Sistem de recunoaștere a genului din semnalul de vorbire și modificarea vocii  
 -Proiect MATLAB-

Spin Carina-Bianca

Stoianovici Anka-Snezana

SIAPS an I

Cuprins

[Motivatia 3](#_heading=h.gjdgxs)

[Arhitectura aplicației 3](#_heading=h.30j0zll)

[Exemple 19](#_heading=h.1fob9te)

[Bibliografie 21](#_heading=h.2et92p0)

# **Motivatia**

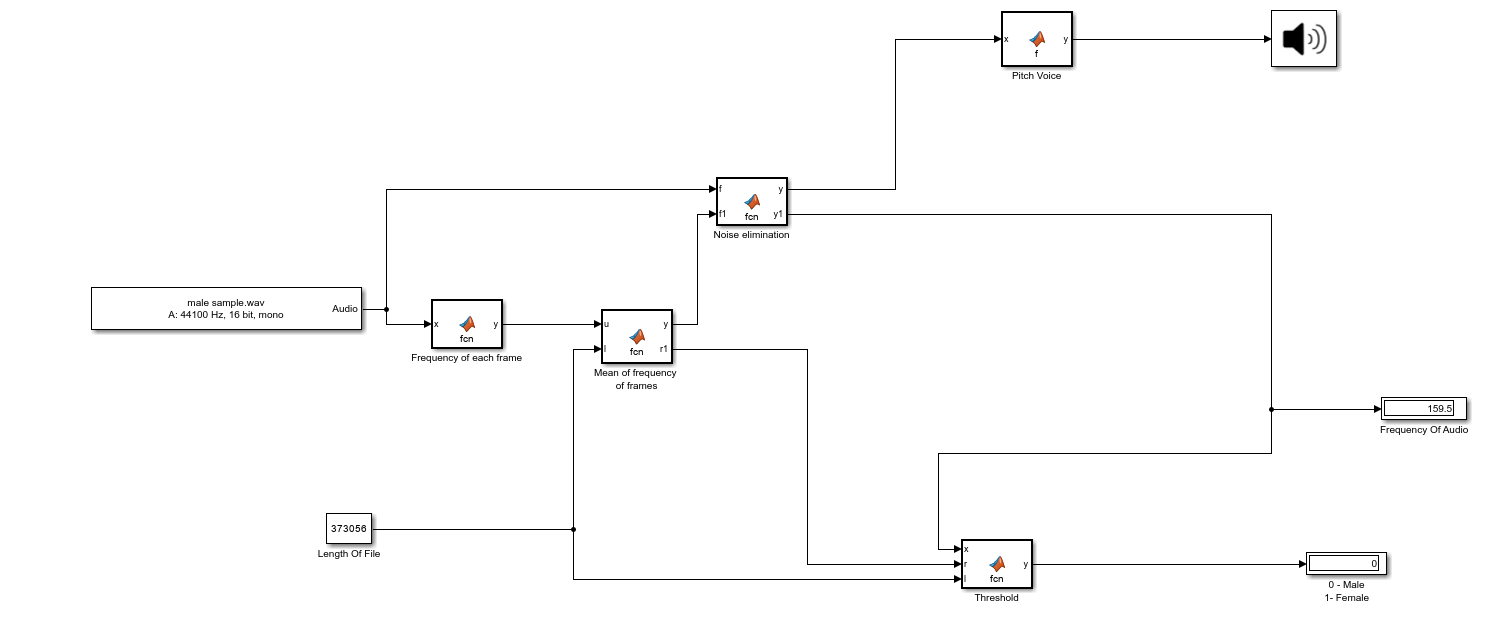
Procesarea semnalului vocal are numeroase aplicații în aproape toate domeniile tehnice. Identificarea de gen este importantă în procesarea vorbirii. Acest proiect descrie o analiză comparativă a semnalelor de vorbire pentru a produce o clasificare automată de gen. Clasificarea de gen în funcție de semnalul de vorbire este o tehnică care analizează diverse caracteristici ale unui eșantion de voce pentru a determina sexul vorbitorului.

Este prezentată o aplicație pentru codificarea vorbirii, analiză, sinteză și identificare de gen. Un sistem tipic de recunoaștere a genului este împărțit în două părți: sistemul front-end și sistemul back-end. Sarcina sistemului front-end este de a extrage informații de gen dintr-un semnal de vorbire și de a o reprezenta ca un set de cadre. Sarcina sistemului back-end (cunoscut și ca clasificator) în faza de recunoaștere este de a crea un model de gen pentru a recunoaște genul din semnalul de vorbire.

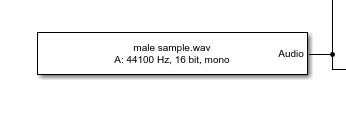
# **Arhitectura aplicației**

|  |
| --- |
| app1 |
| - UIFigure: matlab.ui.Figure  - EcouButton: matlab.ui.control.Button  - ReverseButton: matlab.ui.control.Button  - StopButton\_2: matlab.ui.control.Button  - VocepitigaiataButton: matlab.ui.control.Button  - ModificarevocefrecventaLabel: matlab.ui.control.Label  - freq: matlab.ui.control.NumericEditField  - FrequencyEditFieldLabel: matlab.ui.control.Label  - gender: matlab.ui.control.EditField  - GenderLabel: matlab.ui.control.Label  - file\_name: matlab.ui.control.EditField  - FileNameEditFieldLabel: matlab.ui.control.Label  - StopButton: matlab.ui.control.Button  - ListenAudioButton: matlab.ui.control.Button  - GenerateGraphButton: matlab.ui.control.Button  - SourceButton: matlab.ui.control.Button  - file\_msg: matlab.ui.control.TextArea  - CalculateButton: matlab.ui.control.Button  - UIAxes\_2: matlab.ui.control.UIAxes  - UIAxes: matlab.ui.control.UIAxes  - file\_: string  - flag: int  - path: string  - aud\_file: double[]  - aud\_fs: int |
| + StopButtonPushed(event: any): void  + ListenAudioButtonPushed(event: any): void  + SourceButtonPushed(event: any): void  + GenerateGraphButtonPushed(event: any): void  + CalculateButtonPushed(event: any): void  + StopButton\_2Pushed(event: any): void  + FunnyButtonPushed(event: any): void  + ReverseButtonPushed(event: any): void  + EchoButtonPushed (event: any): void  + NoiseremovalButtonPushed( event: any) : void  + VoiceRecordingButtonPushed(event: any): void  + AnalyzemusicButtonPushed (event: any): void  + createComponents(): void |

Simulink

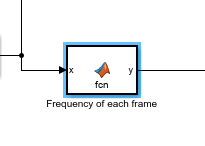
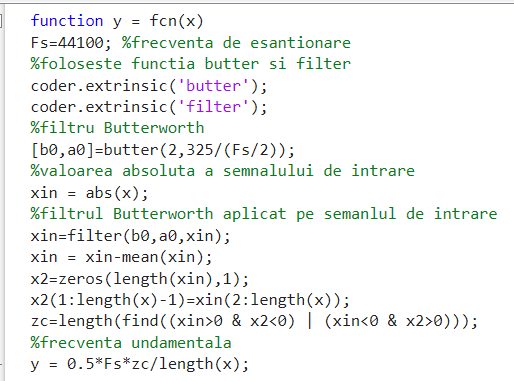


Blocul From Multimedia File



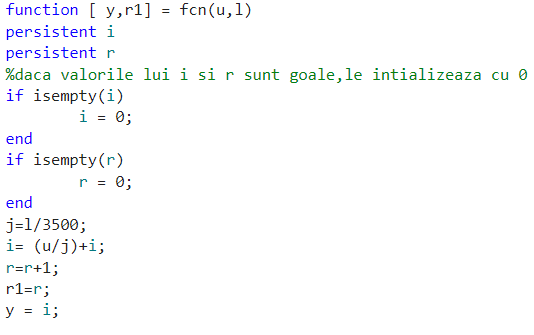
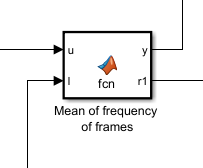
Utilizatorul selectează fișierul audio pe care dorește să îl prelucreze.

Blocul Function Frequency of each frame

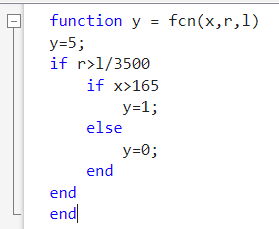
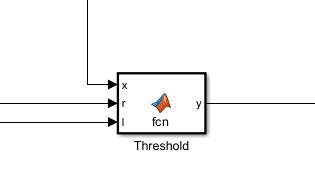
Funcția primește un semnal audio și calculează frecvența fundamentală a semnalului. Frecvența de eșantionare este setată la 44100 de eșantioane pe secundă. Comenzile coder.extrinsic('butter'); coder.extrinsic('filter'); spun că funcțiile butter și filter sunt funcții externe și sunt utilizate pentru a aplica un filtru Butterworth. Apoi se proiectează filtrul Butterworth de ordinul 2 cu o frecvență de tăiere de 325Hz. Se calculează valoarea absolută a semnalului de intrare și se aplică filtrul Butterworth pe acesta. Se creează semnalul x2 care este decalat cu un eșantion față de semnalul xin. Se calculează numărul de zero crossing-uri în semnalul xin prin numărarea momentelor în care semnalul trece de la valori pozitive la negative sau invers. La final se calculează frecvența fundamentală.

Blocul Function Mean ofrequency of frames



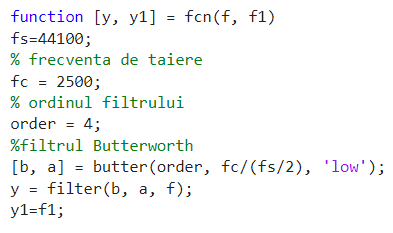
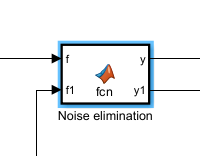
Funcția efectuează o operație de actualizare a unui contor r și calculează termenul y bazat pe parametrii de intrare. Valoarea rezultată este dependentă de intrările anterioare ale funcției și de starea persistentă a variabilelor.

Blocul Function Threshold



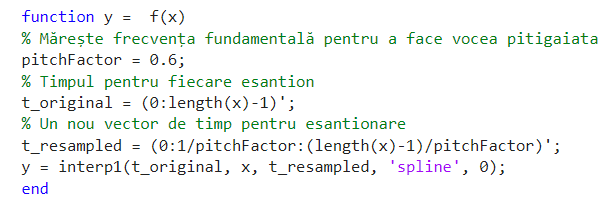
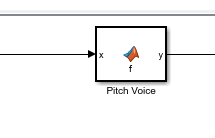
Prin această funcție se verifică dacă sunetul audio este produs de o femeie sau un bărbat. Se verifică dacă frecvența este mai mare de 165, iar dacă este variabila y are valoarea 1, ce reprezintă că este femeie. Dacă variabila y este 0, semnalul audio este produs de un bărbat.

Blocul Function Noise elimination

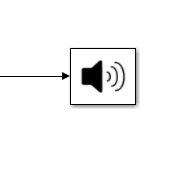
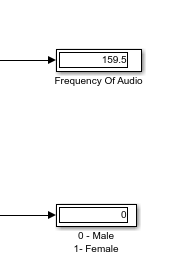


Frecvența de eșantionare este setată la 44100 de eșantioane pe secundă, iar frecvența de tăiere a filtrului Butterworth la 2500Hz. Ordinul filtrului este setat la 4. Se proiectează un filtru Butterworth trece-jos, iar apoi se aplică filtrul. Frecvența este transmisă mai departe prin variabila y1.

Blocul Pitch Voice

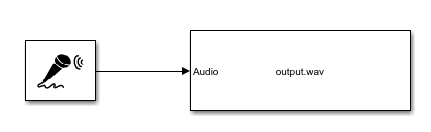


Funcția pentru realizarea vocii pițigăiate mărește frecvența fundamentală. Se creează un vector de timp pentru eșantionare și apoi un nou vector de timp eșantionare în care eșantioanele sunt separate cu factorul de voce pitigaiata.

Sunetul audio modificat se poate asculta în timpul rulării simulării prin blocul Audio Device Writer. Frecvența audio se poate urmări în blocul display Frequency of Audio, iar rezultatul final, dacă este o voce feminină sau o voce bărbătească se poate observa în blocul display 0-Male 1-Female.

Blocul Audio Device Reader și blocul To Multimedia File



Dacă se dorește înregistrarea sunetului se realizează această schemă, iar apoi se poate încărca fișierul output.wav generat în schema prezentată anterior pentru prelucrarea ei.

Codul sursa

|  |
| --- |
| function SourceButtonPushed(app, event)  [file, p] = uigetfile({'\*.mp3;\*.wav', 'Audio Files (\*.mp3, \*.wav)'}, 'Select an audio file');  if ischar(file) && (endsWith(file, '.mp3') || endsWith(file, '.wav'))  app.flag = 1;  app.file\_msg.Value = 'File Uploaded!!';  app.path = p;  s = fullfile(p, file);  app.file\_name.Value = file;  app.file\_ = s;  app.flag = 1;    if endsWith(file, '.mp3')  [app.aud\_file, app.aud\_fs] = audioread(app.file\_, 'native');  elseif endsWith(file, '.wav')  [app.aud\_file, app.aud\_fs] = audioread(app.file\_);  end  else  app.file\_msg.Value = 'Incorrect File Format. Please select a .mp3 or .wav file.';  end  end |

La apăsarea butonului, utilizatorului i se deschide o fereastră de dialog pentru a selecta un fișier audio în formatele .mp3 sau .wav. Dacă utilizatorul alege un fișier valid, informațiile despre fișier (nume, cale, date audio și frecvența de eșantionare) sunt stocate în proprietățile aplicației (app) pentru utilizare ulterioară. În caz contrar, se afișează un mesaj de eroare indicând utilizatorului să selecteze un fișier în formatul corect.

|  |
| --- |
| function CalculateButtonPushed(app, event)  if app.flag == 1  try  [y, Fs] = audioread(app.file\_);  frame = 1024;  app.aud\_file = y;  app.aud\_fs = Fs;    [b0, a0] = butter(4, [50/(Fs/2), 500/(Fs/2)]);    F0 = zeros(1, floor(length(y)/frame));  for i = 1:length(y)/frame  x = y(1+(i-1)\*frame:i\*frame);  xin = abs(x);  xin = filter(b0, a0, xin);  xin = xin - mean(xin);    R = xcorr(xin);  [~, loc] = findpeaks(R(frame:end), 'NPeaks', 1, 'MinPeakHeight', 0.1);    if ~isempty(loc) && loc(1) <= length(R)  F0(i) = Fs / max(round(loc(1)), 1);  else  F0(i) = 0;  end  disp(['Frame ' num2str(i) ', F0: ' num2str(F0(i))]);  end  app.freq.Value = mean(F0);  if app.freq.Value > 165  app.gender.Value = 'Female';  y = y / max(abs(y));  spectrum = abs(fft(y));  vocalRange = [80, 300];  vocalIndices = find(Fs/ 2 \* linspace(0, 1, length(spectrum)/2 + 1) >= vocalRange(1) & Fs / 2 \* linspace(0, 1, length(spectrum)/2 + 1) <= vocalRange(2));  vocalAmplitudes = spectrum(vocalIndices);  app.ampli.Value = mean(vocalAmplitudes);  app.indic.Value = mean(vocalIndices);  app.file\_msg.Value = 'Done';  elseif app.freq.Value > 85 && app.freq.Value < 165  app.gender.Value = 'Male';  app.file\_msg.Value = 'Done';  end    catch  app.file\_msg.Value = 'Error processing file. Please check the format.';  end  else  app.file\_msg.Value = 'Incorrect Format!!';  end  end |

Se verifică dacă flagul (app.flag) este setat la 1, ceea ce indică că a fost încărcat un fișier audio valid. Se citește fișierul audio și se extrag datele și frecvența de eșantionare ([y, Fs] = audioread(app.file\_)). Se aplică un filtru Butterworth pentru a evidenția componentele de frecvență relevantă ([b0, a0] = butter(4, [50/(Fs/2), 500/(Fs/2)])). Se calculează frecvența fundamentală folosind autocorelarea pe ferestre de date (F0). Se afișează frecvența medie estimată pe interfața grafică (app.freq.Value = mean(F0)). Pe baza frecvenței estimate, se determină genul vorbitorului (bărbat sau femeie), iar pentru vorbitoarele de sex feminin se efectuează procesări suplimentare, inclusiv calculul amplitudinii într-un interval vocal specific. Rezultatele sunt afișate în cadrul unei interfețe grafice de utilizator. Gestionarea erorilor este inclusă pentru situațiile în care apare o problemă în procesarea fișierului audio.

|  |
| --- |
| function FunnyButtonPushed(app, event)  pitchFactor = 1.5;  outputAudio = interp1(1:length(app.aud\_file), app.aud\_file, linspace(1, length(app.aud\_file), round(length(app.aud\_file) / pitchFactor)), 'spline');  plot(app.UIAxes\_2, outputAudio);  title(app.UIAxes\_2, 'Funny Voice');  xlabel(app.UIAxes\_2, 'Time');  ylabel(app.UIAxes\_2, 'Amplitude');  grid(app.UIAxes\_2,"on");  sound(outputAudio, app.aud\_fs);  end |

Funcția FunnyButtonPushed realizează o modificare a tonalității vocii prin ajustarea frecvenței fundamentale a semnalului audio încărcat în aplicație. Această ajustare creează un efect amuzant prin mărirea frecvenței fundamentale cu un factor specific (în acest caz, 1.5). Procedura utilizează funcția "interp1" pentru a obține un nou semnal audio cu tonalitatea modificată, pe baza datelor semnalului original. Semnalul rezultat este apoi afișat pe o axă specifică în interfața grafică, iar utilizatorul poate asculta efectul amuzant prin redarea sunetului modificat.

|  |
| --- |
| function ReverseButtonPushed(app, event)  outputAudio = flipud(app.aud\_file);    plot(app.UIAxes\_2, outputAudio);  title(app.UIAxes\_2, 'Reverse');  xlabel(app.UIAxes\_2, 'Time');  ylabel(app.UIAxes\_2, 'Amplitude')  grid(app.UIAxes\_2,"on");  sound(outputAudio, app.aud\_fs);  end |

Funcția ReverseButtonPushed implementează funcționalitatea de redare a sunetului într-un mod inversat. În detaliu, semnalul audio încărcat în aplicație este inversat în ordine, astfel încât începutul devine sfârșitul și viceversa. Semnalul rezultat este apoi afișat pe o altă axă din interfața grafică, iar utilizatorul poate asculta efectul sunetului inversat prin intermediul redării audio. Afișarea grafică include un titlu corespunzător, etichete pentru axele de timp și amplitudine, precum și un sistem de grilă pentru a evidenția aspectele temporale ale semnalului.

|  |
| --- |
| function EchoButtonPushed(app, event)  echo\_IR = zeros(1, round(app.aud\_fs \* 0.5));  echo\_IR(1) = 1;  attenuationFactor = 0.5;  echoSignal = conv(app.aud\_file(:), echo\_IR, 'same') \* attenuationFactor;    outputAudio = app.aud\_file(:) + echoSignal;  outputAudio = outputAudio / max(abs(outputAudio));    plot(app.UIAxes\_2, outputAudio);  title(app.UIAxes\_2, 'Echo');  xlabel(app.UIAxes\_2, 'Time');  ylabel(app.UIAxes\_2, 'Amplitude');  grid(app.UIAxes\_2,"on");  sound(outputAudio, app.aud\_fs);  end |

Se inițializează un vector cu lungimea corespunzătoare unei durate de 0.5 secunde la frecvența de eșantionare a sunetului. Se setează primul element al vectorului la 1 pentru a reprezenta impulsul la început. Se definește un factor de atenuare, în acest caz, 0.5. Se utilizează operația de convoluție pentru a aplica impulsul de răspuns la impuls asupra semnalului audio original. Rezultatul este un semnal cu efect de ecou. Se adaugă semnalul cu ecou la semnalul original pentru a obține sunetul final cu efect de ecou. Se normalizează amplitudinea semnalului rezultat pentru a asigura că nu apare suprapunere (clipping) atunci când sunetul este redat.

|  |
| --- |
| function NoiseremovalButtonPushed(app, event)  j=0;  fc = 2500;  order = 4;  [b, a] = butter(order, fc/(app.aud\_fs/2), 'low');  outputAudio = filter(b, a, app.aud\_file);  plot(app.UIAxes\_2, outputAudio);  title(app.UIAxes\_2, 'Sound without noise');  xlabel(app.UIAxes\_2, 'Time');  ylabel(app.UIAxes\_2, 'Amplitude');  grid(app.UIAxes\_2,"on");  sound(outputAudio, app.aud\_fs);  name = 'filtarezgomot';  form = '.wav';  j=j+1;  outputPath= sprintf('%s\_%d%s', name, j, form);  audiowrite(outputPath,outputAudio,44100);  file = uigetfile(outputPath);  if ischar(file) && (endsWith(file, '.mp3') || endsWith(file, '.wav'))  app.flag = 1;  app.file\_msg.Value = 'File Uploaded!!';  s = fullfile(file);  app.file\_name.Value = file;  app.file\_ = s;  app.flag = 1;    % Read the audio file based on the file extension  if endsWith(file, '.mp3')  [app.aud\_file, app.aud\_fs] = audioread(app.file\_, 'native');  elseif endsWith(file, '.wav')  [app.aud\_file, app.aud\_fs] = audioread(app.file\_);  end  else  app.file\_msg.Value = 'Incorrect File Format. Please select a .mp3 or .wav file.';  end  end |

Functia implementează un proces de eliminare a zgomotului dintr-un fișier audio, aplicând un filtru trece-jos Butterworth. Semnalul audio rezultat este apoi afișat grafic în interfața grafică, redat pentru auditivizare și salvat într-un fișier WAV cu un nume specific și un index pentru a evita suprascrierea. După salvare, utilizatorului i se oferă opțiunea de a selecta noul fișier audio și de a valida formatul acestuia, permițând actualizarea mesajelor și informațiilor în interfața grafică. Fișierele audio pot fi în format .mp3 sau .wav, iar funcția gestionează citirea și prelucrarea acestora în consecință.

|  |
| --- |
| function VoiceRecordingButtonPushed(app, event)  i=0;  %recorder object  recorder = audiorecorder(44100,24, 2);  %record 5 sec  disp('Record your voice 5 sec');  drawnow();  pause(1);  recordblocking(recorder, 5);  play(recorder)  data = getaudiodata(recorder);  name = 'record';  form = '.wav';  i=i+1;  outputPath= sprintf('%s\_%d%s', name, i, form);  audiowrite(outputPath,data,44100);  f=voiceFFT(data);  function [x] = voiceFFT(data)  F=fft(data(:,1));  n=max(real(F));  x=find(real(F)==n,1)  end  file = uigetfile(outputPath);  if ischar(file) && (endsWith(file, '.mp3') || endsWith(file, '.wav'))  app.flag = 1;  app.file\_msg.Value = 'File Uploaded!!';  s = fullfile(file);  app.file\_name.Value = file;  app.file\_ = s;    % Read the audio file based on the file extension  if endsWith(file, '.mp3')  [app.aud\_file, app.aud\_fs] = audioread(app.file\_, 'native');  elseif endsWith(file, '.wav')  [app.aud\_file, app.aud\_fs] = audioread(app.file\_);  end  else  % Display an error message if an incorrect file is selected  app.file\_msg.Value = 'Incorrect File Format. Please select a .mp3 or .wav file.';  end  end |

Se realizează înregistrarea vocii utilizatorului pentru o perioadă de 5 secunde, utilizând un obiect recorder. Datele audio sunt salvate într-un fișier cu numele 'record\_X.wav', unde X reprezintă un număr de înregistrare incremental. Funcția utilizează o subfuncție voiceFFT pentru a efectua transformata Fourier rapidă (FFT) a datelor audio și pentru a identifica frecvența cu cea mai mare componentă reală.

În continuare, utilizatorul este invitat să selecteze un fișier audio salvat, iar funcția verifică dacă fișierul selectat are extensia .mp3 sau .wav. În caz afirmativ, informațiile despre fișier sunt actualizate în cadrul unei interfețe grafice de utilizator (GUI), iar un indicator de flag (app.flag) este setat la 1. Dacă fișierul selectat nu corespunde formatului așteptat, se afișează un mesaj de eroare în GUI

|  |
| --- |
| function AnalyzemusicButtonPushed(app, event)  sound(app.aud\_file,app.aud\_fs);  song = app.aud\_file;  Fs = app.aud\_fs\*4;  t1 = 1;  t2 = 1e6;  if t2 > length(song)  t2 = length(song);  end  y = song(t1:t2);  [~,n] = size(y);  t = linspace(t1,t2,n);  m = 20;  Fsm = round(Fs/m);  p = floor(n/m);  y\_avg = zeros(1,p);  for i = 1:p  y\_avg(i) = mean(y(m\*(i-1)+1:m\*i));  end  close all  y\_thresh = zeros(1,p);  i = 1;  while (i <= p)  thresh = 5\*median(abs(y\_avg(max(1,i-5000):i)));  if (abs(y\_avg(i)) > thresh)  for j = 0:500  if (i + j <= p)  y\_thresh(i) = y\_avg(i);  i = i + 1;  end  end  i = i + 1400;  end  i = i + 1;  end  i = 1;  i\_note = 0;  while i < p  j = 1;  end\_note = 0;  while (((y\_thresh(i) ~= 0) || (end\_note > 0)) && (i < p))  note(j) = y\_thresh(i);  i = i + 1;  j = j + 1;  if (y\_thresh(i) ~= 0)  end\_note = 20;  else  end\_note = end\_note - 1;  end  if (end\_note == 0)  if (j > 25)  note\_padded = [note zeros(1,j)];  Note = fft(note\_padded);  Ns = length(note);  f = linspace(0,(1+Ns/2),Ns);  [~,index] = max(abs(Note(1:length(f))));  if (f(index) > 20)  i\_note = i\_note + 1;  fundamentals(i\_note) = f(index)\*2;  end  i = i + 50;  end  clear note;  break  end    end  i = i + 1;  end  amp = 1;  fs = 20500;  duration = .5;  recreate\_song = zeros(1,duration\*fs\*length(fundamentals));  for i = 1:length(fundamentals)  [letter(i,1),freq(i)]= FreqToNote(fundamentals(i));  values = 0:1/fs:duration;  a = amp\*sin(2\*pi\*freq(i)\*values\*2);  recreate\_song((i-1)\*fs\*duration+1:i\*fs\*duration+1) = a;  end  sound(recreate\_song,fs);  function [note,frequency] = FreqToNote(f)  index = round(17.31232\*log(f) - 47.37620);  frequencies = [16.35 17.32 18.35 19.45 20.6 21.83 23.12 24.5 25.96 27.5 29.14 30.87 32.7 34.65 36.71 38.89 41.2 43.65 46.25 49 51.91 55 58.27 61.74 65.41 69.3 73.42 77.78 82.41 87.31 92.5 98 103.83 110 116.54 123.47 130.81 138.59 146.83 155.56 164.81 174.61 185 196 207.65 220 233.08 246.94 261.63 277.18 293.66 311.13 329.63 349.23 369.99 392 415.3 440 466.16 493.88 523.25 554.37 587.33 622.25 659.25 698.46 739.99 783.99 830.61 880 932.33 987.77 1046.5 1108.73 1174.66 1244.51 1318.51 1396.91 1479.98 1567.98 1661.22 1760 1864.66 1975.53 2093 2217.46 2349.32 2489.02 2637.02 2793.83 2959.96 3135.96 3322.44 3520 3729.31 3951.07 4186.01 4434.92 4698.63 4978.03 5274.04 5587.65 5919.91 6271.93 6644.88 7040 7458.62 7902.13];  notes = ["C0" "Db0" "D0" "Eb0" "E0" "F0" "Gb0" "G0" "Ab0" "A0" "Bb0" "B0" "C1" "Db1" "D1" "Eb1" "E1" "F1" "Gb1" "G1" "Ab1" "A1" "Bb1" "B1" "C2" "Db2" "D2" "Eb2" "E2" "F2" "Gb2" "G2" "Ab2" "A2" "Bb2" "B2" "C3" "Db3" "D3" "Eb3" "E3" "F3" "Gb3" "G3" "Ab3" "A3" "Bb3" "B3" "C4" "Db4" "D4" "Eb4" "E4" "F4" "Gb4" "G4" "Ab4" "A4" "Bb4" "B4" "C5" "Db5" "D5" "Eb5" "E5" "F5" "Gb5" "G5" "Ab5" "A5" "Bb5" "B5" "C6" "Db6" "D6" "Eb6" "E6" "F6" "Gb6" "G6" "Ab6" "A6" "Bb6" "B6" "C7" "Db7" "D7" "Eb7" "E7" "F7" "Gb7" "G7" "Ab7" "A7" "Bb7" "B7" "C8" "Db8" "D8" "Eb8" "E8" "F8" "Gb8" "G8" "Ab8" "A8" "Bb8" "B8"];  note = notes(index);  frequency = frequencies(index);  end  end  end |

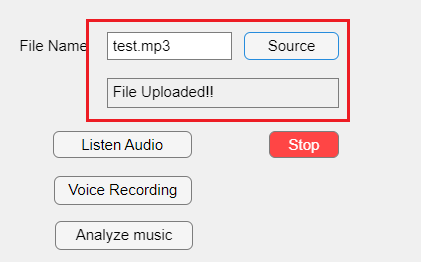
Permite utilizatorului să analizeze și să redea o porțiune specifică a unei înregistrări audio. Înregistrarea audio este prezentată și mărită în frecvență, iar apoi este subesantionată și analizată pentru a identifica notele muzicale. Fiecare notă este redată ulterior și afișată în interfața grafică de utilizator, evidențiind astfel componentele muzicale ale înregistrării. Funcția include și o subfuncție FreqToNote care asociază frecvențe cu note muzicale corespunzătoare.

# Functionalitate

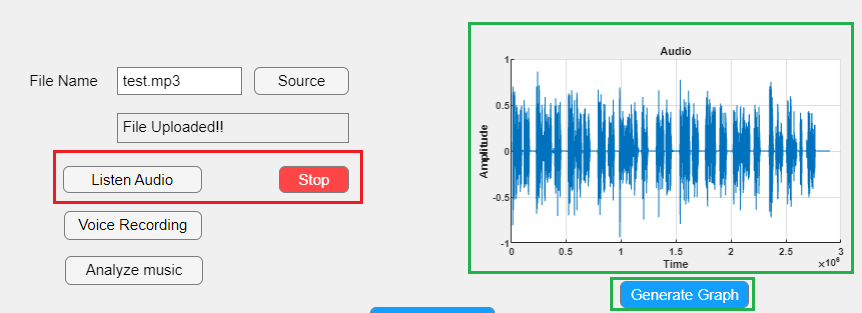
Utilizatorul încarcă un fișier audio utilizând butonul SourceButton.



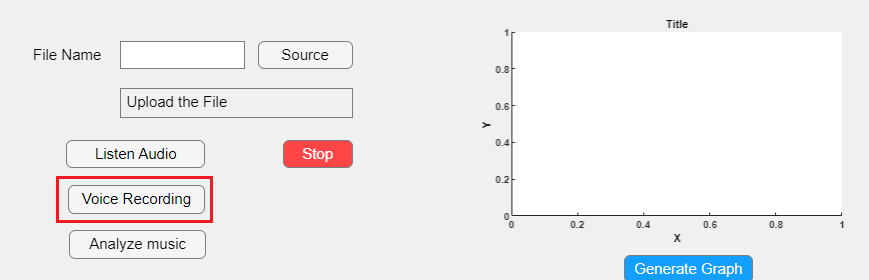
Aplicația afișează numele fișierului și încarcă datele audio.



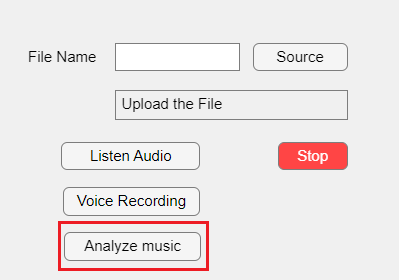
Utilizatorul poate genera un grafic al semnalului audio, reda sunetul sau să-l oprească folosind butoanele corespunzătoare.



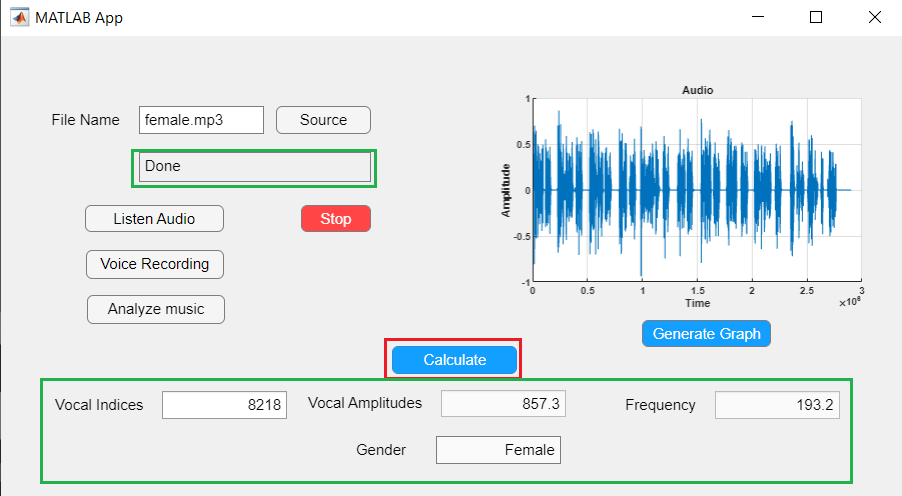
Utilizatorul poate să facă o înregistrare vocală pe care ulterior o poate analiza și realiza diferite calcule și modificări.



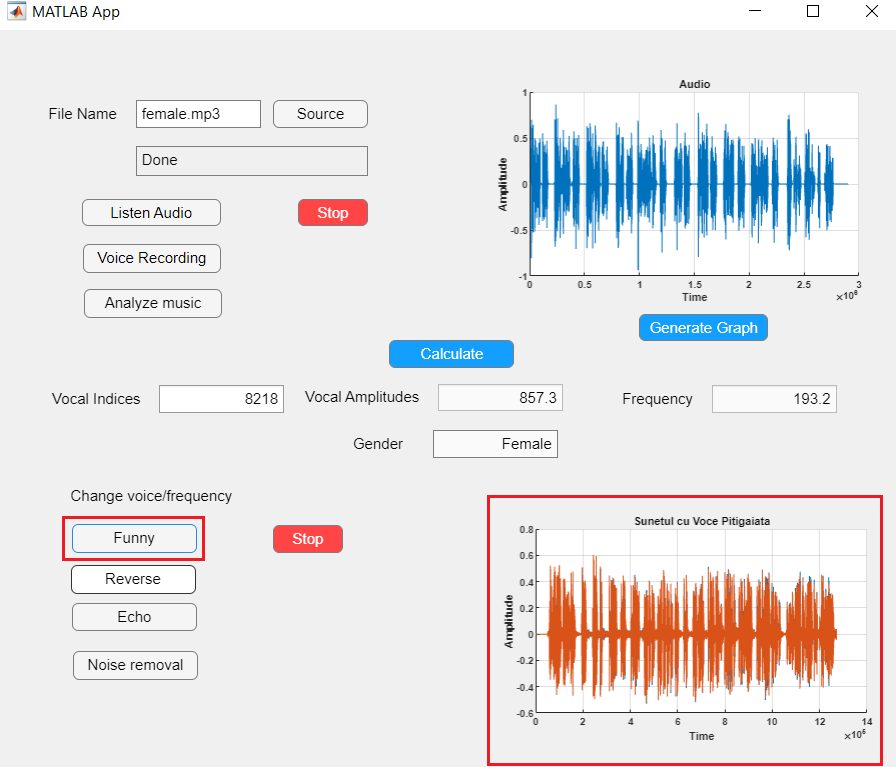
Aplicația oferă posibilitatea de a analiza sunetele muzicale dintr-un instrmunetal, iar apoi acesta reconstruiește partea din audio care a fost analizată.



Aplicația poate efectua analiza vocală, determinând frecvența fundamentală, indicii vocali, amplitudinea vocală și estimând genul vocal.

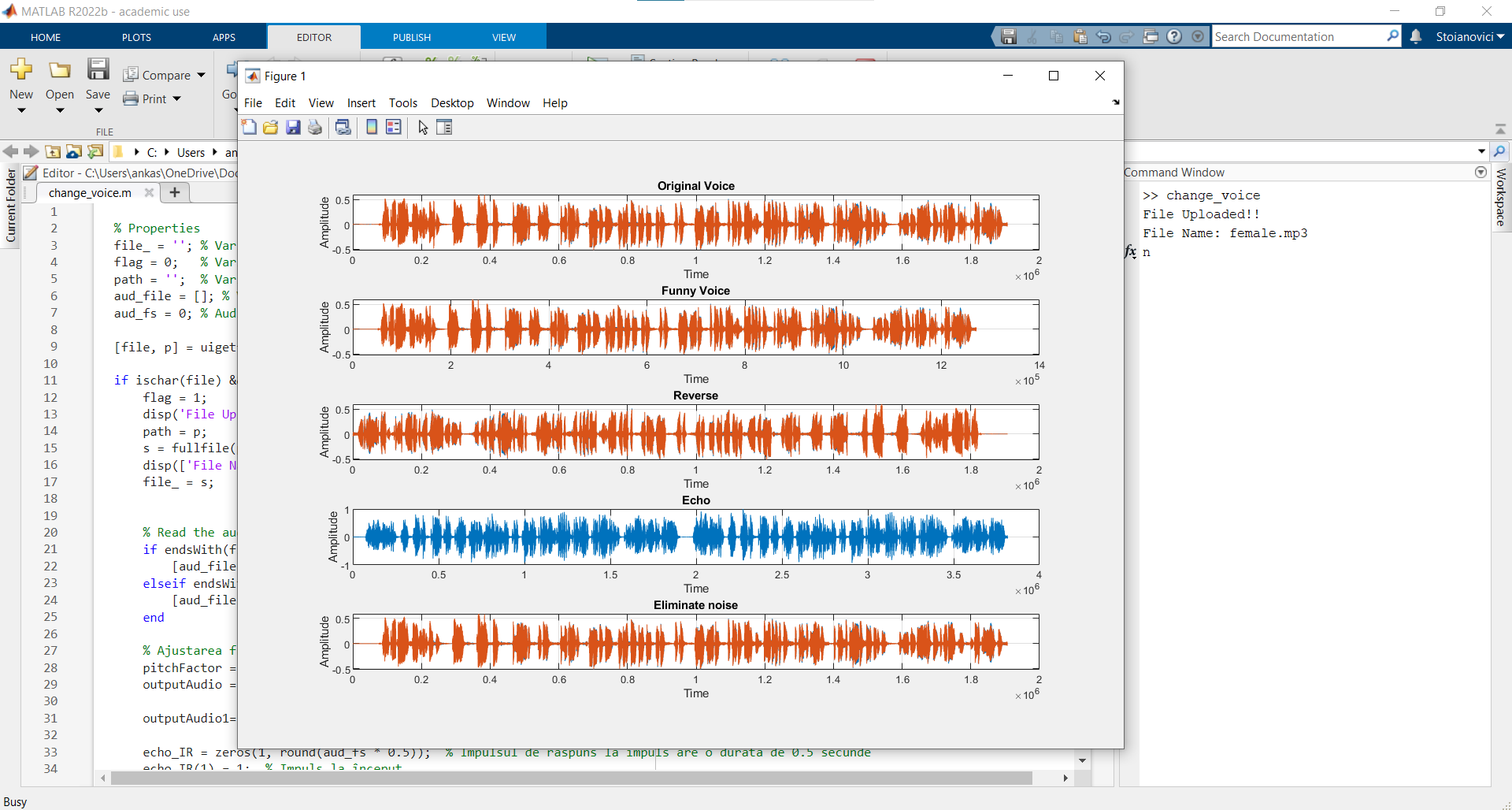


Alte funcționalități, cum ar fi modificarea vocii sau adăugarea de efecte, sunt disponibile prin butoane suplimentare, unde utilizatorul selectează butonul dorit, este redat sunetul, modificat și afișat graficul respectiv.



# 

# **Exemple**



Utilizatorul este invitat să selecteze un fișier audio cu extensia .mp3 sau .wav. Dacă un fișier valid este selectat, variabila flag devine 1, indicând că fișierul a fost încărcat corect. Datele audio și frecvența de eșantionare sunt citite în variabilele aud\_file și aud\_fs. Frecvența fundamentală a vocii este ajustată pentru a crea o voce mai înaltă și amuzantă.

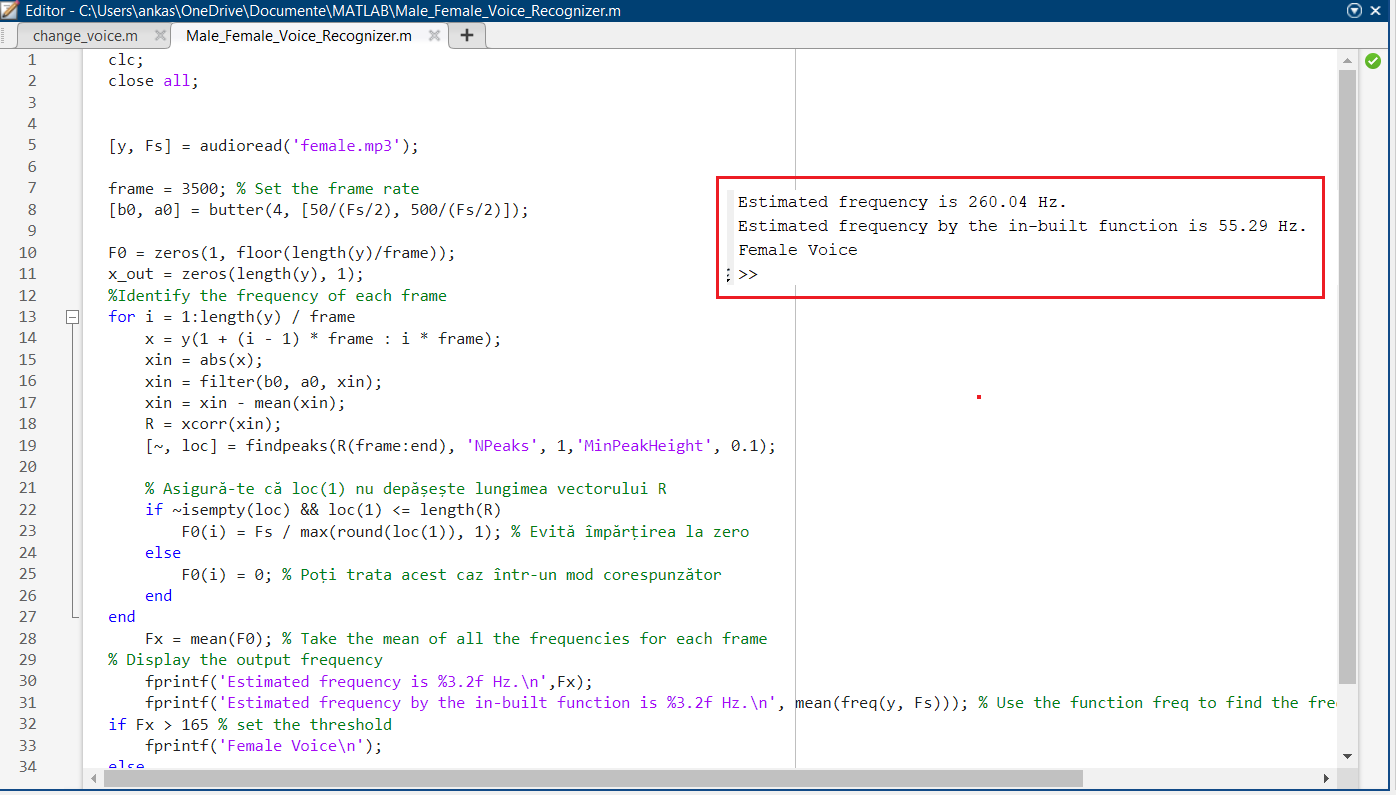
Se generează variante ale sunetului original prin:

Inversarea semnalului (outputAudio1).

Adăugarea unui efect de ecou (outputAudio2).

Eliminarea zgomotului utilizând un filtru trece-jos (outputAudio3).

Se afișează graficele pentru sunetul original și variantele manipulate, inclusiv vocea amuzantă, sunetul inversat, sunetul cu ecou și sunetul fără zgomot. Sunetul original și variantele manipulate sunt redate secvențial, cu o pauză de 5 secunde între ele. Dacă utilizatorul selectează un fișier cu o extensie incorectă, se afișează un mesaj de eroare.



Se foloseste un filtru trece-jos Butterworth pentru a elimina frecvențele nedorite din semnalul audio. Se împarte semnalul în cadre (frames) și se calculează frecvența fundamentală pentru fiecare cadru. Acest lucru se face prin analiza autocorelației semnalului. Se calculează media frecvențelor fundamentale obținute pentru fiecare cadru, iar apoi se compară cu un prag (165 Hz în acest caz). Dacă media este mai mare decât pragul, se consideră că vocea este feminină; altfel, este considerată masculină. Se folosește o funcție încorporată (freq) pentru a calcula frecvența medie a semnalului și se compară cu rezultatul obținut prin analiza autocorelației.Se afișează frecvența estimată și se indică genul vocii bazat pe pragul specificat.

# Concluzii

Aplicația dezvoltată în MATLAB are ca scop analiza semnalelor audio pentru extragerea informațiilor legate de frecvența vocală și estimarea genului vorbitorului. Prin interfața grafică dezvoltată cu App Designer, utilizatorii pot încărca fișiere audio, efectua analize asupra semnalului vocal și aplica diverse modificări asupra acestuia. Algoritmii de autocorelare și filtrare Butterworth sunt implementați pentru evidențierea caracteristicilor semnalelor vocale. Estimarea genului vocal se realizează prin comparația frecvenței medii a semnalului vocal cu intervale prestabilite asociate genurilor masculine și feminine. În plus, aplicația oferă funcționalități precum modificarea înălțimii vocii, redarea sunetului inversa, efectul de ecou și eliminarea zgomotului din fundal. Feedback-ul și rezultatele analizelor sunt comunicate utilizatorilor printr-un câmp de text, iar structura modulară a codului contribuie la claritate și eficiența.

# **Bibliografie**

<https://en.wikipedia.org/wiki/Butterworth_filter>

<https://www.mathworks.com/help/matlab/ref/flipud.html>

<https://www.mathworks.com/help/audio/ref/pitch.html>