost, small dimensions and low energy consumption. The size of such systems allow the integration into the loudspeaker cabinet, or an installation of the board behind the loudspeaker. Additional loudspeaker cables can be avoided if the single-board computer functions as a streaming server (see Section 4.4) allowing audio input via a wireless network connection.

4.2 BINAURAL RENDERING

By psychoacoustically processing signals with two ears, human beings can localize sound sources. The processing of the human brain is understood to a great extent, enabling virtualization of spatial sound perception using head-related transfer functions (HRTFs). If a HRTF database is available (from a measurement of an artificial head or an indivual [12]), virtual sound sources are created by filtering a signal with the corresponding HRTF. The convolution software created in this project has been extended with a module for such a binaural synthesis, using HRTFs in the DAFF [13] file format. The generated binaural signal must be listened to through headphones.
In the web interface (see Section 3.3) the user is able to define a virtual sound source position, move it and immediately perceive the signal from the chosen direction. The interface also allows the exchange of different HRTF datasets, so that, if available, individual HRTF datasets can be compared to other databases.
In general, the concept of creating binaural sounds can be applied in several different fields, ranging from musical projects over acoustic virtual environments to creating spatial audio tracks for movies. An interesting application is the virtualization of a loudspeaker setup, to listen to or create multichannel mixes.

4.3 CROSSTALK CANCELLATION

Binaural signals must be played back over headphones to make sure to supply the ears with the correct individual signals. If played back over loudspeaker the occurring crosstalk, e.g. from the left loudspeaker to the right ear, avoids a successful spatial perception. To avoid this effect, a crosstalk cancellation (CTC) network [14] [15], also known as transaural reproduction, has been implemented and integrated into the convolution software.
In the user interface, the position of the two loudspeakers as well as the listeners positions have be entered. A set of new CTC filters is calculated and applied to the incoming (line input) or synthesized binaural signal (see Figure 4).

4.4 FURTHER APPLICATIONS

With a freely configurable convolution processor, various applications are possible, but only a few can be presented here. In general a popular application of the Raspberry Pi is to act as a home theatre media player. Using modern standards, such as DLNA, the SoC can receive audio from devices such as network attached storages or mobile phones, and send the audio signal to renderers such as a TV or HiFi system via WiFi connection. By implementing DLNA-capability in the presented project, the SoC can combine the network streaming functionality with simultaneous equalization of the loudspeaker system.
Using a multichannel sound card the SoC can serve multiple loudspeakers, e.g. in larger arrays or for sound systems in different rooms, finally becoming a full loudspeaker management system. If supported by the loudspeakers the multichannel output can be used as a digital frequency crossover network implemented as FIR-filters.

5. PERFORMANCE ANALYSIS

The total analysis of the system's performance is extensive and is currently still ongoing. Different configurations of hardware modules as well as software configurations are tested and evaluated. In this chapter, only two aspects of the performance of a real-time convolution on a single-board computer are presented.
First the performance limits of the convolution on both systems, the Raspberry Pi and the Odroid X2, are investigated. Table 2 shows the maximum possible filter length and the latency times for the corresponding buffer size on both chips.

Raspberry PI			Odroid X2		
Buffer Size	**Max. filter length**	**Latency**	**Buffer Size**	**Max. filter length**	**Latency**
64*2	1024	5.4 ms	96*3	20k	13.5 ms
96*2	2048	7.5 ms	128*3	30k	16.4 ms
128*2	4096	9.7. ms	256*3	80k	31.0 ms
256*2	12k	18.0 ms	512*2	200k	48.7 ms

Table 2: Performance limits for convolution on Raspberry Pi (*Codec Shield* WM8731 I²S sound card) and Odroid X2 (UCA202 sound card, sampling rate 44.1 kHz in both cases).

It was found that I²S sound cards in general are not only more stable but also decrease the latency by around 5 ms in comparison to USB devices. The table also shows that, if a latency below 15 ms is desired, both systems are able to perform filter convolutions with at least 1024 coefficients, which is sufficient for a binaural synthesis, crosstalk cancellation and, in most cases, also for loudspeaker equalization [11] [16]. Due to higher computational power, the Odroid X2 board is able to convolve much longer filters, however the minimal buffer size and thus the latency is higher and, due to the USB connection, audio glitches might occur if the system is busy.
To investigate the benefit of the vector floating point unit for the implemented convolution, the CPU usage has been analyzed for the instruction set VFPv2 enabled and disabled (for both sound card types). Figure 5 shows a slight improved for lower period sizes if VFP is enabled, for higher period sizes (>256) the CPU workload does not differ. It also can be stated that processing audio via USB on a Raspberry Pi increases the workload of the chip.

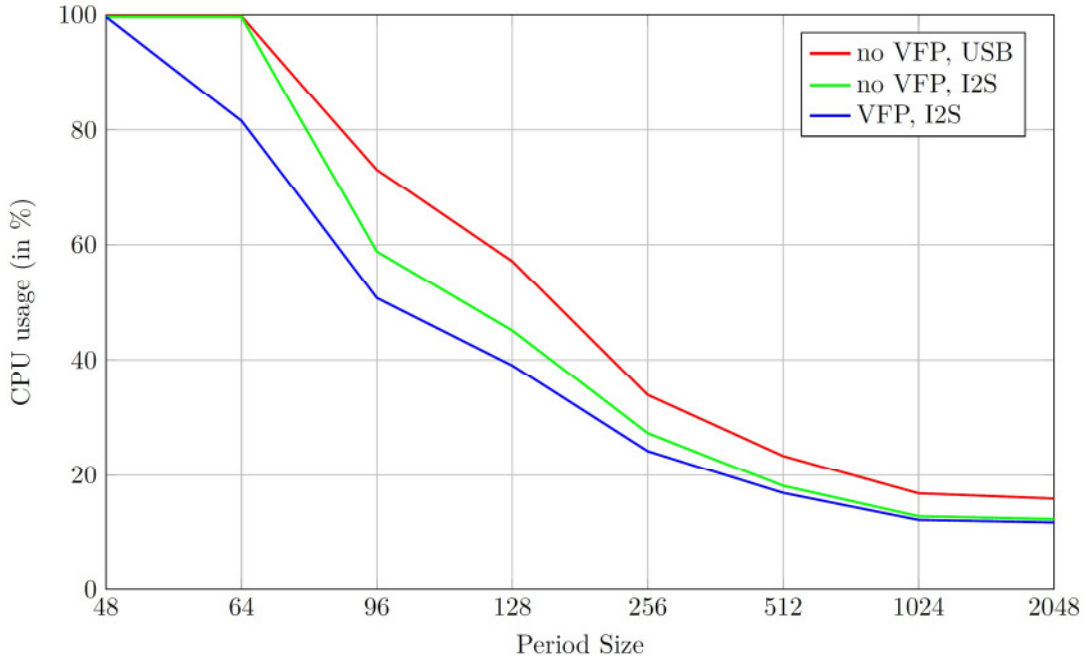

Figure 5: CPU load on Raspberry Pi (stereo, equalization only, filter length 1024, sample rate 44.1 kHz), using a USB sound card and I²S DAC with VFP-optimized convolution. At loads of >98 %, the audio output is massively distorted.

6. CONCLUSION AND OUTLOOK

An application for real-time convolution was successfully ported and configured on two different ARM-based single-board computers. Next to the loudspeaker equalization, modules for binaural rendering of virtual sources and for binaural reproduction via loudspeakers based on crosstalk cancellation were also integrated. Network- and Web-based interfaces provide easy control and configuration. In general, small single-board computers are capable of performing these tasks, however, the Raspberry Pi has difficulties performing more computationally demanding tasks such as rendering multiple sound sources. With these modules being integrated, the systems are suitable for experiments and demonstrations for students.
However, the implementation works only properly on the tested hardware setups and quickly loses stability and reliability if the hardware (sound cards) are exchanged or a different software or operating system configuration is chosen. Adjusting such a system to the personal demands requires a lot of patience and motivation to experiment with and learn about the operating system, drivers, hard- and software configurations.
I²S sound cards were more stable during testing and provided lower latencies, but the performance analysis showed so far, that also with USB sound cards reasonable total latencies of less than 15 ms (and almost less than 6 ms with the optimal system configuration) can be achieved. The convolution algorithm itself can be described as highly efficient and suitable for the real-time application.
In future, it is planned to adjust the configuration to improve the reliability of the software on single-board computers as well to extend the system with the option of multichannel processing. Also different single-board computers, which just recently became available, such as the *Banana Pi* or the *Cubietruck* board will be evaluated for their feasibility to run the presented implementations.

REFERENCES

[1] S. Furber, ARM system-on-chip architecture, Addison-Wesley Professional, 2000.

[2] D. McGrath and A. Reilly, "Huron – A digital Audio Convolution Workstation," 1995.

[3] M. Jang, K. Kim and K. Kim, "The Performance Analysis of ARM NEON Technology for Mobile Platforms," *Proceedings of the 2011 ACM Symposium on Research in Applied Computation,* pp. 104-106, 2011.

[4] L. Abeni, A. Goel, C. Krasic, J. Snow and J. Walpole, "A Measurement-Based Analysis of the Real-Time Performance of Linux," *Proceedings of Real-Time and Embedded Technology and Applications Symposium,* pp. 133-142, 2002.

[5] R. Love, "Lowering latency in Linux: introducing a preemptible kernel," *Linux Journal,* 2002.

[6] J. Corbet, A. Rubini and G. Kroah-Hartman, Linux device drivers, O'Reilly Media, 2005.

[7] F. Wefers, D. Schröder, S. Pelzer and M. Vorländer, "Real-time filtering for interactive virtual acoustic prototyping," *Proceedings of 8th European Conference on Noise Control ,* 2009.

[8] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier," *Mathematics of Computation,* pp. 297-301, 1965.

[9] L. Aspöck, S. Pelzer, F. Wefers and M. Vorländer, "A Real-Time Auralization Plugin for Architectural Design and Education," *Proceedings of the EAA Joint Symposium on Auralization and Ambisonics,* 2014.

[10] S. Pelzer, L. Aspöck, D. Schröder and M. Vorländer, "Interactive Real-Time Simulation and Auralization for Modifiable Rooms," *Building Acoustics,* pp. 65-74.

[11] S. Müller, Digitale Signalverarbeitung für Lautsprecher, RWTH Aachen University, 1999.

[12] B. S. Masiero, Binaural technology for virtual reality, RWTH Aachen University, 2008.

[13] F. Wefers, "OpenDAFF - Ein freies quell-offenes Software-Paket für richtungsabhängige Audiodaten," *Proceedings of DAGA Berlin,* 2010.

[14] B. S. Atal and M. R. Schroeder, "Apparent sound source translator". Patent US3236949 A, 1966.

[15] T. Lentz, Binaural technology for virtual reality, RWTH Aachen University, 2008.

[16] J. Mourjopoulos, "Digital Equalization Methods for Audio Systems"., *Proceedings of AES Convention 84,* 1988.