Sistem dedicat pentru achiziția și procesarea avansată a semnalelor audio folosind un ESP32

Cristi Miloiu Cristian Colăcel

I. Introducere

În acest proiect este prezentat un sistem dedicat pentru achiziția și procesarea semnalelor audio în timp real folosind microcontrolerul ESP32, un SoC (System-on-Chip) de 32 biți cu conectivitate Wi-Fi și Bluetooth. Sistemul efectuează achiziția semnalului, preprocesarea, analiza spectrală și transmiterea datelor către un server cloud unde sunt afișate într-un dashboard interactiv.

Necesitatea unui astfel de sistem apare în aplicații precum monitorizare de mediu, analiză acustică, control vocal sau prototipuri IoT. Am ales ESP32 datorită raportului optim între performanță, conectivitate și cost. Procesarea digitală a semnalelor (DSP) permite extragerea informațiilor utile din semnalele audio, eliminarea zgomotului și identificarea componentelor spectrale de interes [4].

II. Fundamente Teoretice

A. Conversia Analog-Digitală (ADC)

Conversia analog-digitală reprezintă procesul de transformare a unui semnal continuu în timp într-o secvență de valori discrete. ESP32 utilizează un convertor ADC de tip SAR (Successive Approximation Register) cu rezoluție de 12 biți [1].

Arhitectura SAR funcționează prin aproximări succesive: la fiecare pas de conversie, convertorul compară tensiunea de intrare cu o tensiune de referință generată intern, ajustând succesiv biții rezultatului până la obținerea valorii finale [2]. Formula de conversie este:

$$D = \frac{V_{in}}{V_{ref}} \times (2^n - 1) \tag{1}$$

unde D este valoarea digitală, V_{in} tensiunea de intrare, V_{ref} tensiunea de referință și n rezoluția în biți.

Pentru o rezoluție de 12 biți, valoarea maximă este 4095, corespunzând intervalului de tensiune 0 până la $3.3\,\mathrm{V}$. Rata de eșantionare este determinată de teorema Nyquist-Shannon, care stipulează că frecvența de eșantionare trebuie să fie cel puțin dublul frecvenței maxime a semnalului de interes pentru a evita fenomenul de aliasing [5].

B. Transformata Fourier Rapidă (FFT)

Transformata Fourier Discretă (DFT) permite descompunerea unui semnal în componentele sale spectrale. Pentru un semnal discret $x[n]$ cu N eșantioane, DFT este definită ca [3]:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi kn}{N}}, \quad k = 0, 1, \ldots, N-1 \tag{2}$$

Calculul direct al DFT necesită $O(N^2)$ operații, ceea ce devine prohibitiv pentru N mare. Algoritmul FFT Cooley-Tukey, publicat în 1965, reduce complexitatea la $O(N \log N)$ prin exploatarea simetriilor și periodicității factorilor de rotație (twiddle factors) [3].

Algoritmul radix-2 divide recursiv transformata în două subtransformate de dimensiune $N/2$:

$$X[k] = X_e[k] + W_N^k \cdot X_o[k] \tag{3}$$

$$X[k + N/2] = X_e[k] - W_N^k \cdot X_o[k] \tag{4}$$

unde X_e și X_o sunt transformatele eșantioanelor pare, respectiv impare, iar $W_N^k = e^{-j2\pi k/N}$ reprezintă factorii de rotație [4].

În implementarea noastră, am utilizat 128 de eșantioane la o rată de $16\,\mathrm{kHz}$, rezultând o rezoluție spectrală de aproximativ $125\,\mathrm{Hz}$ per bin. Spectrul util acoperă banda de la 0 la $8\,\mathrm{kHz}$, împărțită în 9 benzi optimizate pentru analiza vocii umane.

C. Filtre Digitale

Filtrele digitale sunt sisteme care modifică caracteristicile spectrale ale semnalelor. În implementarea noastră, am optat pentru filtrare în domeniul frecvenței, aplicată direct pe benzile spectrale obținute din analiza FFT [4].

1) Filtrare în Domeniul Frecvenței: Spre deosebire de filtrele clasice în domeniul timpului (FIR și IIR), care operează eșantion cu eșantion, filtrarea în domeniul frecvenței modifică direct amplitudinile componentelor spectrale. Această abordare este eficientă când analiza spectrală este deja necesară pentru vizualizare.

Pentru fiecare bandă de frecvență i cu frecvența centrală f_i, aplicăm o funcție de transfer $H(f_i)$ care poate lua valorile 0 (atenuare completă) sau 1 (trecere), în funcție de tipul filtrului și frecvența de tăiere configurată.

2) Tipuri de filtre implementate: Am implementat în sistem următoarele tipuri de filtre:

- **Filtru trece-jos (Low-Pass)**: Atenuează complet benzile cu frecvența centrală superioară frecvenței de tăiere f_c. Matematic: $H(f_i) = 1$ dacă $f_i \leq f_c$, altfel $H(f_i) = 0$.

- **Filtru trece-sus (High-Pass)**: Atenuează complet benzile cu frecvența centrală inferioară lui f_c. Matematic: $H(f_i) = 1$ dacă $f_i \geq f_c$, altfel $H(f_i) = 0$.
- **Filtru trece-bandă (Band-Pass)**: Permite trecerea doar a benzilor cu frecvența centrală între f_{c1} și f_{c2}. Matematic: $H(f_i) = 1$ dacă $f_{c1} \leq f_i \leq f_{c2}$, altfel $H(f_i) = 0$.
- **Voice Boost**: Amplificare selectivă cu factor configurabil (implicit 2.0) pentru benzile din intervalul vocal (de la 500 Hz la 2500 Hz), corespunzând benzilor 2-5 din cele 9 benzi spectrale.
- **Bypass**: Dezactivează filtrarea, permițând trecerea nemodificată a tuturor benzilor.

3) Tehnici Suplimentare de Procesare: Pe lângă filtrarea spectrală, am implementat și următoarele tehnici de îmbunătățire a semnalului:

Noise Gate: Atenuează complet semnalele sub un prag configurabil, eliminând zgomotul de fond în perioadele de liniște. Pragul este ajustat diferențiat pentru benzile joase (mai agresiv) și benzile vocale (mai permisiv).

Netezire exponențială (Exponential Smoothing): Reduce variațiile bruște ale volumului folosind formula:

$$y[n] = \alpha \cdot x[n] + (1 - \alpha) \cdot y[n - 1] \tag{5}$$

unde α este factorul de netezire (implicit 0.5).

Medie mobilă (Moving Average): Filtrare suplimentară pe ultimele 3 eșantioane pentru stabilizarea afișării volumului.

Calibrare automată: Sistemul măsoară automat zgomotul de fond în primele 30 de cicluri de achiziție și îl scade din semnalul util pentru fiecare bandă.

D. Protocolul WebSocket

WebSocket este un protocol de comunicare care permite canale bidirecționale full-duplex peste o singură conexiune TCP, standardizat în RFC 6455 [6]. Spre deosebire de HTTP tradițional, WebSocket menține conexiunea deschisă, permițând schimbul de mesaje în timp real cu latență minimă.

Conexiunea se stabilește printr-un handshake HTTP care include header-ul "Upgrade: websocket". După confirmarea serverului (cod 101 Switching Protocols), conexiunea trece la protocolul WebSocket. Mesajele sunt transmise în frame-uri care pot conține date text sau binare [7].

Am ales WebSocket pentru avantajele sale în aplicațiile IoT: overhead redus (doar 2 până la 10 octeți per frame față de sute de octeți pentru HTTP headers), comunicare bidirecțională fără polling, și menținerea stării conexiunii [6].

III. Hardware

A. Microcontrolerul ESP32

ESP32, dezvoltat de Espressif Systems, este un SoC (System-on-Chip) care integrează un procesor dual-core Tensilica Xtensa LX6 operând la până la 240 MHz, aproximativ 520 kB SRAM și 4 MB memorie flash [1].

Perifericele relevante pentru acest proiect includ:

- Două ADC-uri SAR pe 12 biți cu 18 canale de măsurare
- Conectivitate Wi-Fi 802.11 b/g/n și Bluetooth 4.2
- Multiple interfețe: SPI, I2C, UART, PWM
- Alimentare nominală: 3.3 V

Am ales ESP32 ca nucleu al sistemului deoarece realizează eficient achiziția ADC, calculul FFT, aplicarea filtrelor digitale și transmisia datelor prin WebSocket (Fig. 1).

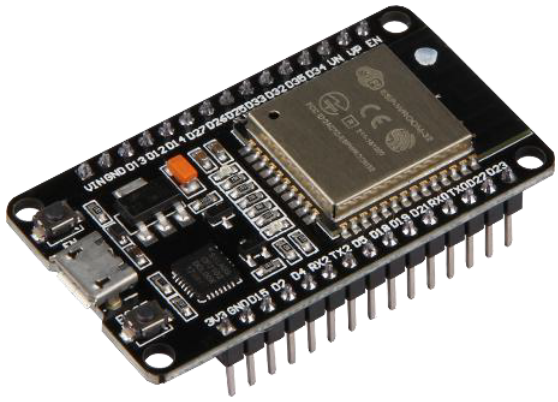

Fig. 1. Placă de dezvoltare ESP32.

B. Microfon Electret

Microfonul electret (ECM, Electret Condenser Microphone) este un traductor electroacustic bazat pe principiul condensatorului cu sarcină permanentă. Am ales acest tip de microfon pentru costul redus și sensibilitatea ridicată. Structura sa constă dintr-o diafragmă mobilă și o placă fixă (backplate), formând un condensator cu capacitatea [8]:

$$C = \varepsilon_0 \varepsilon_r \frac{A}{d} \tag{6}$$

unde A este aria plăcilor, d distanța dintre ele, ε_0 permitivitatea vidului și ε_r permitivitatea relativă a materialului dielectric.

Materialul electret (de obicei PTFE/Teflon) menține o sarcină electrică permanentă Q. Când undele sonore deplasează diafragma, distanța d variază, modificând capacitatea și generând o tensiune proporțională cu presiunea acustică [9]:

$$V = \frac{Q}{C} = \frac{Q \cdot d}{\varepsilon_0 \varepsilon_r A} \tag{7}$$

Microfonul electret include intern un tranzistor JFET (Junction Field-Effect Transistor) ca buffer de impedanță. JFET-ul este necesar deoarece impedanța de ieșire a capsulei este extrem de ridicată (ordinul MΩ), iar JFET-ul oferă o impedanță de intrare suficient de mare pentru a nu descărca sarcina capacitorului [9].

C. Preamplificator pentru Microfon Electret

Semnalul de ieșire al microfonului electret are o amplitudine de ordinul milivolților, insuficientă pentru excursia completă a ADC-ului. Am proiectat un preamplificator în două etaje

care ridică nivelul semnalului la amplitudinea necesară pentru achiziție, bazându-ne pe designul propus de Analog Devices în laboratorul dedicat microfonului electret [10].

Circuitul preamplificator, prezentat în Fig. 2, include:

- Două tranzistoare NPN (2N3904) în configurație de amplificare cu emitor comun
- Prima etapă asigură polarizarea microfonului și amplificarea inițială
- A doua etapă oferă câștig suplimentar și adaptare de impedanță
- Condensatori de cuplaj pentru separarea componentelor DC între etape
- Rețea de polarizare pentru stabilirea punctului static de funcționare

Conform specificațiilor Analog Devices, rezistența de polarizare a microfonului (drain resistor) trebuie să aibă o valoare între $680\,\Omega$ și $2.2\,\text{k}\Omega$ pentru a asigura funcționarea corectă a JFET-ului intern [10]. Am utilizat o rezistență de $2.2\,\text{k}\Omega$ care plasează tensiunea de drain la aproximativ $2.5\,\text{V}$ pentru o alimentare de $5\,\text{V}$.

Câștigul total al preamplificatorului este determinat de configurația celor două etaje și permite amplificarea semnalului audio până la niveluri compatibile cu intervalul de intrare al ADC-ului ESP32.

D. Divizor de Tensiune pentru Offset ADC

ADC-ul ESP32 măsoară doar tensiuni pozitive în intervalul de la 0 la $3.3\,\text{V}$. Semnalul audio, fiind alternativ, oscilează în jurul valorii zero. Pentru a permite achiziția corectă, am aplicat un offset DC de $1.65\,\text{V}$ (jumătate din V_{DD}).

Divizorul de tensiune este format din două rezistențe de $10\,\text{k}\Omega$ conectate între V_{DD} și masă, generând la joncțiune tensiunea de offset. Un condensator de cuplaj de $1\,\mu\text{F}$ izolează componenta DC a preamplificatorului, permițând doar componentei AC să se suprapună peste offset.

E. Circuit Complet

Am utilizat următoarele componente pentru realizarea circuitului:

- ESP32 DevKit
- Breadboard pentru prototipare
- $3\times$ rezistențe de $1\,\text{k}\Omega$
- $4\times$ rezistențe de $10\,\text{k}\Omega$
- $2\times$ tranzistoare NPN 2N3904
- Condensatori: $10\,\mu\text{F}$, $47\,\mu\text{F}$, $220\,\mu\text{F}$, $1\,\mu\text{F}$
- Microfon electret

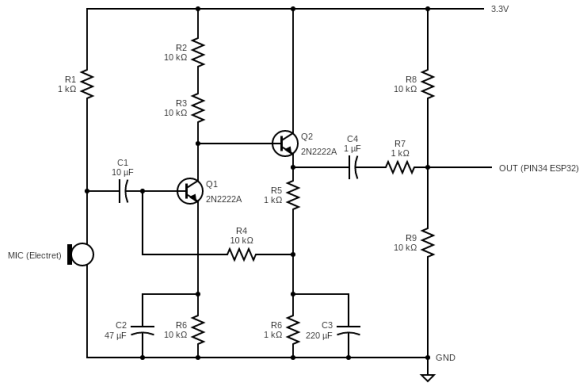

Fig. 2. Schema completă a circuitului.

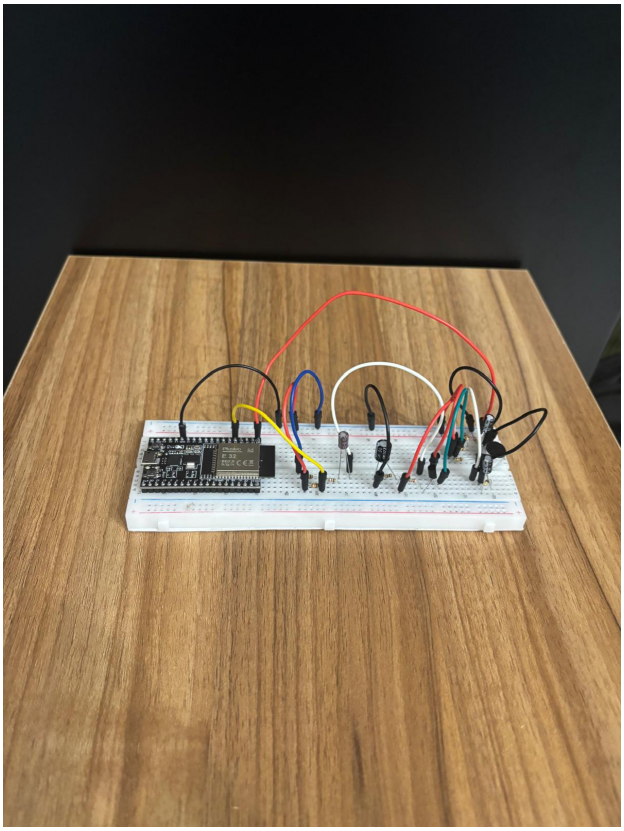

Fig. 3. Montajul real pe breadboard.

IV. Software

A. Firmware ESP32

Am dezvoltat firmware-ul ESP32 în Arduino IDE, implementând următoarele funcții:

- Eșantionarea semnalului audio la $16\,\text{kHz}$
- Calculul FFT pe 128 de eșantioane
- Extragerea a 9 benzi de frecvență (de la 0 la $8\,\text{kHz}$)
- Aplicarea filtrelor digitale configurabile
- Calculul metricilor de calitate (SNR, MSE, PSNR)
- Transmisia datelor prin WebSocket

Am utilizat bibliotecile WiFi (built-in) și WebSocketsClient (Links2004) pentru comunicarea cu serverul backend.

B. Comunicarea prin WebSocket

Am configurat ESP32 să deschidă un canal WebSocket securizat (WSS) către serverul cloud, transmițând periodic (la fiecare $350\,\text{ms}$) un pachet JSON care conține:

- `volume`: Amplitudine RAW (de la 0 la 100%)

- `volumeFiltered`: Amplitudine filtrată (de la 0 la 100%)
- `bands[]`: Vector cu 9 benzi FFT brute
- `bandsFiltered[]`: Vector cu 9 benzi FFT filtrate
- `snrRaw`, `snrFiltered`: Raportul semnal-zgomot
- `mse`, `psnr`: Metrici de calitate
- `min`, `max`, `avg`: Statistici ADC

C. Server Backend (FastAPI)

Am implementat arhitectura backend-ului în Python cu framework-ul FastAPI, cuprinzând:

1) WebSocket Manager:

- Endpoint `/ws` pentru conexiunea ESP32
- Endpoint `/ws-dashboard` pentru clienții frontend
- Broadcast în timp real către toți clienții conectați
- Gestionarea conexiunilor și deconectărilor

2) REST API:

- Endpoint `/api/info` pentru informații despre sistem
- CORS activat pentru acces cross-origin
- Logging structurat pentru debugging

D. Frontend Dashboard (React)

Am dezvoltat dashboard-ul frontend cu React, TypeScript și Tailwind CSS, oferind:

- Vizualizare waveform și spectrogramă
- Controale de filtrare în timp real
- Afișarea indicatorilor de calitate a semnalului
- Statusul conexiunii ESP32 și backend

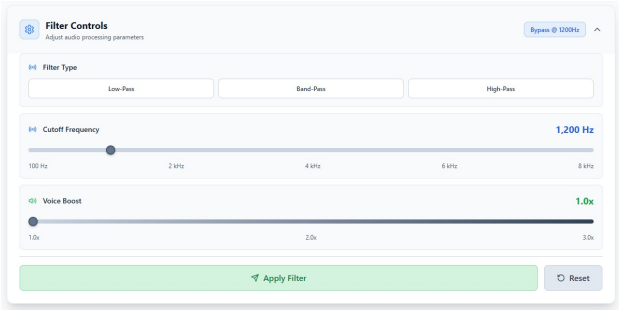

Fig. 4. Meniul de control al filtrelor.

V. Măsurarea Calității Semnalului

A. SNR (Signal-to-Noise Ratio)

Raportul semnal-zgomot cuantifică puterea semnalului util în raport cu puterea zgomotului:

$$SNR = 10 \log_{10} \left(\frac{P_{signal}}{P_{noise}} \right) \text{ [dB]} \tag{8}$$

Am implementat calculul separat al SNR pentru semnalul brut și cel filtrat, permițând evaluarea eficienței filtrării. Valori tipice pentru semnal audio de calitate sunt peste 30 dB.

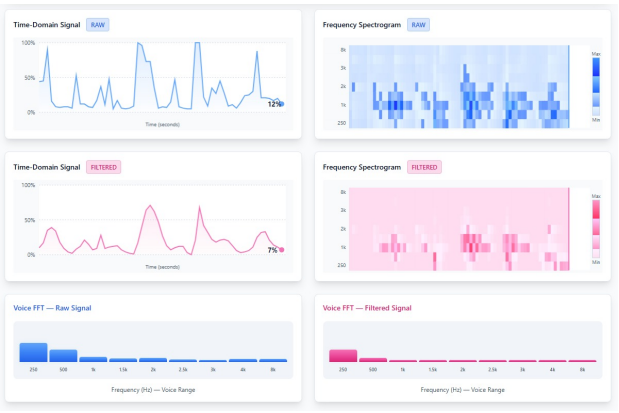

Fig. 5. Reprezentările grafice ale semnalelor (waveform și spectrogramă).

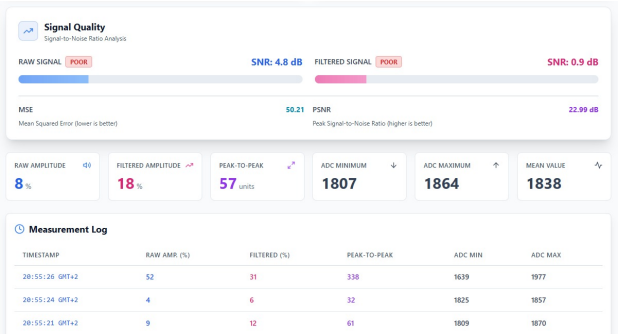

Fig. 6. Indicatorii de calitate a semnalului.

B. MSE (Mean Squared Error)

Eroarea medie pătratică măsoară diferența între benzile de frecvență brute și filtrate:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2 \tag{9}$$

unde x_i sunt valorile brute și $\hat{x}_i$ valorile filtrate.

Am observat că valori MSE mai mici indică o mai bună păstrare a caracteristicilor originale ale semnalului după filtrare.

C. PSNR (Peak Signal-to-Noise Ratio)

PSNR exprimă raportul dintre valoarea maximă posibilă a semnalului și eroarea de distorsiune:

$$PSNR = 20 \log_{10} \left(\frac{MAX}{\sqrt{MSE}} \right) \text{ [dB]} \tag{10}$$

Am constatat că valori PSNR mai mari de 30 dB indică, în general, o calitate bună a semnalului procesat.

VI. Rezultate și Performanțe

Am obținut următoarele performanțe cu sistemul nostru:

- Rată de eșantionare: 16 kHz
- Rezoluție FFT: 128 eșantioane (aproximativ 125 Hz per bin)
- Rată de actualizare: 350 ms

- Bandă de frecvență: de la 0 la $8\,\mathrm{kHz}$
- Latență comunicare: sub $100\,\mathrm{ms}$

Testele noastre au demonstrat că filtrarea îmbunătățește vizibil SNR și PSNR, confirmând corectitudinea implementării algoritmilor DSP.

VII. Probleme Întâmpinate și Limitări

Pe parcursul dezvoltării sistemului, am identificat mai multe limitări care afectează performanța și posibilitățile de extindere ale proiectului.

A. Limitări Hardware

1) Memoria RAM a ESP32: ESP32 dispune de aproximativ $520\,\mathrm{kB}$ SRAM, din care o parte semnificativă este utilizată de stiva Wi-Fi și de sistemul de operare FreeRTOS. Memoria disponibilă pentru aplicație limitează:

- Dimensiunea buffer-ului FFT (am fost nevoiți să limităm la 128 de eșantioane)
- Numărul de eșantioane stocate pentru analiză temporală
- Complexitatea algoritmilor de filtrare implementabili în timp real

2) Caracteristicile Microfonului Electret: Microfonul electret utilizat prezintă câteva limitări intrinseci:

- Raport semnal-zgomot (SNR) limitat, tipic între $58\,\mathrm{dB}$ și $62\,\mathrm{dB}$
- Răspuns în frecvență neliniar, cu atenuare la frecvențele foarte joase (sub $100\,\mathrm{Hz}$) și foarte înalte (peste $10\,\mathrm{kHz}$)
- Sensibilitate la variațiile de temperatură și umiditate
- Distorsiuni armonice la niveluri ridicate de presiune sonoră din cauza neliniarității JFET-ului intern

B. Limitări în Reprezentarea în Timp Real

Procesarea și transmiterea datelor în timp real impun compromisuri între latență și calitate:

- Rata de actualizare de $350\,\mathrm{ms}$ reprezintă un compromis între fluiditatea vizualizării și încărcarea procesorului
- Calculul FFT și al metricilor de calitate consumă cicluri de procesare semnificative
- Transmisia WebSocket adaugă latență variabilă în funcție de condițiile rețelei
- Rezoluția spectrală de $125\,\mathrm{Hz}$ per bin limitează precizia analizei frecvențiale pentru semnale cu componente spectrale apropiate

C. Limitări pentru Integrarea TinyML

Am evaluat posibilitatea integrării unor modele de machine learning (TinyML) pentru clasificare audio, însă am identificat obstacole semnificative:

- Calitatea semnalului de la microfonul electret este insuficientă pentru antrenarea robustă a modelelor ML
- Zgomotul de fond și distorsiunile introduse de lanțul analog afectează acuratețea clasificării
- Memoria limitată a ESP32 restricționează dimensiunea modelelor care pot fi implementate
- Un microfon digital MEMS (precum INMP441 sau SPH0645) ar oferi performanțe superioare pentru aplicații TinyML, cu SNR tipic peste $65\,\mathrm{dB}$ și interfață I2S directă
- Lipsa unui codec audio integrat limitează posibilitățile de preprocesare hardware a semnalului

Aceste limitări sugerează că, pentru aplicații avansate de clasificare audio sau recunoaștere vocală, ar fi necesară o reproiectare a lanțului de achiziție cu componente de calitate superioară.

VIII. Concluzii

Am demonstrat prin acest proiect realizarea unui sistem embedded complet capabil de achiziția, procesarea și transmiterea semnalelor audio în timp real. Am concluzionat că ESP32 este o platformă potrivită pentru aplicații DSP de nivel mediu, oferind un echilibru optim între puterea de calcul, conectivitatea integrată și costul redus.

Contribuțiile noastre principale includ:

- Integrarea completă hardware-software pentru procesare audio în timp real
- Implementarea analizei spectrale FFT pe un microcontroler embedded
- Dezvoltarea unui sistem de monitorizare și control bazat pe WebSocket
- Vizualizare interactivă a datelor într-un dashboard web modern

Ca dezvoltări viitoare, propunem integrarea modelelor de machine learning pentru clasificare acustică sau detecție de evenimente sonore, precum și extinderea către aplicații de monitorizare industrială sau medicală.

References

[1] Espressif Systems, "ESP32 Series Datasheet," Version 4.5, 2024.

[2] Espressif Systems, "ESP-IDF Programming Guide – Analog to Digital Converter," https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc.html.

[3] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[5] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[6] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455, IETF, December 2011. https://datatracker.ietf.org/doc/html/rfc6455

[7] WebSocket.org, "The WebSocket Protocol," https://websocket.org/guides/websocket-protocol/.

[8] G. M. Sessler and J. E. West, "Self-biased condenser microphone with high capacitance," *The Journal of the Acoustical Society of America*, vol. 34, no. 11, pp. 1787–1788, 1962.

[9] Wikipedia, "Electret microphone," https://en.wikipedia.org/wiki/Electret_microphone.

[10] Analog Devices, "Activity S1: Electret microphone preamplifier," ADALM1000 Active Learning Module, https://wiki.analog.com/university/courses/alm1k/alm-lab-s1.

[11] README – Real-Time Audio Analysis Platform, GitHub Repository, https://github.com/cristim67/audio-analysis-platform.