AN ANALOG I/O INTERFACE BOARD FOR AUDIO ARDUINO OPEN SOUND CARD SYSTEM

Smilen Dimitrov
Medialogy, Aalborg University Copenhagen
sd@{imi,create}.aau.dk

Stefania Serafin
Medialogy, Aalborg University Copenhagen
sts@{imi,create}.aau.dk

ABSTRACT

AUDIOARDUINO [1] is a system consisting of an ALSA (Advanced Linux Sound Architecture) audio driver and corresponding microcontroller code; that can demonstrate full-duplex, mono, 8-bit, 44.1 kHz soundcard behavior on an FTDI based ARDUINO. While the basic operation as a soundcard can be demonstrated with nothing more than a pair of headphones and a couple of capacitors - modern PC soundcards typically make use of multiple signal standards; and correspondingly, multiple connectors.

The usual distinction that typical off-the-shelf stereo soundcards make, is between *line-level* signals (line-in/line-out) - and those not conforming to this standard (such as microphone input/speaker output). To provide a physical illustration of these issues in soundcard design, this project outlines an open design for a simple single-sided PCB, intended for experimentation (via interconnection of basic circuits on board). The contribution of this project is in providing a basic introductory overview of some of the problems (PWM output in particular) in analog I/O design and implementation for soundcards through a real world example, which - while incapable of delivering professional grade quality - could still be useful, primarily in an educational scope.

1. INTRODUCTION

In contemporary terms, a soundcard is a device that is recognized by consumer users in having a specific role: it provides an analog input/output hardware interface to high-level PC audio software (from media players like VLC to processing environments like PureData). As such, the development of a soundcard can be seen as a cross-disciplinary effort, requiring understanding of both (analog and digital) electronics, and software engineering (from OS drivers to application level). Consequently, it is (for the most part) commercial enterprises that have the resources to develop soundcard systems, in terms of practical implementation.

However, thanks to the broader penetration of affordable technology in the mass market, as well as developments in open source software - the development of practical soundcard systems could be also within the reach of the DIY developer. This paper can be seen in the context of a wider project aiming to broaden the discussion on open, DIY soundcard designs (as examples of PC controlled digital audio hardware): [2] discusses an obsolete, historic hardware design (with discrete parts) controlled by simple software - while the AUDIOARDUINO [1] project demonstrates a working, open soundcard system, which is based on: an FTDI based ARDUINO DUEMILLANOVE as hardware; specific microcontroller code for ARDUINO's ATMEGA328; and matching ALSA soundcard driver for Linux. Note that the main discussion of the streaming capability, software and microcontroller issues of AUDIOARDUINO is given in [1]; this paper can be seen as an extension, focusing on the issues of analog I/O.

The wider intent is to address those, that wish to start studying the interaction between digital audio hardware and software through practical examples - and as such, would represent 'novices': here understood as people that may have basic understanding of analog (e.g. op-amps) and digital (e.g. 74XX series) electronics, as well as software (C programming) - but not necessarily *practical* experience with application of these domains in the context of digital audio. As such, they are likely to encounter a similar situation, as some of the developers on this project: while there may be ample literature on best practices (also from industrial perspective) in both digital and analog audio, often times the discussion is in reference to simulation results or industry equipment; and it can be difficult to parse for a novice, without a previous practical insight. Then again, the practical insight can be difficult to gain, assuming the easiest (and most obvious) access to parts for such novices is an average electronics lab (typically offering 'classic' through-hole, discrete parts).

The board in this project follows that naïve approach: the intent is not to provide a design competitive or comparable to commercial products; rather, it is to start discussing issues in analog I/O, from the perspective of AUDIOARDUINO, through a board that could relatively easily be assembled by novices. AUDIOARDUINO takes the CD quality (16-bit, 44.1kHz - supported by many soundcards) as a reference, and then attempts to achieve it, insofar as the hardware and the simplicity of the approach allow. Eventually as an 8-bit, mono device, AUDIOARDUINO allows for simpler conceptualization of the the 'analog sample' (as a building block of the digital audio stream) in the software domain; this board, then, aims to complement that approach, and simply allow for the 'analog sample' to be more easily traced in both analog and digital domains - in the same context as real soundcards. Furthermore, there are commercial offerings that use the ARDUINO for sound, such as WAVESHIELD, VOICEBOX SHIELD, SEEEDSTUDIO ARDUINO MUSIC PLAYER SHIELD or RMP3, which

Copyright: ©2011 Smilen Dimitrov et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

provide better fidelity than AUDIOARDUINO - however, they represent standalone audio players/recorders; while AUDIOARDUINO represents a soundcard. As such, while the study of these devices may hold the key for a quality analog I/O for AUDIOARDUINO, the intent here is more on documenting how basic elements behave in simple configurations - as opposed to obtaining performance. This is also the reason why established techniques like data compression, or companding (A-law or μ-law), are not addressed; while many other technical details are included for reference. In this way, the paper could serve as introductory material, especially to people with untraditional electronics engineering background - in particular the wider electronic music instrument community. [1] And while the provided technical details may be difficult to structure in a more meaningful way for non-specialists, they represent a kind of practical experience which, the authors believe, makes the difference between a theoretical concept and a practical exercise - and as such, would be of interest to novices.

Eventually, as this paper concludes, the use of this board does not necessarily deliver any advantage - as opposed to using the 'raw' analog I/O of the ARDUINO. Yet, documenting its development and performance, through this paper and media (schematics, video) on associated webpage [3], could be of use as a starting point to novices - even without building the board. Thus, the contribution of this paper is primarily educational - however, it could possibly lead to the development of an analog I/O board for AUDIOARDUINO, that would be close to matching the contemporary state-of-the-art.

2. PREMISE

In simple terms, the problem here can be stated as follows: while AUDIOARDUINO can demonstrate a soundcard operation, it does so by reading a 0-5V analog input, and providing a 0-5V PWM signal as analog output; however, soundcards typically feature line (line-in and line-out) connectors, as well as microphone input and speaker output - all of which operate with analog signals, with a different format from those that can be obtained directly from the ARDUINO. The question is then, what basic circuitry could we use to address the conversion between the analog interface already present in AUDIOARDUINO, and the typical analog interface found on a soundcard; and what could be expected from a practical implementation of the same.

As discussed in [1], an ARDUINO is *sufficient* to demonstrate the operation of a soundcard - in particular, because on-board facilities of ARDUINO's ATMEGA328 are used to implement analog input/output (I/O). In a soundcard architecture sense, the ARDUINO board can be said to implement the functionality of both the digital *bus* interface, and the analog-to-digital (ADC) and digital-to-analog (DAC) conversion.

In terms of *analog input*, the ATMEGA328 contains a single 10- bit successive approximation ADC unit, which is in turn connected to an 8-channel *multiplexer* [4, p.251]. The use of an ARDUINO for AD conversion of diverse sensor signals is standard practice [5], where the usual user expectation is to obtain values in the 10-bit range (from 0 to 1023) - as representation of a voltage signal, in the range from 0 to 5V, brought to an analog input. Lacking any input filters, signals containing a constant (DC) [2] component can be sampled without a problem (in contrast to the main issue in [2]).

In terms of *analog output*, the ATMEGA328 offers three programmable 'Timer/Counters': Timer1 (16-bit), Timer0 and Timer2 (8-bit) [4]. Each of these can count monotonically up or down (which corresponds to a saw or triangular 'digital' signal in time domain); and have associated 'Output Compare' (OC) units, registers and pins - which allow for comparison of the current counter value with a preset value. The result of this compare operation, can be output on the respective OC pin as high (5V) or low (0V) voltage level. Thus, the OC units can be used as binary (i.e., 'square') signal generators - where the period of the signal is determined by: the clock frequency, maximum and minimum value of the counter (called 'top' and 'bottom' in [4]), the mode and direction of counting, and the result of the current compare operation. In AUDIOARDUINO, the analog representation of the incoming 8-bit/44100 Hz data stream is a binary PWM signal with a frequency of 62500 Hz - where the *duty cycle* of the PWM signal corresponds to the analog sample value.

3. ANALOG I/O AUDIO LEVEL STANDARDS

When discussing analog signal interfaces for audio, related literature often mentions three categories: [6] mentions three levels used in a recording chain - microphone level, line level and speaker level; while [7] talks about low-level analog signals, line-level signals and amplified signals. In principle, these categories would simply represent increasing signal levels - thus having a definition for 'line level' audio, would also partially specify the other two domains.

However, it is not easy to find a single definition of what line level audio is. The term 'line-level' itself may originate from early use in telephony [8], referring to pre-amplified microphone signals [9]; but its possibly easiest definition is as both 'the output level of a preamplifier', and 'the input level of a power amplifier' [8]. Furthermore, there could be differences between professional and consumer grade audio hardware: [6] mentions '*+4 dBu, which is 1.23V*' for professional, and '*-10 dBV, which is 0.316 V*' for consumer applications; [8] mentions '*any level above 25 mV RMS*' for consumer, and '*0 VU*' as reference for commercial applications - where 0 VU could be: 0.775 V RMS; 1.23 V RMS (+4 dBm); or 1.95 V RMS (+8 dBm). Also, [10] defines 0 dB for line-level via maximum amplitude of ± 0.7 V; while [11] reports ± 1.5 V as amplitude for line-level output of an IPOD.

Given how varying definitions for line-level signal can be, one could wonder whether there is a standard that could

[1] which, partly thanks to platforms like the ARDUINO, may have experienced increased exposure to basic analog electronics issues - also for its members with primary expertise in other fields

[2] Note that while DC stands for 'direct current', it is often used, especially in power supply adapters, to describe a constant voltage signal (as in "12Vdc") - and a constant signal in general

be consulted. Bohn's article [12] is solely dedicated to the complexity involved in pursuing audio standards: the complexity arises from multiple organizations (not all of them solely dedicated to audio) issuing standards; these organizations merging and changing character through history, often results with the same set of standards issued under different names; the for-profit characters of standards committees allows for thousands of dollars in cost for complete sets of standards. Also, since standards are copyrighted, it is unclear to what extent they can be legally cited in an open-source project. That being said, 'line-level' signals are most likely defined either in IEC 60268 [3] "Sound System Equipment" (via [12]), or in EIA-RS-160[3] "Sound Systems" (via [8] and search, see also [13]). However, for purposes of this document, we will consider *microphone* level signals to be below 25 mV RMS (35 mV peak) - and *line* level signals to be above microphone levels, and below 1.23 V RMS (1.74 V peak); the peak being expressed in terms of a sinusoid (according to $V_P = V_{\text{RMS}} \cdot \sqrt{2}$).

In AUDIOARDUINO, in terms of *input*, we can essentially abstract the ADC process, as we bring a given analog signal directly to a pin of ARDUINO's ATMEGA328. As such, conforming to line level on the input side will require nothing more than simple scaling [4] in the analog domain. However, in terms of *output*, what we do obtain as a *final* analog signal representation *is* a PWM signal. Thus, if we want to conform to line levels on the output, we must first bring this PWM back to a format, where audio information is encoded as a voltage level - i.e., we need to perform a sort of a PWM to analog conversion (Sec. 5.1) - before we can apply filtering (as appropriate for DA conversion) and scaling[4] (for line level conformance). Such conversion requires an overview of the PWM signal characteristics.

4. PWM AS ANALOG SIGNAL REPRESENTATION

Analog to digital conversion is often introduced through the concept of a *sampled* (discretized in time) signal: an analog signal $V(t)$ (*a*, Fig. 1) is periodically sampled with a frequency f_S; the values of $V(t)$ at the moments of sampling, are taken as representation of the signal (are *held*) for the duration of the entire sampling period $T_S = 1/f_S$ (*b*, Fig. 1). The useful information carried by a sampled analog signal is encoded as an analog voltage level. The sampled values can further be *quantized* (discretized in amplitude) to a finite set of levels, separated by a step value. This allows for finite enumeration of the quantized level set (digitizing), and further encoding with e.g. a binary code. [5] As such, there is a direct correspondence between analog sampled-and-held (SH) and *pulse width modulation* (PWM) [14] signals, which will be briefly outlined here through Fig. 1.

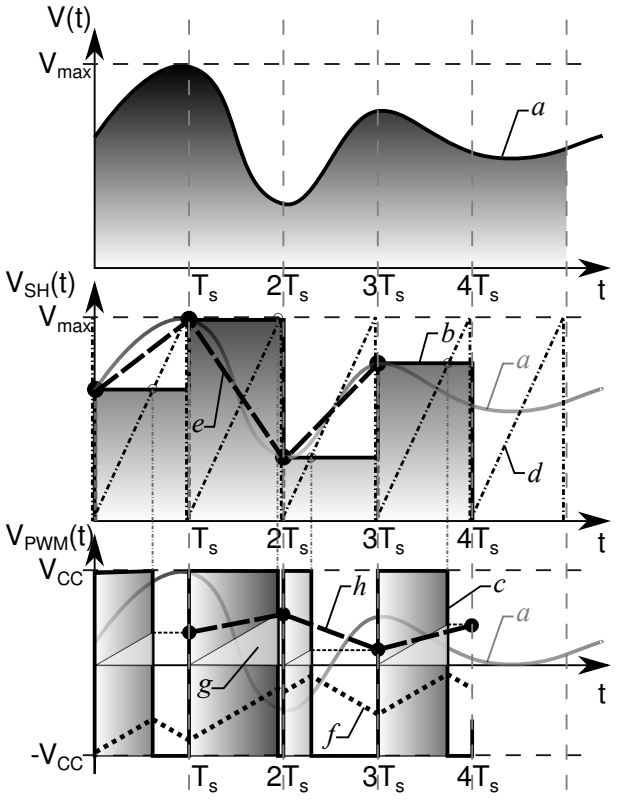

Figure 1. Comparison between analog signal (top, *a*); its sample-and-hold (middle, *b*) and its PWM representation (bottom, *c*) [signals emphasized with gradient filled areas, see text for the other markings].

To begin with, $V_{\text{SH}}(t)$ (*b*, Fig. 1) can be produced by running $V(t)$ (*a*) through a *sample-and-hold* circuit, clocked at frequency f_S; while $V_{\text{PWM}}(t)$ (*c*) can be produced by running $V(t)$ through a *comparator* circuit, which compares $V(t)$ with a ramp (sawtooth or triangular) signal (*d*) of frequency f_S. The SH signal encodes the sampled voltage value as voltage (and the sample representation is in the same domain as the original value); while the PWM signal encodes the same voltage value as the duration of time in which the PWM signal has the high value (V_{CC}), known as *duty cycle* (thus the sample representation and the original value are not in the same domain). Note that these processes simply quantize the analog signal in time; obtaining a binary digital representation requires additional stages. Most AD converters work with voltage as input, enforcing a given sampling resolution, and can thus be directly applied to a SH signal – while obtaining a binary digital value from a PWM signal essentially requires resampling it with a frequency N times higher than the sampling frequency f_S, where N is the number of quantization levels; [6] and then counting the number of times the PWM signal has been high within a period.

Using a sawtooth signal (*d*, Fig. 1) for the PWM comparison results with single-edge PWM - and in terms of [15], the PWM signal (*c*) on Fig. 1 is a single-edge, or specifi-

[3] IEC: International Electrotechnical Commission; IHF: Institute of High Fidelity; EIA: Electronics Industries Association (now Alliance); RS: Recommended Standard

[4] In this paper, *amplification/attenuation* refers to multiplication, i.e. $y(t) = a \cdot x(t)$; and *scaling* refers to multiplication and constant addition, i.e. full linear transform $y(t) = a \cdot x(t) + b$

[5] Note that many resources may often refer to visualizations of sampled, quantized analog signals as "PCM"; however, pulse-code modulation (PCM) is a binary encoding technique, which can have the return-to-zero (RZ) or non-return-to-zero (NRZ) waveforms.

[6] e.g., for $n = 8$ bit values, there are $N = 2^n = 256$ quantization levels, so resampling must occur at frequency $256 \cdot f_S$

cally *uniform-sampling trailing-edge* PWM signal; the same kind which is generated by ATMEGA328's Timer0 in 'Fast PWM' mode. For this kind of PWM in particular, we can easily establish a correspondence to SH: on Fig. 1, the actual signals in each case have the 'area under the curve' filled with a gradient. For $V_{\text{SH}}(t)$ on Fig. 1, the dots indicate the sampled values and the moments of sampling of the original analog signal - the thick dashed line connecting them (*e*) shows a *linear interpolation* reconstruction of the original signal, based on these sampled values. Simultaneously, the $V_{\text{SH}}(t)$ diagram shows a sawtooth signal (*d*), which explains how the particular PWM signal below is derived: first, the sample value for PWM is the same as the one for SH - because the saw period is the same as the SH sampling period, and the sample value is taken at the beginning of the period. Thereafter, while this sampled value (same as the SH signal level) is higher than the current sawtooth value, $+V_{\text{CC}}$ is output as PWM signal value; as soon as the sawtooth signal becomes higher than the sampled value, $-V_{\text{CC}}$ is output - hence, there is a linear correspondence between the PWM duty cycle and SH level, [7] in representing a single sample value. This is also visualized on Fig. 1: the grey triangles (*g*) within $V_{\text{PWM}}(t)$ represent the result of integration of $V_{\text{PWM}}(t)$ *only* while it is in the duty cycle (active or high). If the integrated value at end of each duty cycle is translated at the end of the PWM period and taken to be the sample value (indicated by dots on $V_{\text{PWM}}(t)$), then the linear interpolation between these points (indicated again by a dashed line) (*h*) will be a scaled version of the linear interpolation of the SH signal (*e*).

In terms of spectrum, we can approximate the SH signal to a multiplication of the original modulating signal with a comb (infinite Dirac pulse sequence) signal with frequency f_S - a basic result in sampling theory is that this corresponds to convolution of the original and the comb spectra, which results with sideband images (of the baseband spectrum) around the harmonics of f_S that extend to infinity. To reconstruct a baseband signal with a spectrum limited by frequency f_{max}, Nyqist-Shannon's sampling theorem $f_S \geq 2f_{\text{max}}$ must be satisfied. A PWM signal can be approximated to a square one [16], and thus to a series of odd harmonics at $f_S, 3f_S, 5f_S...$; it can be shown that the PWM spectrum will contain: the original modulating signal (baseband); the harmonics; and sidebands around the harmonics (note, however, the PWM spectrum in reality is more complex [17]). Thus, Nyqist-Shannon's theorem should be applicable to PWM as well - meaning that it should be, in principle, possible to reconstruct an analog representation of a PWM signal just by using a low-pass filter. In fact, simple LPF has been used for PWM reconstruction at least since 1937 (Reeves patent [18]); see [19] for a practical note on using simple RC filters.

Furthermore, use of PWM is common in audio amplifiers, known as 'Class-D'; often, PWM is used directly [20] to drive a loudspeaker, counting on the speaker's filtering properties; an approach already used for raw demonstration of AUDIOARDUINO in [1]. Given that a typical loudspeaker [8] uses a coil to electromagnetically move a diaphragm membrane, electrically it can be approximated to a coil (inductor). As the magnetic field that moves the diaphragm is caused by *current*, the speaker can be considered (electrically) to be a current-driven element. Note however, that while $V_{\text{PWM}}(t)$ on Fig. 1 pulses between -V_{CC} and +V_{CC} – the PWM signal generated by the ARDUINO pulses between 0V and 5V. This, in turn, means that the current generated from such a PWM source will be strictly *unidirectional* [9] - causing the diaphragm to move in one direction only (which is audibly weaker in comparison to a diaphragm driven both ways). This problem - generating bidirectional current in absence of negative voltage supplies - is often addressed electronically with H-bridge circuits (mentioned also in Sec. 5.2).

5. BOARD DESIGN / IMPLEMENTATION

The design approach for the board was to allow for quick testing of several approaches to analog I/O with AUDIOARDUINO; hence the board consists of several modules, charted on Fig. 2.

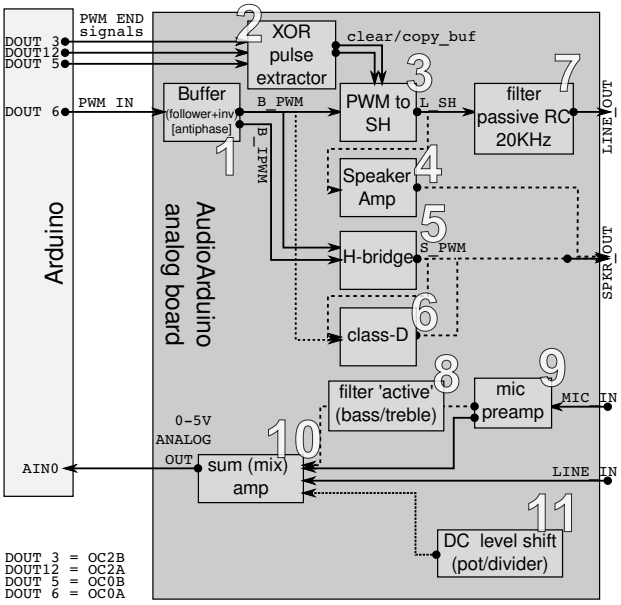

Figure 2. Block diagram, representing units of the AUDIOARDUINO analog I/O board.

Fig. 2 shows that the PWM to SH conversion is performed by a buffer (1), XOR pulse extractor (2) and PWM to SH circuit (3); further discussed in Sect. 5.1. There are three types of amplifiers: an 'analog' speaker amplifier (4), and PWM-oriented H-bridge (5) and Class-D (6) amps; further discussed in Sect. 5.2. There are two filters; passive RC (7), and active 'bass/treble' filter (8); and for handling input, there is a mic preamp (9), mixer or summing amplifier (10) and a DC level shifter (11); further discussed in Sect. 5.3.

[7] as percent of time when the signal is high vs. percent of the voltage level in respect to the range represented by V_{max}

[8] including most headphones; however, excluding piezoelectric and electrostatic speakers

[9] i.e., the current will flow in one direction during the duty cycle; and outside of it, current will not flow at all

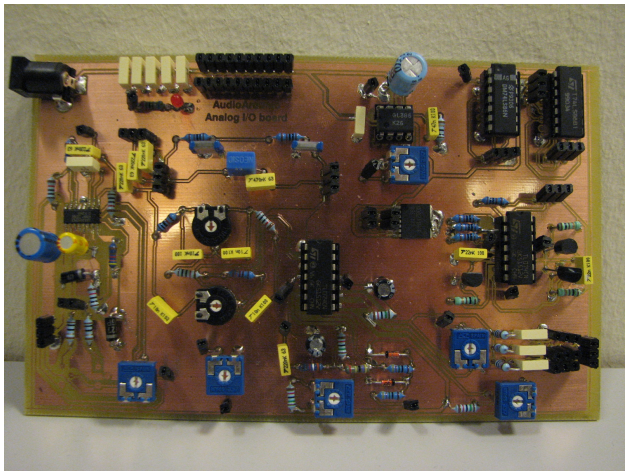

Figure 3. Photo of a finished AUDIOARDUINO analog I/O board.

The board schematics and PCB layout files have been implemented in the open-source `kicad` software, and have been released on [3]. The board has a single-sided design, implemented on a UV photosensitive PCB, which hosts both surface-mounted parts, and through hole ones (like resistors) soldered on the surface. Each of the aforementioned units is essentially a standalone module, with only some connections (like power) implemented on the board; for establishing connections, multi-pin single-row IDC socket pins are split in single connectors and soldered, in which wire can directly be inserted (to allow for simple manual wire-wrapping). Thus, the connections on Fig. 2 are not fixed: the full lines simply represent a starting configuration, and the dashed lines represent alternative ones; the completed board (without wires) is shown on Fig. 3. The board also has an adapter socket to accept DC power supply, which is distributed to most (but not all) parts of the board – and audio connectors can be added, by attaching them to respective pin sockets.

5.1 PWM to analog (SH) conversion

As noted previously, low-pass filtering is standard practice for reconstructing a PWM signal in the analog domain; the terms usually applied are PWM "reconstruction", "filtering" or "demodulation". However, a technique known as "ramp and hold" can be considered a PWM to sampled, quantized analog (or "PWM to SH", for sample-and-hold) signal converter: the quantization of a microcontroller-generated PWM duty cycle will be effectuated as quantized analog voltage levels; and due to a dependence on a clocked, regular 'clearing', quantization in time is inherent. This section describes the reasoning behind a simple "ramp and hold" implementation with discrete parts, utilized here as PWM to SH converter.

To discuss the PWM to SH conversion, note first that PWM audio power amplifiers typically work with switching (PWM) frequency between 200 kHz and 800 kHz [16] – even if Sec. 4 implied that 44.1 kHz (which, by Nyqist, cover the audible 22.05 kHz analog spectrum) is applicable as lower bound for PWM frequency. Then, let's return to AUDIOARDUINO's method of generating PWM on the ATMEGA328. The 'Timer0' timer/counter is used for this purpose: once set, it runs continuously at the specified timer clock frequency (in 'parallel' with the main code execution). The achievable frequency of the waveform it can generate is: $f_{\text{PWM}} = {}^{f_{\text{clk\_IO}}}/_{256N}$ for Fast PWM mode [4, p.103], or $f_{\text{PCPWM}} = {}^{f_{\text{clk\_IO}}}/_{510N}$ for Phase Correct PWM mode [4, p.104]; where N is the 'prescale factor' (1, 8, 64, 256, or 1024). Thus, the highest possible PWM frequency on ATMEGA328 with 16 MHz clock is is ${}^{16000000}/_{256} =$ 62500 Hz; achieved in Fast PWM mode with prescaler $N = 1$.

A PWM signal with frequency of 62.5 kHz should be able to reproduce the content of a 44.1 kHz digital stream in the analog domain, given that the sample sizes are the same (here 8-bit). The next possible waveform frequency (for $N = 2$) is 31.25 kHz, which - being lower than 44.1 kHz - is not suitable for reproduction; and that is the case for all other PWM frequencies achievable on this ARDUINO. Here, let's note that the Timer0 counter simply increases the value of the register (variable) TCNT0 at each clock tick; since TCNT0 is 8-bit wide, when it reaches 255 it 'overflows' on the next tick (that is, it is reset to 0) - and this is what sets the PWM period. A matching register, OCR0A, is used to set the duty cycle - it is continuously (at each clock tick) compared to TCNT0: and if it is bigger, the matching pin OC0A is set to high voltage (Vcc); else it is set to 0 (ground).

We can now identify some sources of error in this arrangement. The microcontroller code writes a single sample to PWM (that is, OCR0A) at 44.1 kHz; the PWM runs independent of that at 62.5 kHz. At the moment of writing, the counter may still process the previous (analog sample) value - and the new value will be output first at the beginning of the next PWM period. This could be also seen as (analog) SH samples being displaced from their default positions, and as such could be considered 'analog' jitter of sorts (see [21] for jitter measurements in professional equipment). Additionally, note that "*the extreme values for the OCR0A Register represent special cases* [4]" - meaning that reproduction errors could be experienced for values 0 and 255, the limits of the 8-bit range. Assuming that these errors will be tolerable, the problem now is how to extract a SH type of voltage from the PWM signal, to conform with 'line-level' format.

Now, let's briefly return to the case of a loudspeaker, which we can discuss as an inductor with inductance L. The current through an inductor L is the integral of the voltage across: $i(t) = {}^{1}/_{L} \int_0^t v(t)\mathrm{d}t$ – ideally, [10] constant voltage would result in linear ramp current. Thus, if an inductor is driven by a *voltage* signal $V_{\text{PWM}}(t)$ as on Fig. 1 - then the *current* through it (shown as the dotted line on the PWM signal, *f* on Fig. 1) will be: an upward ramp during the duty cycle (for $+V_{\text{CC}}$); and downward ramp outside of it (when the voltage is $-V_{\text{CC}}$). Ideally, since $\text{abs}(-V_{\text{CC}}) = +V_{\text{CC}}$, the upward and downward ramp of the current would

[10] A problem is that we cannot really approximate a speaker to an ideal inductor; taking resistances into account, we now discuss RL circuits - which instead of a linear ramp, will produce an *exponential* current (which further increases the reproduction error; see [16] for the same problem, but in terms of coding carrier linearity).

have the same slope (30° on *f*, Fig. 1) - however, note that even in this ideal case, the current signal thus obtained does *not* have a shape that follows the shape of the SH voltage interpolation (the dashed line *e* on Fig. 1) we are interested in. While this can be addressed by increasing the PWM frequency, [11] we would still have inaccurate sample reproduction in the audio domain (for the particular type of PWM signal on Fig. 1), even in the case of an ideal inductor (speaker).

However, if we perform integration *only* during the duty cycle, the integrated values at end of *each* PWM period will correspond to the SH values. And this is ideally what happens with the unipolar (0/+V_{CC} V) PWM generated by ARDUINO fed to a speaker: integration is performed during duty cycle; and off duty cycle, there is no current and thus no ramp in the other direction (although current should leak). We can apply the same thinking to obtain integrated signal in the voltage domain, by replacing the inductive speaker with an *integrator* circuit. The basic integrator typically charges a capacitor (CInt0 on Fig. 5) as a way to obtain an integrated signal. Thus, to obtain integration *per each* PWM period, we must discharge the capacitor at end of each PWM period - which means we must somehow detect the beginning and end of the PWM period.

Technologies like clock recovery or phase-locked loop circuits are usually needed to extract a signal describing the PWM period from an unspecified PWM signal. Here, we instead use remaining timers on the ARDUINO to generate these signals: we can set the other 8-bit timer/counter Timer2 to run in sync with Timer0 (causing counter values TCNT0 == TCNT2 at all times) - and set the remaining OC pins (OC2A, OC2B, OC0B) to turn high when counter reaches values 252, 253 and 254, respectively [12] - thus indicating the end of PWM period. These signals all turn zero at the start of the PWM period, when the counter is 0 - meaning that they overlap in time (Fig. 4 bottom left). To extract mutually *exclusive* pulses indicating counter value 252, 253 and 254 (Fig. 4 bottom right), we can employ the XOR extraction circuit, shown on bottom part of Fig. 4.

For electronic buffering of the PWM signal, XOR circuits (from a 74LS86 chip) are used, because they can be configured (in terms of binary TTL level signals) to work as either *follower* or *inverter* buffers. [13] The pulse extractor XOR circuit (Fig. 4, bottom) accepts the end-of-PWM-period OC pins' signals, and outputs buffered end-of-period pulses (xBEOP) corresponding to 4, 3 and 2 counter ticks (counts 252, 253 and 254) before start of next PWM period. These end-of-PWM-period pulses can then be applied as COPY and CLEAR (respectively, xBEOP-3 and xBEOP-2 on Fig. 4) of the PWM-to-SH circuit, whose schematic is shown on Fig. 5. The PWM-to-SH converter on Fig. 5 consists of: a basic *integrator* (resistors RInt1-4=RInt, opamp (A) and capacitor CInt0); structure for discharging the capacitor (transistor QInt_Empty0 and resistor RbEmpty0); a structure known as *transmission gate* or *pass transistor* which behaves as an analog switch (transistors Q_Pass1,2 and resistors RbPass1,2); a 'copy buffer' capacitor C_cpbuf_0 (which together with the analog switch forms a '*Sample and Hold*' circuit); and intermediate (B) and output (C) buffer followers.

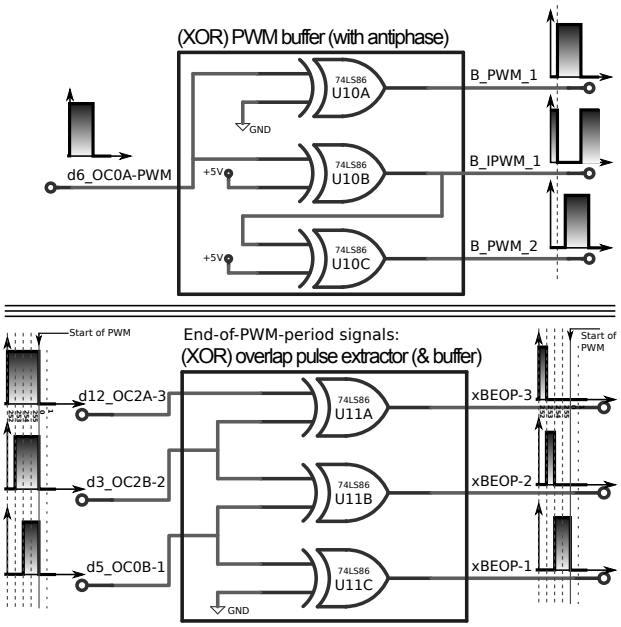

Figure 4. Schematics of PWM buffer (top) and PWM end-of-period pulse extractor (bottom), implemented with XOR gates.

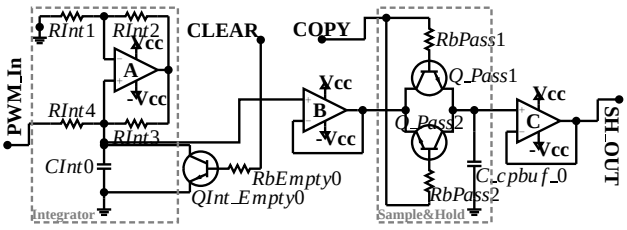

Figure 5. Schematics of the PWM to SH converter.

The integrator used here is a *non-inverting* integrator (see [22]), [14] so as to preserve the phase of the integrated signal as on Fig. 1. As in most RC structures, this circuit too will generate exponentially changing voltage as the result of the processing of a constant voltage input; however, if the RC product is much smaller than the PWM period, the exponential voltage change can be approximated to a linear ramp. The transmission gate structure is based on the assumption that its transistors are either off or saturating, making the structure appear as either high or low resistance. This approximation to an analog switch will be better for transistors with higher β (h_{FE}, forward current gain) parameter, and very low cut-off currents. While FET transistors would be more appropriate for this role - BJT

[11] Note that (unlike Fig. 1) literature may often show a single sinusoid period, sampled by eight [14] up to tens of PWM periods [16]; higher PWM frequency forces the integrated signal to more closely resemble the original one

[12] Using values 253, 254 and 255 turns out to be problematic, due to the special status of 255 as range boundary

[13] Note that the LS (Low Power Schottky) family of 74XX TTL series has a standard propagation delay of 10 ns; thus inverting a signal twice, results with a pulse which is delayed (in respect to a pulse buffered by a follower - which is shown on top part of Fig. 4).

[14] Note that the simplest opamp integrator circuits typically represent inverting integrators

transistors were used here, simply because they may be conceptually more accessible to novices, [15] and it would be instructive to observe their behavior in a circuit, based on just the previously stated assumptions. For that reason, the transistors on the actual board are simply those with the highest β, available to the project at the time (in this case, BC337-25); this is also the reason why a transmission gate is implemented through discrete components, instead of using an integrated bilateral switch (such as 4016 or 4066).

The principle of the Fig. 5 circuit operation is: PWM input is brought to the integrator; during duty cycle, the +5V of PWM input cause a constant current to charge CInt0, which develops a linear ramp voltage $V_{C_{\text{Int0}}}(t)$ (as integral of the constant current). This voltage $V_{C_{\text{Int0}}}(t)$ is buffered through (B) and brought to input of the analog switch structure. When COPY is high, the analog switch turns ON, which should allow (B) to quickly charge C_cpbuf_0 to the same voltage held by CInt0 (i.e., $V_{C_{\text{Int0}}}(t)$). When COPY is low, the analog switch is turned off - so C_cpbuf_0 can *not* discharge, being buffered by (C); and should hold its voltage constant. On the other hand, when CLEAR is high, QInt_Empty0 turns on, and short-circuits CInt0 - forcing its voltage to 0. Thus, the following sequence of events can be pursued: at start of PWM period, CInt0 is charged for as long as the duty cycle is active, and keeps this value off duty cycle; near end of PWM period, COPY is triggered, and C_cpbuf_0 is set to same voltage as CInt0; then COPY is deactivated, which isolates C_cpbuf_0 from CInt0, causing it to hold the last copied value; then CLEAR activates, which discharges CInt0; then finally CLEAR turns off, just in time before next PWM period starts - where integration can start again, by charging a newly emptied CInt0 capacitor.

Applying xBEOP-3 and xBEOP-2 (Fig. 4) to COPY and CLEAR respectively, would ensure that COPY and CLEAR run in sequence just before the end of PWM period. Ultimately, this should result with the SH_OUT signal which is a 62.5 kHz analog SH voltage representation of the 62.5 kHz PWM signal, in the voltage range from 0V to +5V. In AUDIOARDUINO context, SH_OUT would be an oversampled (but jittered - for more, see [1]) reproduction of the original 44.1 kHz audio stream played back by high-level software. Running this SH_OUT signal through low-pass filter, DC-blocking capacitor, and amplifier, should finally result with a audio voltage signal conforming to the *line-level* range of $\mp$1.95V around a zero volt reference.

Note that this specific copy/clear PWM-to-SH process guarantees errors in reproduction for levels above the COPY pulse locations (i.e. above 252). Additionally, most opamps work properly as followers (as on Fig. 5) only when fed by a symmetric power supply ($\mp$Vcc); powering basic opamps like TL074 with a single supply (between GND and Vcc) will introduce additional errors in operation. [16] Most of the experiment videos on [3] have been performed with a single supply of 5V. [17]

5.2 Speaker amp, H-bridge and Class-D

Whereas for 'line-out', we had to consider some form of a PWM to SH conversion, for a speaker output we need to ensure that the signal is properly amplified - and there are ICs that can work either with either type of signals. However, the choice can often be overwhelming for a novice, as PWM parts can often be intended for motor use; therefore the board includes several parts for comparison.

As mentioned, there are three amplifier structures on board. The 'speaker amp', is based on a NATIONAL SEMICONDUCTOR LM386 [18] (marketed as 'Audio Power Amplifier') and intended to drive a small loudspeaker from line-level input. The other two amplifiers are intended to handle raw PWM input: a standalone RAHM BD6211F H-bridge driver chip, marketed as 'full-bridge driver for brush motor applications' with 400 ns dead time; and class-D amplifier based on INTERNATIONAL RECTIFIER's IRS20954S[18] IC, marketed as half-bridge 'protected digital audio driver', with selectable dead time between 15 and 45 ns.

All of these parts make use of an *H-bridge* structure, where transistors are used as switches to cause bidirectional flow of current through a load from a single supply: LM386 and IRS20954S use half-bridge (while BD6211F uses full-bridge) configuration. Note that LM386 as 'class-B push-pull' amp uses BJT (whereas the others, as class-D, feature FET) transistors as bridge switches (IRS20954S also needs additional MOSFETs). Also, BD6211F needs two PWM inputs (in antiphase), which is provided by XOR buffer in Fig. 4. The H-bridge structure typically has an issue when mutually exclusive switches briefly stay turned on together, which causes a short-circuit of the bridge output (alias shoot through); this is usually handled by introducing so-called *dead-time* [16]. Note that BD6211F's dead time of 0.4 µs represents 2.5% of the 16 µs period of the 62.5 kHz PWM signal.

5.3 Analog filters and input preamplification

The filter units can be used for either input or output through wirewrapping. Assuming that the output PWM signal from AUDIOARDUINO has a baseband spectrum up to 20 kHz, and harmonics (with sidebands) starting at 62.5 kHz, an ideally steep low-pass filter with cut-off at 20 kHz would be able to reconstruct the baseband analog signal. The board provides the RC (7, Fig. 2) unit as a passive low-pass filter, designed for a cut-off frequency at 20 kHz - simply as a means for experiencing the influence of the simplest filter design on an audio PWM signal (even if, as a first order filter, it cannot be expected to achieve anything close to ideal reconstruction). The card design used as a base in [2], serves as a source of: the 'bass/treble' filter (8, Fig. 2); and the mic preamplifier (9,11 Fig. 2) unit – while the mixer (10, Fig. 2) unit is a basic inverting op-amp summer; all these units (8-11) are based on generic opamps (i.e. TL074).

[15] as BJT are often used as a starting point in discussing semiconductor transistors

[16] although, there exist pin-compatible alternatives, known as 'rail-to-rail' op-amp

[17] Further notes about the operation and performance of the board's circuits can be found on [3].

[18] Used circuit design taken from part datasheet.

6. CONCLUSIONS

This paper outlined some basic issues in analog I/O for soundcards, primarily by discussing a 'line-level' interface for the AUDIOARDUINO open soundcard system, implemented by the analog I/O board. As the line input was deemed to be manageable by scaling - the focus was mostly on introducing the role of PWM voltage as digital output signal, and it's relationship to analog voltage in context of audio; supported with a basic, first-principles proposal for a simple PWM to (discretized) analog converter utilizing specific capabilities of ATMEGA328.

Further media on [3] documents that this board, in its current form, does not bring about any advantage to the use of AUDIOARDUINO as a soundcard; it is useful only as a subject of introductory study. In particular, the media documentation on [3] aims to provide experiential familiarity to novices with the approached outlined here, even without the need to actually build the board. As such, this paper and project aim to provide a basis for further development - and in that, promote the discussion of DIY digital audio hardware implementations among researchers and hobbyists.

7. ACKNOWLEDGMENTS

The authors would like to thank the Medialogy department at Aalborg University in Copenhagen, for the support of this work as a part of a currently ongoing PhD project.

8. REFERENCES

[1] S. Dimitrov and S. Serafin, "Audio Arduino - an ALSA (Advanced Linux Sound Architecture) audio driver for FTDI-based Arduinos," in *Proceedings of the 2011 conference on New interfaces for musical expression*, 2011.

[2] S. Dimitrov, "Extending the soundcard for use with generic DC sensors," in *NIME++ 2010: Proceedings of the International Conference on New Instruments for Musical Expression*, 2010, pp. 303–308.

[3] ——, "AudioArduino Analog Board homepage," WWW: http://imi.aau.dk/~sd/phd/index.php?title=AudioArduino-AnalogBoard.

[4] www.atmel.com, "Atmel ATmega48A/48PA/88A/88PA/168A/168PA/328/328P datasheet," WWW: http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf, Accessed: 29 Dec, 2010.

[5] arduino.cc, "Arduino homepage," http://www.arduino.cc/.

[6] T. Amyes, *Audio post-production in video and film*. Focal Pr, 1998.

[7] L. Ahlzen and C. Song, *The Sound Blaster Live! Book: A Complete Guide to the World's Most Popular Sound Card*. No Starch Pr, 2003.

[8] G. White and G. Louie, *The audio dictionary*. Univ of Washington Pr, 2005.

[9] R. Donald and T. Spann, *Fundamentals of television production*. Wiley-Blackwell, 2000.

[10] D. Walters, *How to Build a Radio Station*. Lulu. Com, 2006.

[11] mitat.tuu.fi, "Line level audio signal voltage," WWW: http://mitat.tuu.fi/?p=45, Accessed: 5 January, 2011.

[12] D. A. Bohn, "The bewildering wilderness–navigating the complicated and frustrating world of audio standards," *S&VC, September*, vol. 2000, pp. 56–64, 2000, URL: RANE reference,http://www.rane.com/pdf/bewilder.pdf.

[13] J. A. Caffiaux, "A brief review of eia standards in the audio field," *J. Audio Eng. Soc*, vol. 16, no. 1, pp. 21–25, 1968.

[14] K. Nielsen, "A review and comparison of pulse width modulation (pwm) methods for analog and digital input switching power amplifiers," in *PREPRINTS*. AUDIO ENGINEERING SOCIETY, 1997, p. 57pp, URL: http://www.icepower.bang-olufsen.com/files/convention/4446.pdf.

[15] Z. Song and D. Sarwate, "The frequency spectrum of pulse width modulated signals," *Signal Processing*, vol. 83, no. 10, pp. 2227–2258, 2003.

[16] A. Knott, "Improvement of out-of-band behaviour in switch-mode amplifiers and power supplies by their modulation topology," Ph.D. dissertation, Technical University of Denmark, Department of Electrical Engineering, Electronics, 2010. [Online]. Available: http://orbit.dtu.dk/getResource?recordId=270897&objectId=1&versionId=1

[17] A. Knott, G. Pfaffinger, and M. A. Andersen, "On the Myth of Pulse Width Modulated Spectrum in Theory and Practice," in *Audio Engineering Society Convention 126*, 2009.

[18] W. Kester, Analog Devices, Inc. *et al.*, *Data conversion handbook*. Newnes, 2005.

[19] A. Palacherla, "Using PWM to Generate Analog Output," 1997, Microchip Technology, (AN538).

[20] F. T. Agerkvist and L. M. Fenger, "Subjective test of class d amplifiers without output filter," in *117th Audio Engineering Society Convention*, 2004.

[21] J. Dunn, "Jitter: Specification and assessment in digital audio equipment," in *Presented at AES 93rd Convention*. Citeseer, 1992.

[22] J. W. Marshall Leach, "Ideal operational amplifier (op amp) circuits," 2010, ECE3050 Analog Electronics Class notes, Georgia Institute of Technology. WWW: http://users.ece.gatech.edu/mleach/ece3050/sp04/OpAmps01.pdf.