

Adaptive Noise Controller

Abstract

This project aims to design and implement a **real-time adaptive noise controller** using an **STM32F446RE** microcontroller. The goal of the system is to attenuate unwanted environmental noise that distorts a desired audio signal, while preserving the underlying speech. The noise cancelling technique used is based on the **Least Mean Squares (LMS)** algorithm implemented as a finite impulse response (**FIR**) filter.

System Overview

The system contains two analog microphones on a breadboard and is separated by approximately 15 cm. The primary microphone captures the desired speech signal combined with ambient noise, while the reference microphone captures a correlated noise signal that is intended to be canceled.

Both microphones are interfaced directly to the STM32F446RE's internal 12-bit *analog-to-digital converter* (ADC). The primary microphone signal is treated as the desired signal $D(n)$, and the reference microphone signal is used as the input $X(n)$ to the adaptive filter.

Audio signals are sampled at 44.025 kHz to satisfy the Nyquist criterion for the human auditory range, which extends up to 20 kHz. The ADC operates using the peripheral clock (PCLK) at 84 MHz with a prescaler of 6, yielding a 14 MHz ADC clock. With a total of 159 clock cycles per conversion, the resulting sampling frequency closely matches the required audio rate. Since both microphones are sampled through a shared ADC buffer, the effective sampling rate per channel is halved. *Direct Memory Access* (DMA) is used to efficiently transfer ADC data into memory with minimal CPU overhead, enabling real-time performance.

Adaptive noise cancellation is achieved using an LMS FIR filter, implemented using the ARM CMSIS-DSP library. FIR filters were chosen due to their inherent stability and predictable behavior in real-time audio systems. The filter output $Y(n)$ is subtracted from the desired signal $D(n)$ to generate an error signal $E(n)$, which is fed back to update the filter coefficients iteratively. This adaptive process allows the system to track time-varying noise characteristics and continuously improve noise suppression.

Captured audio samples are processed in buffers of 128 samples, resulting in a latency of approximately 2.9 ms per processing block. This design balances computational responsiveness, and ensures total system latency remains below 10

ms. Internally, ADC samples are scaled to 32-bit floating-point values in the range $[-1, 1]$ for digital signal processing, then converted back to 16-bit integers for audio output via a UDA1334A digital-to-analog converter using the I2S interface.

For demonstration and debugging purposes, processed audio data is transmitted over UART to a host computer, where it is logged and visualized using a Python script. This allows real-time observation of the adaptive filtering performance and verification of noise cancellation behavior.

Overall, this project showcases an embedded adaptive noise cancellation system, integrating analog input, real-time digital signal processing, DMA-based data acquisition, and audio output. The implementation highlights key tradeoffs between latency, buffer size, and computational complexity in real-time embedded audio applications.

Design Calculations & Formulas

- Analog Microphones (AM1, AM2)
 - AM1 = Primary Mic
 - Voice + Noise
 - AM2 = reference signal Mic
 - Noise to cancel
 - Separated by 10 cm
- Two ADC INPUTs (ADC1, ADC2)

LMS FIR FILTER Formula

X(n) = Reference Noise (Noise Signal) (First ADC Channel Input)

D(n) = Desired Signal (Primary Signal + Noise) (Second ADC Channel Input)

Y(n) = Current Filter Output

E(n) = Error calculation = $D(n) - Y(n)$. Feedforward error term.

ADC Sampling

Using Right Analog Mic for Reference X(n)

Using Left Analog Mic for Primary Signal (noise + speech) D(n)

ADC1_IN0 = D(n)

ADC1_IN1 = X(n)

44khz of sampling frequency is needed as human hearing range goes until 20 KHz (Nyquist Frequency of Hearing)

Using a 12 bit resolution ADC and it uses the PCLK as clock source (84 MHz).
With a prescaler of 4, we have ADC Clock = 84 MHz / 4 = 21 MHz.

ADC sampling frequency is calculated from taking the ADC input CLOCK and the time taken to do analog to digital conversion .

Conversion Clock Cycles = #Sampling Clock_Cycles + #Conversion_Clock_Cycles
144 sampling cycles + 15 conversion cycles = **159** Clock Cycles

21 MHZ / 159 Clock cycles = **44.025 KHz Sampling Frequency of ADC**

Since one buffer is used to get data from two channels, reference mic and desired mic, the ADC sampling frequency is halved for each input channel.

ADC Resolution downscaling

[-1,1] 32 bit float then turn back into 16 bit integer for i2s. (16 bit data frame)

LMS FIR Filter cont...

Leveraging the CMSIS-DSP library from ARM to perform the FIR Filtering to cancel desired noise. FIR Filters are more stable (no feedback) and can more easily change coefficients in real-time (safety as no accumulation of error).

Latency Calculation

ADC → DMA (half word → since ADC is 12 bit and fits in 16 bit naturally)
128 samples (buffer size) * 1/44.1KHZ = 2.9 ms latency between each time you check dma,
The more samples you have, the more latency you get.

Tradeoff: More latency = more time to do other tasks. However, for Audio's real time constraint, < 10ms sampling latency is good enough.