Audio Signal Conditioning Circuits for Arduino Platforms

William D'Andrea Fonseca[1], Artur Zorzo[1], Paulo Mareze[1] and Eric Brandão[1]

[1] *Acoustical Engineering, Federal University of Santa Maria, Santa Maria, RS, Brazil*

{*will.fonseca, artur.zorzo, paulo.mareze, eric.brandao*}@eac.ufsm.br

Abstract

Arduino is an open-source platform that combines the use of microcontrollers with a user-friendly interface. With these microcontrollers, it is possible to read all kinds of sensors as well as manage actuators based on readings and codes previously programmed by the user. For audio transducers, like microphones and loudspeakers, some adjustment circuits are necessary. The signal provided by the microphone is too small in amplitude to be sampled by the microcontroller. Therefore, a preamplifier circuit must be designed. In a similar way, a loudspeaker may not be attached directly to the controller's output without a driver circuit. The primary goal of this paper is to demonstrate and analyze signal conditioning circuits for audio transducers designed to work with Arduino platforms (input and output). Furthermore, a frequency analysis is carried out to determine the expected coloration in the recording signal.

Introduction

Microprocessors, as well as DSP chips, are essential tools in embedded real-time signal and audio processing. They provide the power to perform mathematical operations on input signals. Moreover, they can be used to generate a desired output signal. The circuits shown herein were applied to a real-time adaptive active noise control (ANC) project. In general, they demonstrated good performance. The complete ANC system required several filtering steps in order to generate the *anti-noise signal*. More information about the project can be consulted in Zorzo et al. [1].

In recent years, Arduino [2, 3] has made the technology regarding microcontrollers more accessible and easier for first-time users. Therefore, there is a great demand to exemplify and explain audio processing [4] techniques aimed at using Arduino boards.

For this reason, this paper will introduce audio signal adjustment circuits that can be used with most Arduino boards. Two types of circuits will be shown, an electret microphone preamplifier and an output buffer. With these circuits connected to the board, it is possible to process audio signals acquired from microphones and to generate signals to be emitted from headphones (or small speakers).

Hardware Description

Most of the microcontrollers in the Arduino family have many analog inputs, meaning that they have analog-to-digital converters (ADC) [5]. These converters are key elements when receiving or processing audio signals since they are usually transmitted in terms of alternating current. However, there are only two models of Arduino boards that have digital-to-analog converters (DAC), the Arduino/Genuino Zero and Arduino Due[1] [6]. This paper focuses on the Arduino Due (see Figure 1) because it is capable of greater clock frequency and bit resolution (considering the DAC)

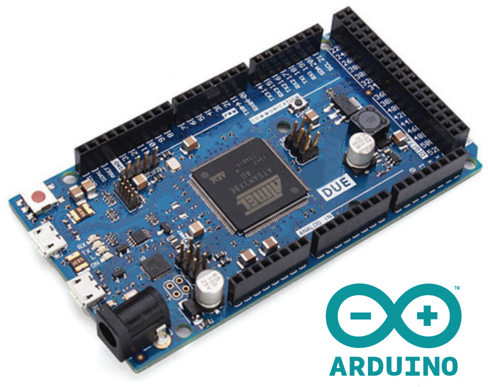

Figure 1: Arduino Due board [2].

The sampling rate can be set depending on the clock frequency and application. If no calculations must be carried out in real time, the sampling rates obtained are usually high and can easily be set to the audio standards, such as 44100 and 48000 Hz. However, in real time processing application, especially in streaming processing, the sampling rate may be reduced depending upon the time that the processor takes to complete the necessary calculations.

Some Arduino Due [2] specifications are :

- ARM Cortex-M3 CPU (32 bits);
- 84 MHz clock;
- 12 analog inputs (12 bit ADC);
- 2 true analog outputs (12 bits DAC);
- 54 digital input/output pins;
- Input voltage (recommended): 7-12 volts;
- Operating voltage: 3.3 volts.

It is important to notice that there are several microcontrollers compatible with the Arduino library and IDE. Some of them can be more powerful and better suited to deal with audio signals.

[1]Other models of Arduino boards have PWM outputs that can simulate an analog output through a digital modulation technique. However, these outputs cannot generate analog signals at higher frequencies in a satisfactory way.

Input Circuits

The analog-to-digital converters (ADC) of microcontrollers typically work on a voltage span from 0.0 V to +3.3 V or +5.0 V. To achieve good sampling results it is important that the signal's peaks are close to the ADC's maximum value (quatization process) [7]. In addition, voltage amplitudes above this threshold may damage the controller's input port and/or generate undesired harmonics. Therefore, a signal adjusting circuit is necessary to guarantee a suitable sampling of audio signals as well as to protect the system.

If it is desired to connect a microphone's output to the board, a preamplifier circuit (or simply preamp) must be designed. The circuit design depends heavily on the microphones' characteristics. For electret microphones, a preamp circuit based on an operational amplifier (or opamp) was designed, as can be seen in Figure 2. This approach is used to amplify the signal out of a JLI-61A microphone capsule [8], see Figure 3 (for the active noise control system aforementioned).

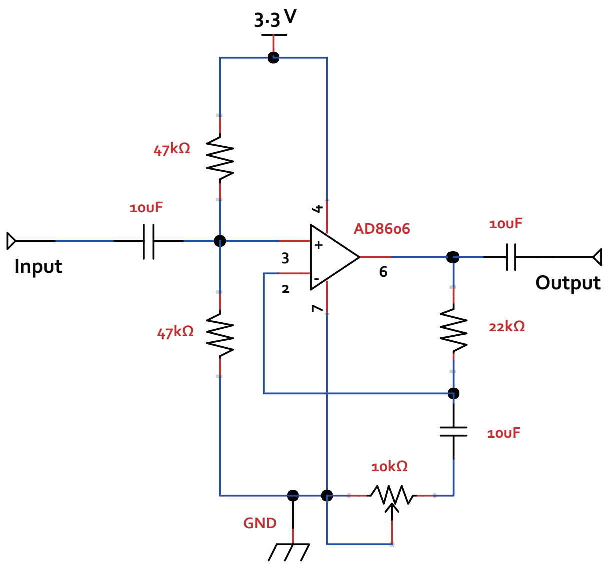

Figure 2: Preamplifier project for electret microphones.

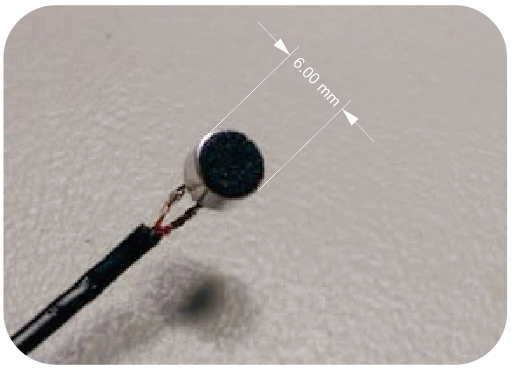

Figure 3: JLI-61A electret microphone capsule.

Since the input ports in the controller can only sample positive voltage levels, an offset must be introduced, as exemplified in Figure 4. The 47 kΩ resistors in the preamplifier circuit are responsible for providing the offset centered at half the supply voltage.

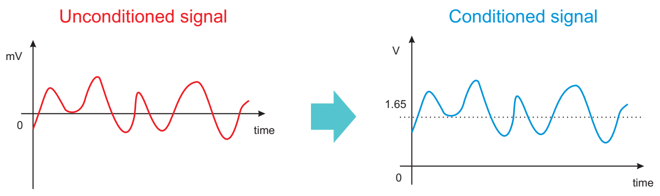

Figure 4: Offset and amplification applied to the input signal.

This circuit is a modified version of Andy Collinson's design [9]. The original operational amplifier was substituted by a precision, low-noise and rail-to-rail version – AD8606. This enables the amplifier system to be powered with the same +3.3 V (or +5.0 V) source that powers the processor inside the board[2]. This opamp switch has rendered low distortion and clipping protection.

Most of the common opamps are unable to reach an output voltage as high as the power supply. The maximum they can reach is commonly 3 or 4 volts below the supply, which becomes a huge problem when trying to build a system that works on low-voltage batteries. Due to this effect, a rail-to-rail opamp was chosen, meaning that it can reach the full voltage range of the supply. Another advantage of the AD8606 opamp [10] is its single-supply operation, avoiding the need of a symmetrical power supply.

The circuit was simulated from 20 Hz to 20 kHz (audible frequency range) in a circuit modeling and simulation software. The frequency and phase response curves can be observed in Figure 5. The simulation was carried out with the highest possible gain option (which depends on the position of the potentiometer).

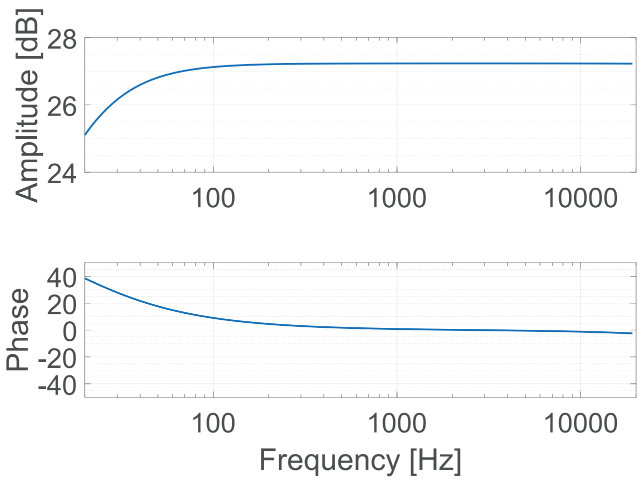

Figure 5: Simulation of the frequency and phase responses of the preamplifier from 20 Hz to 20 kHz (audible frequency range).

From the simulation results, it can be seen that the frequency response does not vary more than 3 decibels over a range from 20 Hz to 20000 Hz approximately.

[2]The board voltage supply is usually a few volts higher than the real processor's supply. Some voltage regulators in the board are responsible for the voltage reduction and regulation. The Arduino Due board, for example, can be powered with 7-12 V, but only 3.3 V reaches the ARM processor. In addition, there are +3.3 V and +5.0 V extra ports on the board to power other hardware.

This frequency range is usually enough for most audio and signal processing applications. However, if a larger frequency span is desired the circuit might have to be redesigned.

Before connecting the preamplifier to the board, an anti-aliasing filter must be designed according to the goal of the system and relative frequency range needed. Without this filter between (or before) the amplifier and the analog inputs of the board, the system will provide unreliable results and might work incorrectly (spectral folding effects or aliasing effects). There are several low-pass integrated circuits, such as the MAX 291/292/295/296 [11], that can be readily used as anti-aliasing filters. Elementary low-pass filtering (e.g. 2nd order RC filter) may also present effective results. More information about the anti-aliasing filter [12] and how to design it can be found in the application texts by Baker [13, 14].

Figure 6 depicts the prototype board containing the input and output circuits. In this project, a dual AD8606 chip was used. This made it possible to integrate both circuits into a single universal PCB board.

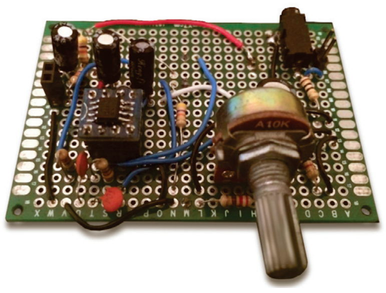

Figure 6: Prototype board containing the input and output circuits.

Output Circuits

Loudspeakers, in general, have low resistance value [15]. Therefore, if the outputs of the controller are connected directly to a loudspeaker (or headphone speaker), the current drawn will be too much for the output ports, i.e. enough to damage them.

A solution to this problem is to connect a voltage buffer to the output. With this configuration, the necessary current to power the speakers will be drawn directly from the power source. The voltage buffer approach specifically designed to power headphones can be seen in Figure 7. This project was proposed by the manufacturer of the opamp model used in this study. The complete description of the AD8606 and additional circuits can be seen in its datasheet [10].

The simulated frequency response of the circuit in Figure 7 can be observed in Figure 8. The -3 dB point occurs in the lower frequency of 31 Hz, demonstrating its linearity for the audible frequency range.

Besides the voltage buffer, the design also contains a high-pass filter and decoupling capacitors to prevent DC currents and frequencies below the loudspeaker's capacity (that would interfere during its operation).

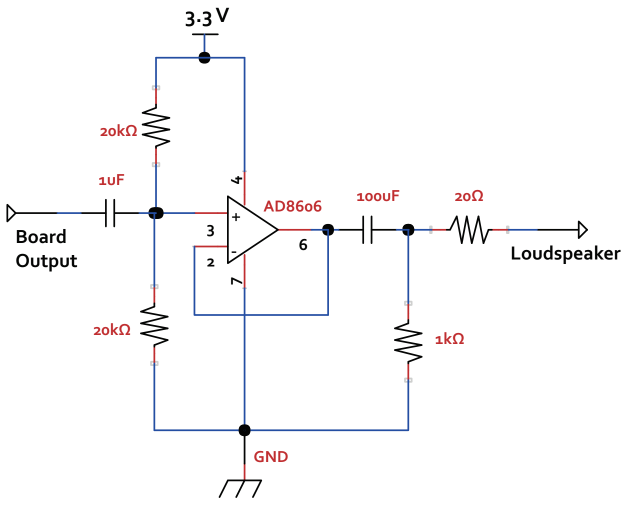

Figure 7: Headphone driver circuit (voltage buffer).

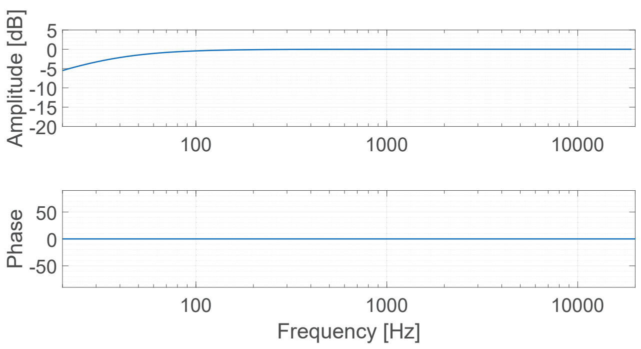

Figure 8: Voltage buffer frequency response (see Figure 7)

Final Considerations

The Arduino Due board is a suitable platform for audio application (input and output). Nevertheless, care must be taken regarding its limits and sampling frequency.

When learning to use microcontrollers, sometimes people can be careless and end up feeding or draining too much current (or voltage) to/from the ports. Such a mistake would probably lead to damage in the boards' inputs and/or outputs. The circuits shown herein are reliable and safe starters for students and professionals who desire to work with audio signal processing with microcontrollers. For special projects, these circuits can be modified (or tuned) to suit the desired functionality.

For virtual prototyping (simulation), there are several options on the market. Some are basic circuit simulators, and others include Arduino processing (making combined hardware and software simulation possible).

For prototype manufacturing, a good first starter is a breadboard (which eases the connection of components). Evolving the project, the next approach would be to build it into a universal PCB board, which may reduce sensitivity to external interference. Finally, with an established design, a dedicated PCB is indicated. Proper housing for the circuit may also reduce unwanted electromagnetic interference.

All the diagrams shown were designed for the Arduino Due board. However, they probably can be used with other microcontrollers without major changes (for example, some Teensy boards [16]). It is always important to check the processors' maximum voltages and currents (of each port) before connecting anything to them. This procedure will prevent undesired conditions during hardware use.

References

[1] Zorzo, A., Fonseca, W. D'A., Brandao, E. and Mareze, P. (2017). Design and analysis of a digital active noise control system for headphones implemented in an Arduino compatible microcontroller. In: *7º Simpósio de Processamento de Sinais (SPS)*, Campinas, SP, Brazil. Available at: `http://bit.ly/ANC-Arduino`.

[2] Arduino Website.
Available at: `https://www.arduino.cc`.
Accessed Jan. 2018.

[3] Perea, F. (2015). Arduino Essentials. Packt Publishing.

[4] Shin K. and Hammond J. (2008). Fundamentals of signal processing for sound and vibration engineers. England: John Wiley & Sons.

[5] Williston, K. (2009). Digital Signal Processing: World Class Designs. Burlington, Mass.: Newnes.

[6] Arduino. analogWriteResolution() - Arduino Function Reference. Available at: `https://www.arduino.cc/en/Reference.AnalogWriteResolution`.
Accessed Jan. 2018.

[7] Tan, L. (2008). Digital Signal Processing: Fundamentals and Applications. Amsterdam: Academic Press.

[8] JLI Electronics. Omnidirectional Back Electret Condenser microphone Cartridge JLI-61A (Datasheet). Available at: `http://bit.ly/JLI61A`. Accessed Jan. 2018.

[9] Collinson, A. (2017). Op-Amp Microphone Preamp. Available at: `http://bit.ly/OpAmp-Collinson`. Accessed Jan. 2018.

[10] Analog Devices. (2017). AD8606 - Precision, Low Noise, CMOS, Rail-to-Rail, Input/Output Operational Amplifiers (Datasheet: Rev. O). Available at: `http://bit.ly/AD8606-AnDev`. Accessed Jan. 2018.

[11] Maxim Integrated. (2010) MAX291/MAX292 /MAX295/MAX296: 8th-Order, Lowpass, Switched-Capacitor Filters (Datasheet: Rev. 5). Available at: `http://bit.ly/MAX291-MAX296`. Accessed Fev. 2018.

[12] Kester, W. (2009). What the Nyquist Criterion Means to Your Sampled Data System Design. In: *Analog Devices Tutorial (MT-002, Rev. A)*. Available at: `http://bit.ly/Kester-AD-MT002`. Accessed Fev. 2018.

[13] Baker, B. C. (1999). Anti-Aliasing, Analog Filters for Data Acquisition Systems In: *Microchip Application Note (AN699)*. Available at: `http://bit.ly/Baker-AN699`. Accessed Fev. 2018.

[14] Baker, B. C. (2015). Designing an anti-aliasing filter for ADCs in the frequency domain. In: *Analog Applications Journal (Texas Instr.: AAJ 2Q 2015)*. Available at: `http://bit.ly/Baker-TI-AAJ2Q2015`. Accessed Jan. 2018.

[15] Borwick, J. (2001). Loudspeaker and Headphone Handbook. Oxford: Wright.

[16] Stoffregen, P. J. and Coon, R. C. (2017). PJRCs: Teensy USB Development Board. Available at: `http://bit.ly/Teensy36`. Accessed Fev. 2018.