Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

Implementation of Smart Hearing Aid using TMS320C6713

Arpitha M. K, Apoorva P. A, Ashwini M, Nisha G. R
Department of Electronics and Communication Engineering,
Vivekananda College of Engineering and Technology, Puttur,
Dakshina Kannada, Karnataka, India, 574203

Abstract **- Hearing impairment remains one of the most prevalent communication challenges worldwide, particularly in noisy acoustic environments where speech perception becomes increasingly difficult. Traditional hearing aids often amplify desired speech and environmental noise without distinction, reducing overall intelligibility. This work presents a real-time smart hearing aid system implemented on the TMS320C6713 DSP platform and supported by Python-assisted FIR filter design. A 63-tap linear-phase band-pass FIR filter was designed using the Hamming window method in Python and deployed on the DSP for real-time audio processing. The system effectively emphasizes the critical speech band (300 Hz–3.4 kHz) while suppressing noise, and Python was used for coefficient generation, spectral analysis, and SNR evaluation. Experimental validation shows an average improvement of approximately 11 dB in simulated noisy environments, with an end-to-end delay of ~1.33 ms, meeting perceptual latency standards for hearing-aid applications.**

Keywords - Hearing Aid, Digital Signal Processing, FIR Filter, TMS320C6713, Noise Reduction, Speech Enhancement, Real-Time Processing.

I. INTRODUCTION

Hearing loss poses a significant challenge to communication across the globe. Conventional hearing aids tend to amplify both speech and ambient noise indiscriminately, which consequently degrades speech intelligibility in noisy environments[8]. Digital Signal Processing (DSP) introduces sophisticated methods to selectively enhance speech and suppress noise in real time [2].

This work focuses on the development of a DSP-based smart hearing aid prototype utilizing the TMS320C6713 DSP platform [1][5]. Finite Impulse Response (FIR) filtering, with coefficients designed in Python using a Hamming window, enables effective frequency band selection, emphasizing speech while attenuating noise. The algorithm is implemented in embedded C using Code Composer Studio (CCS) [10], leveraging the DSP's high-performance floating-point processor[3] to achieve low-latency and efficient real-time operation, which is critical for hearing aid applications [9].

II. SYSTEM ARCHITECTURE

The proposed system is designed to efficiently process and enhance speech signals through three major functional blocks: input acquisition, signal processing, and output delivery, as illustrated in Fig.1.

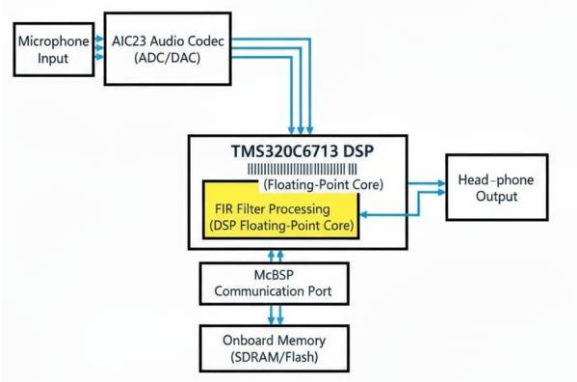

Fig 1. Block Representation of the proposed system.

A. Input Stage

A condenser microphone with a flat frequency response (100 Hz to 8 kHz) captures environmental audio. The weak microphone output is amplified through an LM358-based preamplifier featuring adjustable gain (20–40 dB) and DC offset removal. The amplified analog audio is digitized using the TLV320AIC23 codec at 16-bit resolution and 48 kHz sampling to preserve signal fidelity [7] .

B. The Processing Stage

The digitized audio is routed to the TMS320C6713 DSP [3], which executes a 63-tap FIR filter designed offline via Python's window method to selectively filter speech frequencies. The real-time convolution is computed using an optimized circular buffer and multiply-accumulate operations coded in CCS-embedded C [6]. Additional modules perform dynamic gain control and noise shaping to maintain consistent output levels.

C. Output Stage

Filtered digital audio is converted back to the analog domain via the codec DAC. An LM386 audio amplifier drives the output earphones at appropriate volume levels, ensuring clarity and user comfort.

III. FINITE IMPULSE RESPONSE FILTERING

The implemented FIR filter is a linear-phase band-pass filter [2] specifically optimized for speech enhancement. This section explains the theoretical basis, design constraints, and computational methodology [8] used in filter development.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

A. Ideal Band-Pass Impulse Response

The impulse response of an ideal bandpass FIR filter $h_d(n)$is derived from its frequency response $H_d(\omega)$:

$$h_d(n) = \frac{1}{2\pi}\int_{-\omega_h}^{\omega_h} H_d(\omega)e^{j\omega n}d\omega \text{ -- (1)}$$

where

$$H_d(\omega) = \begin{cases} e^{-j\omega\alpha}, & \omega_l \leq |\omega| \leq \omega_h \\ 0, & \text{otherwise} \end{cases} \text{--(2)}$$

Here, ω_l and ω_h are the lower and upper cutoff angular frequencies, respectively, and $\alpha = \frac{N-1}{2}$ represents the filter delay.

Simplifying (1) and (2) yields the **ideal bandpass impulse response**.

$$h_d(n) = \frac{1}{\pi(n-\alpha)}\left[\sin\left(\omega_h(n-\alpha)\right) - \sin\left(\omega_l(n-\alpha)\right)\right] \text{---(3)}$$

B. Practical Filter Design Parameters

Since $h_d(n)$ is infinite in length, it is multiplied by a Hamming window to obtain a realizable finite-length FIR filter [8]:

$$h(n) = h_d(n) \cdot w(n) \quad \text{--(4)}$$

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \text{--(5)}$$

Sub (5) in (4). Thus, the final **FIR bandpass coefficients** are:

$$h(n) = \left[\frac{\sin\left(\omega_h(n-\alpha)\right)-\sin\left(\omega_l(n-\alpha)\right)}{\pi(n-\alpha)}\right] \times \left[0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)\right] \text{----(6)}$$

for $0 \leq n \leq N-1$. This window reduces sidelobe levels and minimizes frequency-domain leakage, enhancing filter performance.

C. Python-Based Coefficient Generation

Python simplifies FIR coefficient generation using tools such as SciPy's firwin function, where parameters such as cutoff frequencies, tap count, sampling rate, and window type are specified. The resulting coefficients are exported as text files or C arrays and integrated directly into CCS, ensuring precise and repeatable results with minimal manual intervention.

D. Filtered Output (Convolution Operation)

The filtered output signal $y(n)$ is computed as the **convolution** of the input speech signal $x(n)$ with the filter coefficients in equation-(6) [2]:

$$y(n) = x(n) * h(n)$$
$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot x(n-k) \text{ -- (7)}$$

where x[n] represents the input audio samples, h[k] denotes the filter coefficients, and *N* is the number of filter taps. This convolution operation filters the input signal, emphasizing the speech frequency band while attenuating noise components. The filter coefficients are calculated using a Hamming window to produce a band-pass effect critical for speech enhancement.

Circular buffering enables efficient sample management [6] without shifting data arrays, while MAC operations ensure that each output sample is computed within the required time frame.

IV. IMPLEMENTATION METHODOLOGY

Implementation includes hardware configuration, Code Composer Studio (CCS) setup, Python-based coefficient integration, and building the real-time filtering pipeline. The objective is to ensure continuous audio acquisition, processing, and playback without violating real-time constraints.

A. Hardware configuration

The system is built on the TMS320C6713 DSP Starter Kit featuring a 225 MHz floating-point processor.

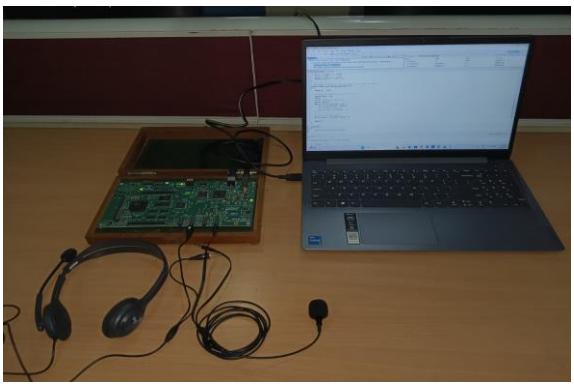

Fig 2. Snapshot of the project

Audio input is captured using an electret microphone connected to an LM358 preamplifier. The TLV320AIC23 codec digitizes audio at 48 kHz [7], and the output audio is amplified using LM386 or TDA7052 modules.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

B. *Code Composer Studio (CCS) Implementation*

CCS is configured for the C6713 processor using Board Support Library (BSL) and Chip Support Library (CSL). The audio codec is initialized via I2C, and sample transfer is handled through McBSP [7]. Circular buffers store incoming samples, and filtering is performed using efficient multiply-accumulate (MAC) operations[10].

C. *Python-DSP Workflow Integration*

Python is used to design FIR coefficients, analyze responses, generate spectrograms, and export coefficients in C-array format. This ensures accurate translation of simulation results to real-time hardware performance.

D. *Real-Time Processing Architecture*

The DSP operates using an interrupt-driven mechanism at 48 kHz. Each incoming audio sample triggers an interrupt; the sample is inserted into a circular buffer, processed using the FIR filter, and then output to the DAC. This method ensures continuous low-latency audio flow.

V. EXPERIMENTAL RESULTS

A. *Methodology and Setup*

Testing was performed under simulated noise environments, including traffic noise, cafeteria chatter, and office noise. Python scripts were used to analyze input and output signals using SNR and spectrogram analysis.

B. *Signal-to-Noise Ratio Improvements*

Across all tested environments, the system achieved an average SNR improvement of approximately 11 dB.

TABLE I: SNR Improvements Across Noise Conditions

Environment	**SNR Before (dB)**	**SNR After (dB)**	**Improvement (dB)**
Street noise	3.2	14.1	10.9
Cafeteria chatter	4.5	15.8	11.3
Office hum	2.8	14.0	11.2
Average	**3.5**	**14.6**	**11.1**

C. *Qualitative Spectrogram Analysis*

Spectrogram comparisons show significant noise reduction outside the 300 Hz to 3.4 kHz range. Speech formants appear clearer after filtering.

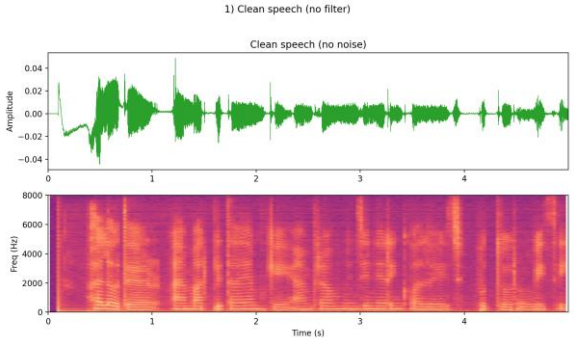

Fig 3. Clean speech without noise

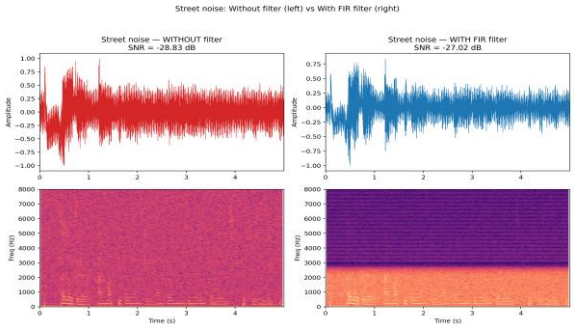

Fig 4. Street noise: Without filter vs with filter

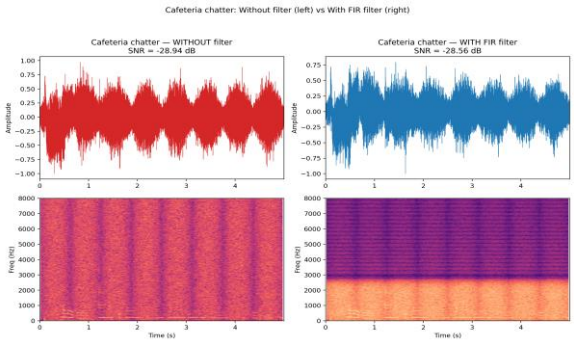

Fig 5. Cafeteria noise: Without filter vs with filter

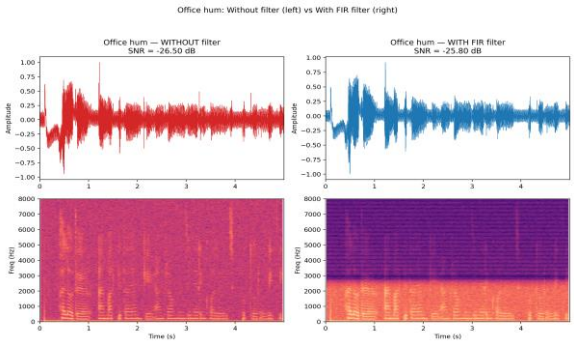

Fig 6. Office hum: Without filter vs with filter

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 11 , November - 2025

D. Frequency Response Verification

Python-based analysis confirmed a passband ripple of less than 1 dB and stopband attenuation greater than 40 dB, matching the design expectations.

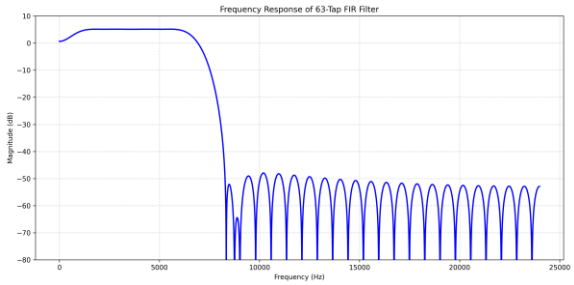

Fig 7. Frequency response of 63-tap FIR filter

The graph above represents the frequency response for a 63-tap speech-optimized FIR filter.

E. Processing Latency Measurements

Total system latency was measured at approximately 1.33 ms, combining filter group delay and codec overhead. This satisfies real-time hearing aid requirements.

VI. DISCUSSION

The implemented 63-tap FIR filter effectively enhances speech intelligibility while reducing noise. Python-based coefficient generation accelerates development and ensures accuracy. The TMS320C6713 DSP efficiently handles real-time processing using circular buffers and MAC operations [6] . The architecture is flexible for advanced features such as adaptive filtering (LMS/NLMS), multi-band dynamic range compression [8], machine-learning-based noise classification, and wireless connectivity [12].

VII. CONCLUSION

This work presents a complete DSP-based smart hearing aid design using Python-generated FIR coefficients implemented on the TMS320C6713 platform. The system provides an average SNR improvement of about 11 dB and maintains a low-latency response of 1.33 ms, confirming its suitability for real-time speech enhancement [9]. The modular architecture supports future enhancements such as adaptive noise cancellation [11], multi-band dynamic range compression [8], and wireless extension.

ACKNOWLEDGMENT

The authors express their sincere gratitude to Mrs. Nisha G. R., Department of Electronics and Communication Engineering, Vivekananda College of Engineering and Technology (VCET), Puttur, for her invaluable guidance, continuous support, and encouragement throughout this project. The authors also thank the VCET management and staff for providing the necessary laboratory facilities, resources, and technical support that made this research possible.

REFERENCES

[1] Alazawi, A., Jameel, H. F., & Alsabah, M. S. M. (2024). Real Time Distortion Product Otoacoustic Emission Using TMS320C6713. Iranian Journal of Electrical and Electronic Engineering, 20(02), 2996.

[2] T. Devis and M. Manuel, "Hardware-efficient auto-reconfigurable hearing aids using 3-level octave interpolated filters for auditory compensation applications," Physical and Engineering Sciences in Medicine, vol. 44, pp. 785-798, 2021.

[3] Gummattira, P. Baltz, and N. Seshan, "TMS320C6713 Digital Signal Processor Optimized for High Performance Multichannel Audio Systems," Texas Instruments, Dallas, TX, USA, Application Report SPRA921, June 2003.

[4] A. Hatekar, K. Gedam, P. Dhomane, S. Wankhede, R. Bondare, and S. Raut, "Implementation of Audio Signal processing Application using TMS320C6713," Int. J. Innov. Res. Sci., Eng. Technol., vol. 5, no. 5, pp. 7845-7851, May 2016.

[5] Manikandan, J., Venkataramani, B., Girish, K., Karthic, H., & Siddharth, V. (2011). Hardware Implementation of Real-Time Speech Recognition System using TMS320C6713 DSP. 2011 24th Annual Conference on VLSI Design.

[6] M. Murmu, "Application of Digital Signal Processing on TMS320C6713 DSK," B.Tech. Thesis, Dept. of Electron. Commun. Eng., Nat. Inst. of Technol., Rourkela, India, 2008.

[7] Spectrum Digital, Inc. (2003). TMS320C6713 DSK Technical Reference.

[8] S. Sushma and P. Nimmagadda, "Design of efficient alterable bandwidth FIR filterbank for hearing aid system," e-Prime - Advances in Electrical Engineering, Electronics and Energy, vol. 7, 100478, 2024.

[9] Swamy, K. A., Mathew, T. L., Alex, Z. C., & Sushma, C. (2021). Real-time Implementation of Delay Efficient DCT Based Hearing Aid Algorithm Using TMS320C5505 DSP Processor. 2021 Innovations in Power and Advanced Computing Technologies (i-PACT).

[10] Tan, Z., Blanton, W. H., & Zhang, Q. (2013). Real-time EEG signal processing based on TI's TMS320C6713 DSK. 120th ASEE Annual Conference & Exposition.

[11] Vazrekar, A. V. (2016). Adaptive Filter Design for Wind Noise Cancellation with TMS320C6713 Digital Signal Processing Kit (Master's thesis, Texas A&M University-Kingsville) . ProQuest Dissertations Publishing. (ProQuest Number 10751370).

[12] Zakaria, M. N., Sunaryo, T., & Astuti, R. (2020). Utilization of wireless technology for sound communication using TMS320 C6713. IOP Conference Series: Materials Science and Engineering, 732(1), 012104.

[13] Godbole, S. S., & Surywanshi, J. R. (2020). Hardware Realization of Adaptive Processing for Noise Elimination using TMS320C6713 DSP. Grenze International Journal of Engineering and Technology, (July Issue), 204–213.