Documentație Telefonie 1

Butica Cristian-Viorel

Grupa 2232, Semigrupa 1

1 Script Unu. Ton de disc, ton ocupat, ton revers apel, ton sonerie

1.1 Fundamentare teoretică

1.1.1 Ton de disc

Tonul de disc este, în general, un semnal sinusoidal continuu cu o frecvență cuprinsă între 400 și 420 Hz, transmis pe bucla de abonat. Acest semnal aparține categoriei semnalizărilor de acces, care au rolul de a informa centrala locală sau centrala PBX despre starea terminalului telefonic. Atunci când terminalul trece în stare activă (de exemplu, când utilizatorul ridică receptorul), acest gest este interpretat ca o cerere de acces la rețea. În cazul în care rețeaua are suficiente resurse disponibile pentru a continua inițierea apelului, centrala răspunde cu tonul de disc, semn că terminalul poate forma un număr. Amplitudinea semnalului este de aproximativ 0.5V.

1.1.2 Ton de ocupat

Tonul de ocupat este un semnal acustic utilizat în telefonie pentru a indica apelantului că linia numărului format este deja utilizată sau momentan indisponibilă pentru a primi apeluri. Acest ton este, în mod obișnuit, un semnal sinusoidal intermitent cu o frecvență situată în intervalul 400–420 Hz și cu o cadență caracteristică: 0.5 secunde activ (ON), urmate de 0.5 secunde inactiv (OFF). Amplitudinea sa este de aproximativ 0.5 V. Acest tip de semnal face parte din categoria semnalizărilor de supervizare, oferind apelantului informații despre starea apelului. Atunci când numărul apelat este ocupat sau există probleme în rețea care împiedică realizarea conexiunii, terminalul apelant primește acest ton specific de ocupat.

1.1.3 Ton revers apel

Tonul de revers apel face parte din categoria semnalizărilor de supervizare. Acesta este transmis către terminalul apelant atunci când apelul a fost inițiat cu succes, iar terminalul apelat începe să sune. Tonul este, de obicei, un semnal sinusoidal (sau sinusoidal modulat) intermitent, diferit însă de tonul de ocupat. El este generat de centrala telefonică la care este conectat abonatul apelat, iar caracteristicile acestuia pot varia în funcție de tipul centralei. În general, temporizarea tonului de revers apel coincide cu cea a semnalului de apel transmis către abonatul apelat.

1.1.4 Ton sonerie

Tonul de sonerie face parte din categoria semnalizărilor de alertare, având rolul de a anunța terminalul telefonic că a fost recepționat un apel. Cu alte cuvinte, acest semnal este cel care determină telefonul să sune. În cazul accesului analogic, centrala telefonică transmite către telefon un semnal aplicat direct circuitului de sonerie. De obicei, acest semnal este un semnal sinusoidal sau un semnal sinusoidal modulat în amplitudine, cu o frecvență cuprinsă între 20 și 40 Hz și o amplitudine mare, între 80 și 150V. Din cauza valorii ridicate a tensiunii, pot apărea

interferențe sau impulsuri de zgomot pe liniile vecine. Cadența semnalului de sonerie este, în mod obișnuit, de 2 secunde ON și 4 secunde OFF, însă aceasta poate varia în funcție de centrala telefonică utilizată.

1.2 Implementare tonuri în Matlab

1.2.1 Implementare ton de disc

```
Te = 1/8000 % per. de esantionare

t = 0:Te:10 % vector de timp pentru semnalul sinusoidal

ton_de_disc = 0.5*sin(2*pi*400*t) % tonul de disc, un semnal sinus

plot(t,ton_de_disc) % plotează tonul de disc în funcție de timp

player = audioplayer(ton_de_disc,1/Te) % obiectul cu ajutorul căruia putem auzi tonul
play(player) % play la ton

[spectr, f] = spectrum_analyzer(ton_de_disc, 1/Te) % analiză în domeniul frecvență
figure('Name','Spectru')

plot(f, spectr)
```

Figura 1.1: Implementare ton de disc

Acest cod va genera un semnal sinusoidal de amplitudine 0.5V, frecvență 400Hz, pe un interval de timp de 10 secunde. Vectorul de timp a fost implementat cu pasul de 1/8000, acesta reprezentând perioada de eșantionare.

Analiza în domeniul frecvență:

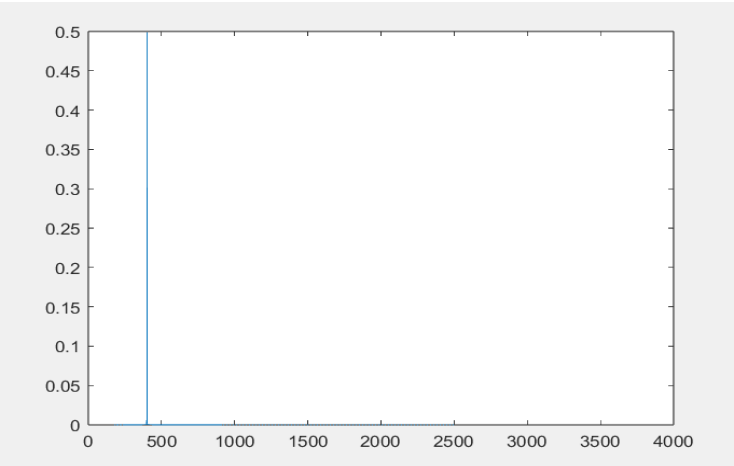

Figura 1.2: Analiză spectrală ton de disc

Conform figurii de mai sus, se observă că pe axa orizontală regăsim frecvența, exprimată în Hz, iar pe axa verticală amplitudinea semnalului sinusoidal, exprimată în volți. Acest ton de disc fiind reprezentat de un sinus, vom avea o singură componentă spectrală, de amplitudine 0.5V la frecvența de 400Hz, așa cum am definit în codul din figura 1.1.

Implementare analizor spectral:

```
function [Yfft, f] = spectrum_analyzer(signal, Fs)

fourier_transform = fft(signal); % aplicăm transformata fourier rapidă peste semnal

L = length(signal); % lungimea semnalului în eșantioane

P2 = abs(fourier_transform/L); % normalizare și păstrarea amplitudinii din valoarea complexă
P1 = P2(1:floor(L/2)+1); % convertire din spectru bilateral în unilateral
P1(2:end-1) = 2*P1(2:end-1); % înmulțire cu 2 componente pt conservarea energiei totale

f = Fs/L*(0:(L/2)); % vector de frecvențe
Yfft = P1; % amplitudinea normalizată

end
```

Figura 1.3: Implementare analizor spectral

Funcția spectrum_analyzer calculează și returnează spectrul de amplitudine al unui semnal real, folosind Transformata Fourier Rapidă (FFT). Aceasta extrage componentele frecvențiale pozitive ale semnalului, normalizează amplitudinile și generează vectorul de frecvențe corespunzător pe intervalul [0, Fs/2].

Voi folosi această funcție în toate celelalte implementări viitoare.

Analiza în domeniul timp:

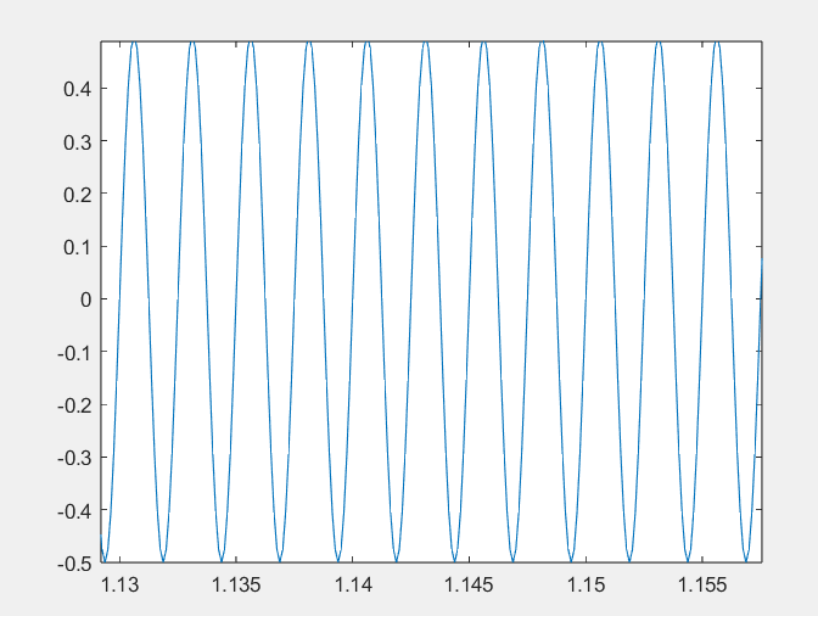

Figura 1.4: Analiză în domeniul timp pentru ton de disc

În figura de mai sus se observă un semnal sinusoidal de amplitudine 1V vârf-la-vârf, pe axa orizontală avem timpul exprimat în secunde. Am dat zoom-in pentru a putea observa oscilațiile semnalului sinusoidal.

1.2.2 Implementare ton de ocupat

```
1   A = 0.5; % amplitudine semnal
2   Te = 1/8000; % perioada de eșantionare
3   T_ON = 0.5; %perioada de ON
4   T_OFF = 0.5; % perioada de OFF
5
6   t_on = 0 : Te : T_ON - Te; % vector timp
7
8   signal_on = A*sin(2*pi*400*t_on); % generez semnalul on
9   signal_off = zeros(1, length(t_on)); % generez vector semnalul off
10
11  signal = [signal_on signal_off]; % concatenez cele 2 semnale
12
13  Numar_rep = 10; % de cate ori sa se repete semnalele on si off
14
15  signal_rep = signal;
16  for i=1:Numar_rep - 1
17      signal_rep = [signal_rep signal]; % tot timpul concatenez vectorul rezultat la un alt vector,
18      % deci va creste dimensiunea la fiecare iteratie
19  end
20
21  time_full = (0:length(signal_rep)- Te)*Te; % recalculez vectorul de timp pentru intreg semnalul
22
23  figure;
24  plot(time_full, signal_rep)
25
26  [spectr, f] = spectrum_analyzer(signal_rep, 1/Te);
27  figure('Name','Spectru')
28
29  plot(f, spectr)
30
31  sound(signal_rep, 1/Te)
```

Figura 1.5: Implementare ton de ocupat

Acest semnal este realizat prin concatenarea a unui semnal sinusoidal care are durata de 0.5 secunde, cu un semnal de pauză care are aceeași durată. Apoi, această concatenare este repetată de Numar_rep ori, în for-ul de la linia 16. Apoi se va calcula vectorul de timp pentru întreg semnalul repetat, se va plota în timp și în frecvență, la sfârșit cu ajutorul funcției sound() vom auzi acest ton de ocupat.

Analiza în domeniul frecvență:

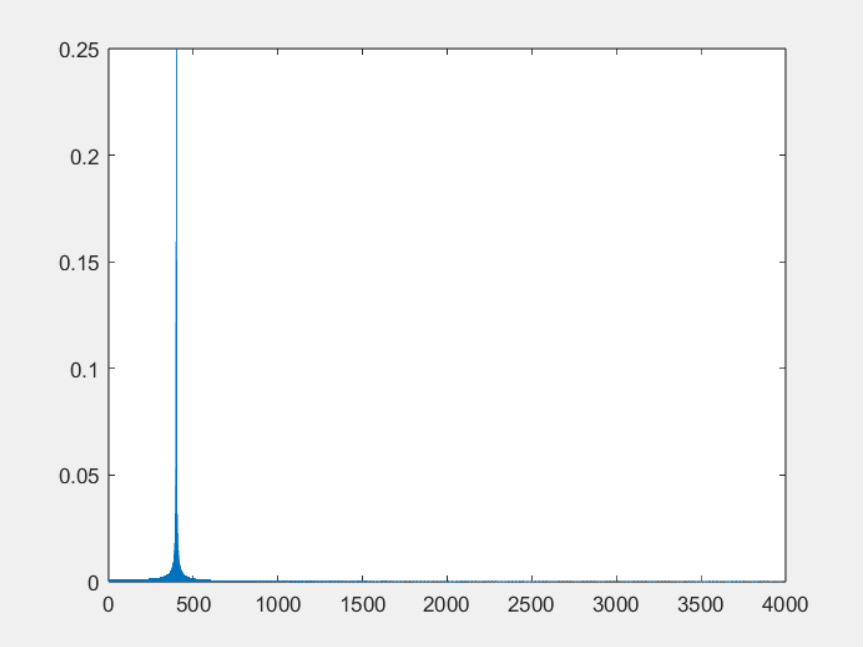

Figura 1.6: Analiză spectrală ton de ocupat

Se observă o singură componentă spectrală la frecvența de 400Hz, cu o amplitudine de 0.25V. Înjumătățirea amplitudinii semnalului sinusoidal reiese din simetria perioadelor ON și OFF, jumătate din timp semnalul este pornit, iar cealaltă jumătate este oprit.

Analiza în domeniul timp:

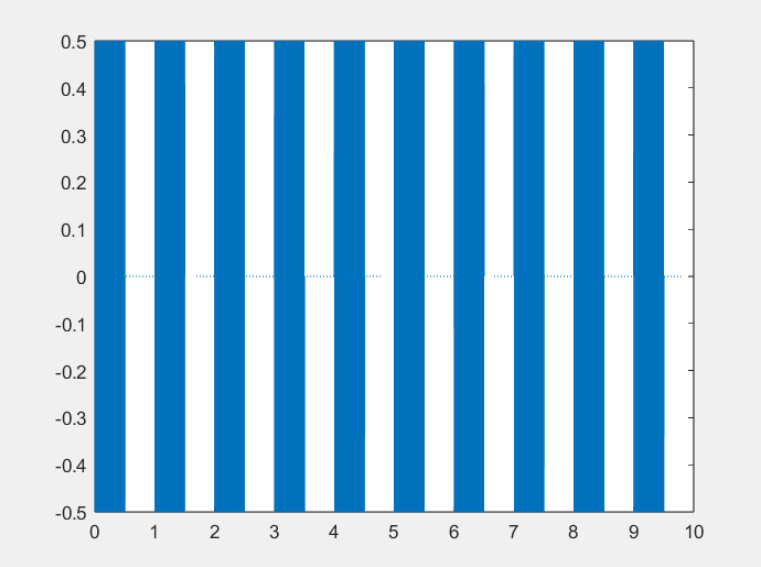

Figura 1.7: Analiză în domeniul timp pentru ton de ocupat

Pe axa orizontală se regăsește timpul în secunde, iar pe axa verticală amplitudinea semnalului în volți. Se observă cu ușurință faptul că pentru o jumătate de perioadă, semnalul este activ, iar pentru cealaltă jumătate este oprit, caracterizând un ton de ocupat clasic.

1.2.3 Implementare ton de revers apel

```
A = 0.5; % amplitudine
Te = 1/8000; % per. de esantionare
T_ON = 1; %perioada de ON
T_OFF = 4; % perioada de OFF

t_on = 0 : Te : T_ON - Te; % vector timp
t_off = 0 : Te : T_OFF - Te; % vector timp pentru signal off

signal_on = A*sin(2*pi*400*t_on); % generez semnalul on
signal_off = zeros(1, length(t_off)); % generez semnalul off

signal = [signal_on signal_off]; % concatenez cele 2 semnale

Numar_rep = 10; % de cate ori sa se repete semnalele on si off

signal_rep = signal;
for i=1:Numar_rep - 1
    signal_rep = [signal_rep signal]; % tot timpul concatenez vectorul rezultat la un alt vector,
    % deci va creste dimensiunea la fiecare iteratie
end


time_full = (0:length(signal_rep)- Te)*Te; % recalculez vectorul de timp pentru intreg semnalul

figure;
plot(time_full, signal_rep)

sound(signal_rep, 1/Te)

[spectr, f] = spectrum_analyzer(signal_rep, 1/Te);
figure('Name','Spectru')

plot(f, spectr)
```

Figura 1.8: Implementare ton de revers apel

Această implementare este foarte asemănătoare cu cea de la tonul de ocupat, singura diferență este aceea că acum perioadele de ON și OFF nu mai sunt simetrice. De regulă, pentru un ton de revers apel avem o perioadă în care semnalul este activ de 1 secundă, urmând ca acesta să fie inactiv timp de 4 secunde, desigur acest lucru depinde de centrala telefonică. Se generează semnalul sinusoidal cu durata de 1s, la care se concatenează semnalul de pauză. Apoi, se repetă această concatenare de 10 ori, se recalculează vectorul de timp final, se plotează semnalul în timp și în frecvență, și se trimite la placa de sunet.

Analiza în domeniul frecvență:

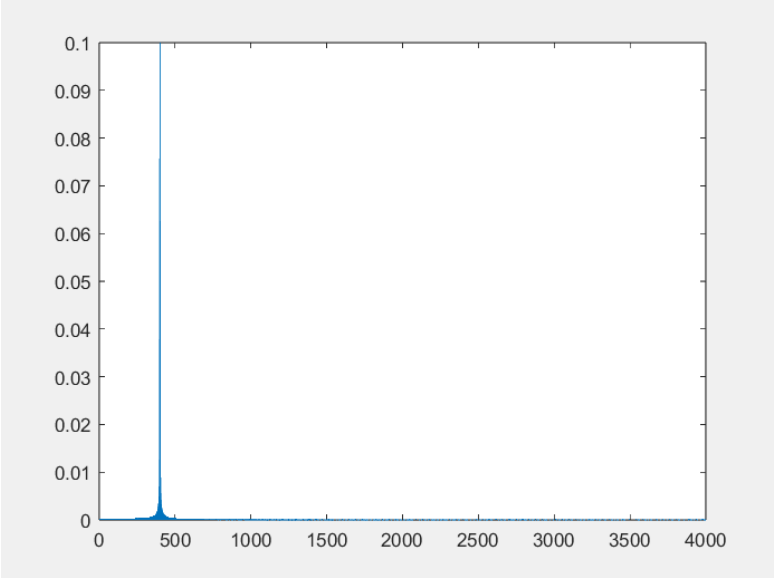

Figura 1.9: Analiză spectrală ton de revers apel

Pe axa verticală avem amplitudinea semnalului exprimată în volți, iar pe axa orizontală avem reprezentată frecvența, în Hz. Observăm că avem o singură linie spectrală la frecvența de 400Hz, cu o amplitudine de 0.1V. Dacă semnalul este activ 1s, inactiv 4s, iar perioada acestuia este de 5s, rezultă că semnalul nostru este activ 20% din timp. Conceptual, componenta principală va avea o amplitudine redusă:

$$A = 0.5V * 20\% \cong 0.1V \quad (1)$$

Analiză in domeniul timp:

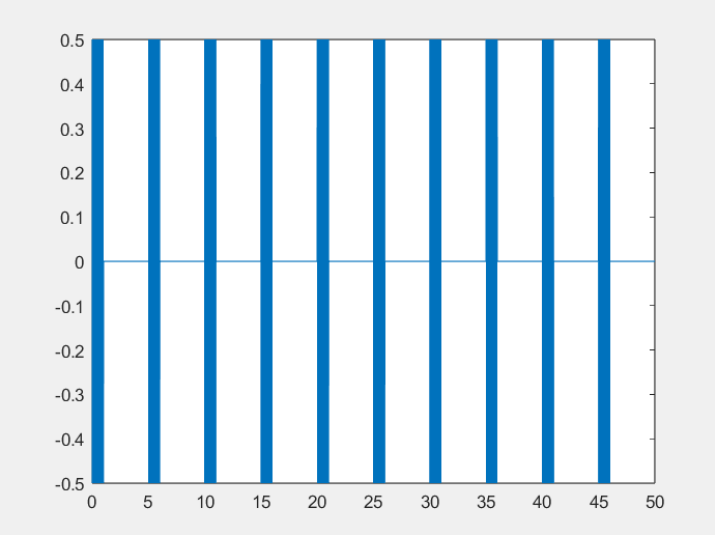

Figura 1.10: Analiză în domeniul timp pentru ton de revers apel

În figura de mai sus se observă că semnalul este activ 20% dintr-o perioadă. Pe axa orizontală avem timpul exprimat în secunde, iar pe axa verticală amplitudinea semnalului, exprimată în volți.

1.2.4 Implementare ton de sonerie

```
[y, Fs] = audioread('telephone-ring-02.wav');% citire fișier audio și extragere doar canalul 1
y = y(:, 1); % păstrăm doar primul canal
tson = 1; % cât timp e semnalul activ
tpause = 6; % cât timp e inactiv

nson = tson * Fs; % nr esantionane pt sonerie on
npause = tpause * Fs; % nr esantioane pt sonerie off\
Nrep = 3; % numar de repetari semnal on+off
sonerie = y(745 : 745 + nson - 1); % incep de la esantionul 745, doar de acolo am semnal
% creare coloană pauză (zero-uri)
pauza = zeros(npause, 1);
% concatenare verticală semnal + pauză
sonerie1 = [sonerie; pauza];
% repetare semnal Nrep ori
sonerie_rep = repmat(sonerie1, Nrep, 1);

% plot
figure;
plot(sonerie_rep);
xlabel('Timp (eșantioane)');
ylabel('Amplitudine');
title('Semnal sonerie mono');

% analiză spectrală
[s, f] = spectrum_analyzer(sonerie_rep, Fs);
% plot spectru
figure;
plot(f, s);
xlabel('Frecvență (Hz)');
ylabel('Amplitudine');
title('Spectru semnal mono');

% redare audio
sound(sonerie_rep, Fs);
```

Figura 1.11: Implementare ton de sonerie

Pentru acest ton, am folosit functia audioread(), care va citi un fisier audio și va extrage din el semnalul stereo, plus frecvența lui proprie de eșantionare. Acest semnal va fi inițial reprezentat ca 2 vectori coloană, fiecare reprezentând canalul drept și stâng. Se va păstra doar primul canal, pentru simplitatea implementării. Vom calcula câte eșantioane avem nevoie pentru fiecare semnal de sonerie + pauză. Le vom concatena pe verticală, deoarece ambele semnale sunt vectori coloană, iar după aceea cu ajutorul funcției repmat, vom repeta această concatenare de Nrep ori, adică de 3 ori. După aceea, vom plota semnalul în timp și în frecvență și îl vom trimite la placa de sunet.

Analiză în domeniul spectral:

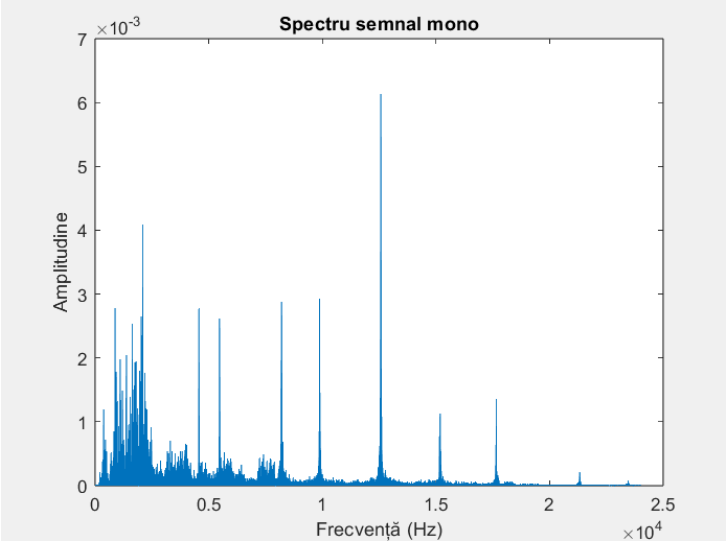

Figura 1.12: Analiză spectrală ton de sonerie

Nefiind un semnal armonic, se observă componente de o anumită amplitudine, la frecvențe diferite. Majoritatea componentelor sunt situate la frecvențe sub 12kHz, alcătuind un semnal tipic pentru tonurile de sonerie.

Analiză în domeniul timp:

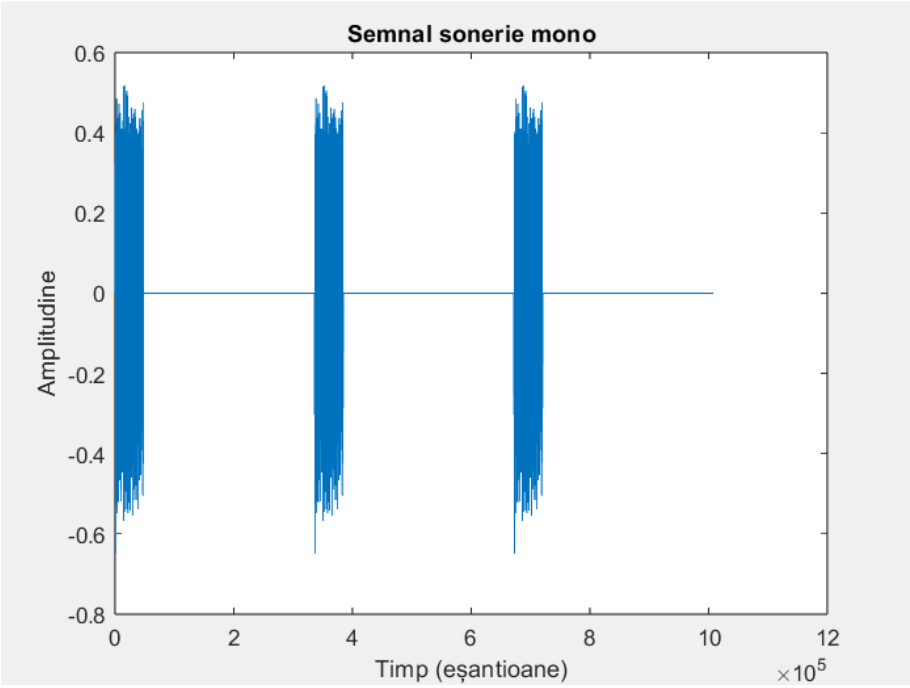

Figura 1.13: Analiză în domeniul timp pentru ton de sonerie

Se poate observa în figura de mai sus forma de undă a tonului de sonerie generat. Aceasta respectă impunerile din program, perioada în care semnalul să fie activ este de 1s, iar perioada în care aceste este inactiv de 6s.

1.3 Implementare interfață grafică în Matlab

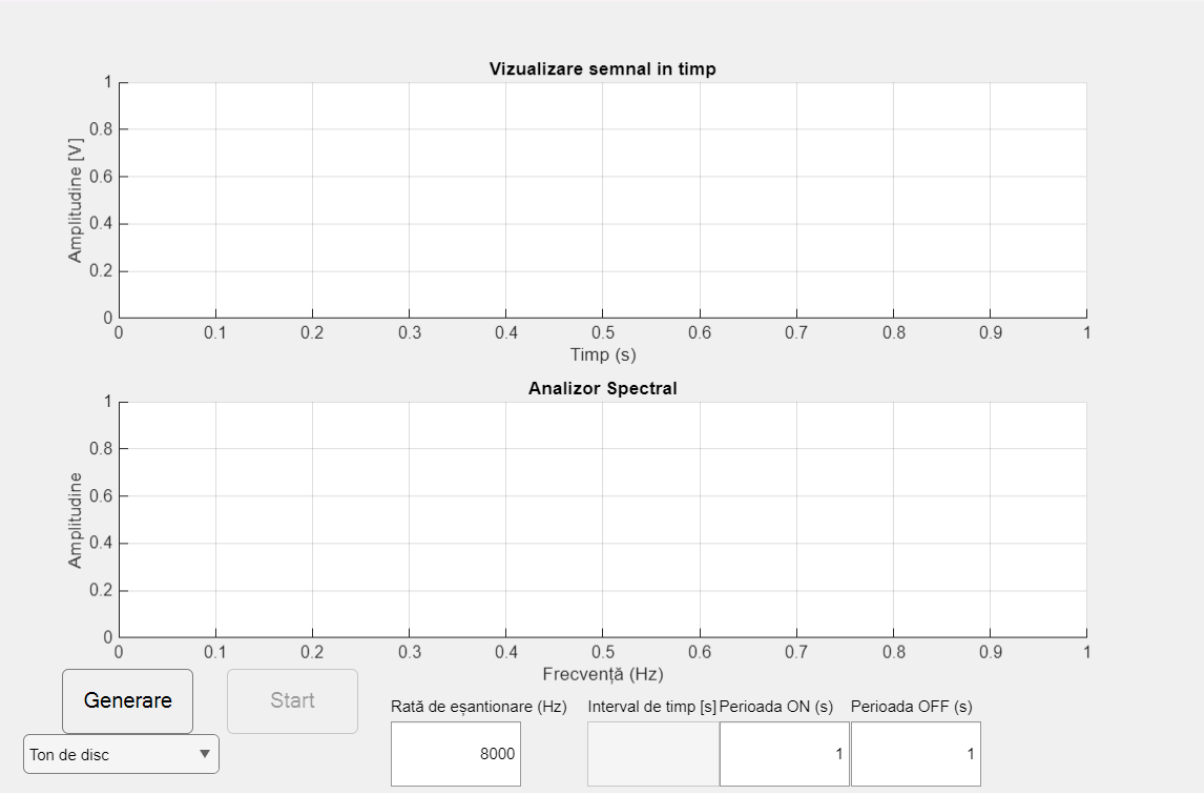

Figura 1.14: Interfață grafică tonuri

În această interfață au fost implementate următoarele componente:

- Un câmp din care se poate selecta unul dintre tonuri
- Un buton pentru generarea semnalului ales
- Vizualizarea semnalului în domeniul timp și frecvență
- Trimiterea semnalului la placa de sunet, cu ajutorul funțiilor audioplayer și sound
- Un buton ce controlează evenimentele start/stop ale semnalelor generate
- Un câmp pentru introducerea frecvenței de eșantionare
- Un câmp pentru determinarea intervalului de timp al semnalului
- Un câmp pentru modificarea perioadelor on/off ale semnalelor

Codul prin care am implementat interfața se regăsește mai jos:

```
function GUI_tonuri

T_ON_ocupat = 0.5; % per ton ocupat on+off

T_ON_revers = 1; % per ton revers on
T_OFF_revers = 4; % per ton revers off

tson_sonerie = 1; % per în care semnalul suna
tpause_sonerie = 6; % per în care semnalul nu suna

Nrep_sonerie = 10; % numărul de repetiții pt sonerie

sine_tone = uifigure('Name', 'Tonuri', ...
    'units', 'normalized', ...
    'position', [0.1 0.1 0.6 0.7]);

% figură pentru reprezentare semnal în timp
ax_signal = axes('Parent', sine_tone, 'Position', [0.1, 0.6, 0.8, 0.3]);
xlabel(ax_signal, 'Timp (s)');
ylabel(ax_signal, 'Amplitudine [V]');
title(ax_signal, 'Vizualizare semnal in timp');
grid(ax_signal, 'on');

% figură pentru reprezentare semnal în frecvență
ax_spectrum = axes('Parent', sine_tone, 'Position', [0.1, 0.2, 0.8, 0.3]);
xlabel(ax_spectrum, 'Frecvență (Hz)');
ylabel(ax_spectrum, 'Amplitudine');
title(ax_spectrum, 'Analizor Spectral');
grid(ax_spectrum, 'on');

% inițializarea tonurilor într-un dicționar
keyTones = ["Ton de disc", "Ton de ocupat", "Ton revers apel", "Ton de sonerie"];
[ton_disc, t_disc] = getTonDisc(1/8000); % se trimite perioada de esantionare initiala
[ton_ocupat, t_ocupat] = getTonOcupat(1/8000, T_ON_ocupat); % t_on_ocupat pt ca trb sa fie simetric t_on cu t_off
[ton_revers, t_revers] = getTonReversApel(1/8000, T_ON_revers, T_OFF_revers);
[sonerie_rep] = getTonSonerie(tson_sonerie, tpause_sonerie, Nrep_sonerie, 48000); % 48kHz f_es initial

t_sonerie = (0:length(sonerie_rep)-1); % vector timp pentru tonul de sonerie

% pentru fiecare cheie vom avea tonul si timpul
valueTones = {...
    {ton_disc, t_disc}, ...
    {ton_ocupat, t_ocupat}, ...
    {ton_revers, t_revers}, ...
    {sonerie_rep,t_sonerie} ...
};

% construim dicționarul
d = dictionary(keyTones, valueTones);

% etichetă pentru câmpul de rată de eșantionare
uilabel(sine_tone, ...
    "Text", "Rată de eșantionare (Hz)", ...
    "Position", [300 60 150 22], ...
    "FontSize", 12);

% edit field pentru rată de eșantionare
sampleRate = uieditfield(sine_tone, "numeric", ...
    "Position", [300 10 100 50], ...
    "Limits", [1000 16000], ...
    "Value", 8000, ...
    "ValueChangedFcn", @(src,event) changeOnOffPeriodAndSampleRate(src, d, Nrep_sonerie));

% etichetă pentru câmpul de perioadă ON
uilabel(sine_tone, ...
    "Text", "Perioada ON (s)", ...
    "Position", [550 60 150 22], ...
    "FontSize", 12);
```

```
% edit field pentru perioadă ON
TonField = uieditfield(sine_tone, "numeric", ...
    "Position", [550 10 100 50], ...
    "Limits", [0.5 15], ...
    "Value", 1, ...
    "ValueChangedFcn", @(src,event) changeOnOffPeriodAndSampleRate(src,d,Nrep_sonerie));

% etichetă pentru câmpul de perioadă OFF
uilabel(sine_tone, ...
    "Text", "Perioada OFF (s)", ...
    "Position", [650 60 150 22], ...
    "FontSize", 12);

% edit field pentru perioadă OFF
ToffField = uieditfield(sine_tone, "numeric", ...
    "Position", [650 10 100 50], ...
    "Limits", [0.5 15], ...
    "Value", 1, ...
    "ValueChangedFcn", @(src,event) changeOnOffPeriodAndSampleRate(src,d,Nrep_sonerie));

% etichetă pentru câmpul interval de timp
uilabel(sine_tone, ...
    "Text", "Interval de timp [s]", ...
    "Position", [450 60 150 22], ...
    "FontSize", 12);

% edit field needitabil pentru câmpul interval de timp
timeInterval = uieditfield(sine_tone, "text", ...
    "Position", [450 10 100 50], ...
    "Enable","off");

% buton start-stop semnal
bStartStop = uicontrol(sine_tone, ...
    'Style', 'pushbutton', ...
    'Enable', 'off', ...
    'String', 'Start', ...
    'FontSize', 12, ...
    'Position', [175 50 100 50], ...
    'Callback', @(src, event) startStop(src, ax_signal,ax_spectrum,d,sampleRate.Value,sine_tone));

% buton generare semnal
bGenerare = uicontrol(sine_tone, ...
    'Style', 'pushbutton', ...
    'Enable', 'on', ...
    'String', 'Generare', ...
    'FontSize', 12, ...
    'Position', [50 50 100 50], ...
    'Callback', @(src, event) updatePlot(src, ax_signal,ax_spectrum,d,sampleRate.Value,bStartStop,sine_tone, ...
    timeInterval));

% meniu drop-down pentru selectare tonuri
dd = uidropdown(sine_tone, ...
    "Items", keyTones, ...
    "Position", [20, 20, 150, 30], ...
    "ValueChangedFcn", @(src,event) setTone(src, d, bGenerare,bStartStop,sampleRate,Nrep_sonerie));

% structură în care salvez toate componentele de care am nevoie, butoane...
guidata(sine_tone, struct('bStartStop', bStartStop, 'bGenerare', bGenerare,'sampleRate',sampleRate,'tonField', ...
    TonField,'toffField',ToffField, 'toneMenu',dd));

end
```

Figura 1.15: Implementare interfață grafică tonuri

Prima dată vom avea nevoie să creăm fereastra principală a aplicației, cu o componentă uifigure, și vom inițializa perioadele de on/off ale semnalelor cu niște valori implicite, urmând a putea fi schimbate din interfață. Creăm încă 2 figuri pentru reprezentarea semnalului ales în timp și frecvență, iar după aceea construim un dicționar, pe baza căruia vom putea alege unul dintre tonurile disponibile în meniul drop-down.

- **keyTones** – cheile tonurilor, pe care le voi trimite ca și elemente pentru meniul de selectare
- **valueTones** – pentru fiecare cheie din dictionar (adică numele tonului), vom avea semnalul și timpul corespunzător. Aceste valori au fost returnate de niște funcții getTonDisc, getTonOcupat, etc, foarte asemănătoare cu funcțiile din subcapitolul 1.2
- Câmp modificare rată de eșantionare – va avea inițial valoarea 8000Hz, cu limite între 1kHz și 16kHz, iar de fiecare dată când se modifică valoarea acestui câmp, funcția setSampleRate(...) va fi apelată
- Câmpuri editare perioade on/off – implicit valoarea de 1s, cu limite între 1-15 secunde, iar când aceste câmpuri sunt modificate, se apelează funcția changeOnOffPeriod(...)
- Câmp pentru determinarea intervalului de timp – acest câmp va determina durata semnalului ales. Desigur, acest câmp este needitabil
- **bStartStop**– Acest buton poate fi activat atunci când semnalul a fost generat dinainte, inițial fiind dezactivat. La apăsare se execută funcția startStop(...)
- **bGenerare** – Butonul prin care generăm tonul ales. La apăsare se execută funcția updatePlot(...)
- dd – meniul drop-down, din care se poate selecta un anumit ton, mai apoi putându-se genera. La modificarea acestui meniu, se va apela funcția setTone(...)
- Guidata(...) – în această structură salvez toate componentele de care am nevoie, pentru a nu fi nevoit să le trimit ca și parametri la funcții

Mai jos voi prezenta pe scurt funcțiile pentru determinarea tonurilor:

```
function [ton_de_disc, t] = getTonDisc(Te)

t = 0:Te:10; % durata tonului
ton_de_disc = 0.5*sin(2*pi*400*t); %  tonul de amplitudine 0.5V si frecvență 400Hz

end
```

Figura 1.16: Implementare funcție getTonDisc

Această funcție primește ca și parametru perioada de eșantionare și calculează vectorul de timp pentru semnalul sinusoidal care reprezintă tonul de disc. Returnează tonul și vectorul timp.

```
function [sonerie_rep] = getTonSonerie(tson,tpause,Nrep,Fs_field)

[y, Fs] = audioread('telephone-ring-02.wav');
y = y(:, 1); % păstrăm doar primul canal

nson = tson * Fs_field; % nr esantionane pt sonerie on
npause = tpause * Fs_field; % nr esantioane pt sonerie off

sonerie = y(745 : 745 + nson - 1); % incep de la esantionul 745, doar de acolo am semnal

% creare coloană pauză (zero-uri)
pauza = zeros(npause, 1);

% concatenare verticală semnal + pauză
sonerie1 = [sonerie; pauza];

% repetare semnal Nrep ori
sonerie_rep = repmat(sonerie1, Nrep, 1);

end
```

Figura 1.17: Implementare funcție getTonSonerie

Această funcție primește ca și parametri:

- Tson – reprezintă timpul în care semnalul este activ
- Tpause – timpul în care semnalul este inactiv
- Nrep – numărul de repetiții ale semnalului de sonerie + semnal de pauză
- Fs_field - frecvența de eșantionare, dată de câmpul de setare a frecvenței din interfață

Semnalul returnat de funcția audioread este un semnal stereo, iar pentru simplitate am selectat doar primul canal, adică primul vector coloana din semnal. Se calculează numărul de eșantioane necesare pentru sonerie on, respectiv sonerie off, se va crea un vector coloană cu elemente zero, reprezentând semnalul de pauză, vom concatena aceste 2 semnale, și le vom repeta de Nrep ori. La final, funcția returnează semnalul repetat, adică tonul de sonerie propriu-zis.

```
function [signal_rep,time_full] = getTonOcupat(Te,T_ON)

t_on = 0 : Te : T_ON - Te; % vector timp pt ton si toff, sunt de aceeasi lungime

signal_on = 0.5*sin(2*pi*400*t_on); % generez semnalul on
signal_off = zeros(1, length(t_on)); % generez semnalul off

signal = [signal_on signal_off]; % concatenez cele 2 semnale

Numar_rep = 10; % de cate ori sa se repete semnalele on si off

signal_rep = signal;
for i=1:Numar_rep - 1
    signal_rep = [signal_rep signal]; % tot timpul concatenez vectorul rezultat la un alt vector,
    % deci va creste dimensiunea la fiecare iteratie
end

time_full = (0:length(signal_rep)- Te); % recalculez vectorul de timp pentru intreg semnalul

end
```

Figura 1.18: Implementare funcție getTonOcupat

Implementarea este foarte asemănătoare cu funcțiile din subcapitolul 1.2, ceea ce este diferit este că funcția primește ca și parametri perioada de eșantionare și perioada în care semnalul este activ. Fiind vorba de un ton de ocupat, s-a decis ca vectorii de timp pentru semnal on și off să fie de aceeași lungime, deci s-a folosit T_ON pentru ambii. Funcția va returna tonul de ocupat și vectorul timp.

```
function [signal_rep,time_full] = getTonReversApel(Te,T_ON, T_OFF)

t_on = 0 : Te : T_ON - Te; % vector timp pt signal on
t_off = 0 : Te : T_OFF - Te; % vector timp pentru signal off

signal_on = 0.5*sin(2*pi*400*t_on); % generez semnalul on
signal_off = zeros(1, length(t_off)); % generez semnalul off

signal = [signal_on signal_off]; % concatenez cele 2 semnale

Numar_rep = 10; % de cate ori sa se repete semnalele on si off

signal_rep = signal;
for i=1:Numar_rep - 1
    signal_rep = [signal_rep signal]; % tot timpul concatenez vectorul rezultat la un alt vector,
    % deci va creste dimensiunea la fiecare iteratie
end

time_full = (0:length(signal_rep)- Te); % recalculez vectorul de timp pentru intreg semnalul

end
```

Figura 1.19: Implementare funcție getTonReversApel

Funcția este foarte asemănătoare cu cea din subcapitolul 1.2, doar trimitem în plus ca și parametri perioadele în care semnalul este activ, respectiv inactiv. Se returnează tonul de revers apel și vectorul de timp corespunzător.

```
% functie pentru schimbarea perioadelor ON/OFF ale tonurilor și modificarea
% ratei de eșantionare
function changeOnOffPeriodAndSampleRate(src, d, Nrep_sonerie)

    % preiau toate componentele salvate anterior pentru ușurința
    % implementării
    components = guidata(src);

    % reapelez funcțiile pt determinarea tonurilor, cu parametri din
    % edit_field-uri
    [semnal_sonerie] = getTonSonerie(components.tonField.Value, components.toffField.Value, Nrep_sonerie, ...
        components.sampleRate.Value);
    t_sonerie = (0:length(semnal_sonerie)-1); % recalculez vectorul de timp pt sonerie

    d("Ton de sonerie") = {{semnal_sonerie, t_sonerie}};

    [ton_disc, t_disc] = getTonDisc(1/components.sampleRate.Value);
    d("Ton de disc") = { {ton_disc, t_disc} };

    [ton_ocupat, t_ocupat] = getTonOcupat(1/components.sampleRate.Value, components.tonField.Value);
    d("Ton de ocupat") = { {ton_ocupat, t_ocupat} };

    [ton_revers, t_revers] = getTonReversApel(1/components.sampleRate.Value, components.tonField.Value, ...
        components.toffField.Value);
    d("Ton revers apel") = { {ton_revers, t_revers} };

    % actualizez tonul ales în UserData-ul butoanelor, voi avea nevoie mai
    % jos în cod
    components.bStartStop.UserData = d(components.toneMenu.Value);
    disp(components.toneMenu.Value);
    components.bGenerare.UserData = d(components.toneMenu.Value);

    % dacă am schimbat perioadele ON/OFF sau rata de eșantionare, trebuie să generez din nou tonul,
    % abia după aceea pot folosi butonul start/stop
    components.bStartStop.Enable = 'off';
    components.bGenerare.Enable = 'on';
end
```

Figura 1.20: Implementare funcție changeOnOffPeriodAndSampleRate

De fiecare dată când modific unul dintre câmpurile de perioadă on/off sau rata de eșantionare, se va apela această funcție. Se va actualiza dicționarul de tonuri, trimițând la funcțiile getTonDisc, etc, valorile perioadelor on/off din câmpurile aferente, plus frecvențele/perioadele de eșantionare corespunzătoare. Apoi stochez tonul actualizat și ales din meniu în userData-ul butoanelor de generare și start/stop, deoarece voi avea nevoie de el mai jos în cod. Desigur, o dată ce perioadele on/off sau rata de eșantionare au fost modificate, tonul trebuie regenerat, deci butonul de generare va deveni activ, iar cel de start/stop inactiv.

Funcție pentru setarea tonului ales din meniul drop-down:

```
% funcție pentru setarea tonului curent din meniul drop-down, aceasta se
% execută de fiecare dată când aleg alt ton din meniu
function setTone(src, d, bGenerare,bStartStop,sampleRate,Nrep_sonerie)

    components = guidata(src);
    % salvez tonul selectat în UserData-ul butoanelor start/stop și
    % generare
    tonSelectat = src.Value;
    bGenerare.UserData = d(tonSelectat);
    bStartStop.UserData = d(tonSelectat);
    % după ce am selectat tonul, trebuie întâi să îl generez
    bGenerare.Enable = 'on';
    bStartStop.Enable = 'off';
    % intervale și valori implicite diferite pentru ton de sonerie și
    % celelate tonuri
    if tonSelectat == "Ton de sonerie"
        sampleRate.Limits = [1000 96000]; % pot balea frecvența de eșantionare în acest interval
        sampleRate.Value = 48000;
        % trebuie actualizat tonul de sonerie în dicționar pentru că am modificat
        % frecvența de eșantionare, ea fiind inițial de 8kHz
        [semnal_sonerie] = getTonSonerie(components.tonField.Value, components.toffField.Value, Nrep_sonerie, ...
        components.sampleRate.Value);
        t_sonerie = (0:length(semnal_sonerie)-1); % recalculez vectorul de timp pentru sonerie

        d("Ton de sonerie") = {{semnal_sonerie, t_sonerie}};

    else
        % pentru celelalte tonuri am alt interval de frecvență și altă
        % valoare implicită
        sampleRate.Limits = [1000 16000];
        sampleRate.Value = 8000;

    end
end
```

Figura 1.21: Implementare funcție setTone

Aceasta va salva tonul selectat în UserData-ul butoanelor, va activa butonul de generare semnal, și va modifica valoarea implicită și intervalul pentru câmpul de schimbare a frecvenței de eșantionare, în funcție de semnalul ales.

Funcție pentru generarea tonurilor:

```
% funcție pentru generarea tonurilor, aceasta se apelează de fiecare dată
% când apăs pe butonul de generare
function updatePlot(src, ax_signal, ax_spectrum, d,Fs,bStartStop,sine_tone,timeInterval)

    persistent player; % dacă nu îl făceam persistent,
    % după ce se apela funcția obiectul player se distrugea și nu se mai auzea nimic

    % verificăm dacă UserData-ul e gol
    if isempty(src.UserData)
        data = d("Ton de disc"); % când se porneste plot-ul pt prima dată nu am
        % nimic în userdata, deci inițializez cu ton de disc
    else
        data = src.UserData;
    end

    % preiau toate componentele din structură
    components = guidata(src);

    % am generat semnalul, înseamnă că acum pot doar să îi dau stop
    bStartStop.Enable = 'on';
    bStartStop.String = 'Stop';

    % dacă vreau să generez alt ton, dar deja a fost generat unul, trebuie
    % să șterg axele întâi pentru a nu suprapune plot-urile unul
    % peste altul
    cla(ax_signal);
    cla(ax_spectrum);

    % daca nu am hold on, plot-ul de mai jos va recrea de fapt axa ca și
    % cum ar fi una nouă, deci aici salvez xlabel, ylabel, etc.
    hold(ax_signal, 'on');
    hold(ax_spectrum, 'on');

        inner_data = data{1}; % din nou valoarea tonului este un cell array
        tone = inner_data{1}; % tonul propriu zis e pe prima pozitie din array

        % la toate tonurile trebuie să împart timpul
        % cu frecvență de eșantionare, pentru a afla de fapt timpul în
        % secunde, altfel voi avea de fapt numărul de eșantioane
        t = inner_data{2}/components.sampleRate.Value; % timpul pe a 2-a poz in array

        % plot in timp si spectru
        plot(ax_signal, t, tone, 'b', 'LineWidth', 1.5);

        [Yfft, f] = spectrum_analyzer(tone, Fs);
        plot(ax_spectrum, f, Yfft, 'r', 'LineWidth', 1.5);

        hold(ax_signal, 'off');
        hold(ax_spectrum, 'off');

        player = audioplayer(tone, Fs);
        play(player);
        setappdata(sine_tone, 'audioPlayer', player);  % salvez player in appdata

        src.Enable = 'off';

        timeInterval.Value = num2str(length(tone)/Fs);

end
```

Figura 1.22: Implementare funcție updatePlot

Funcția va genera în mod dinamic tonul ales, adică va plota semnalul în timp și frecvență, îl va trimite spre placa de sunet, iar dacă semnalul a fost generat în prealabil și doresc să îl regenerez, aceasta va curăța ambele plot-uri, păstrând axele și etichete lor corespunzătoare, și nu va suprapune semnalul de fiecare dată când apăs pe butonul de generare.

Funcție pentru gestionarea evenimentelor start/stop ale semnalelor generate:

```
% funcție pentru controlarea evenimentelor start/stop ale semnalelor
% generate
function startStop(src, ax_signal, ax_spectrum, d,Fs,sine_tone)

    % preiau player-ul din appdata, salvat in functia updatePlot
    player = getappdata(sine_tone,'audioPlayer');

    % preiau componentele din structura de la început
    components = guidata(src);

    if isempty(src.UserData)
        data = d("Ton de disc"); % când se pornește plot-ul pt prima dată nu am nimic în UserData,
        % deci inițializez cu ton de disc
    else
        data = src.UserData;
    end

    % dacă semnalul este pornit (generat) și apăs pe butonul start/stop, înseamnă că
    % doresc să îl opresc. Butonul are textul 'Stop' în această situație
    if strcmp(src.String, 'Stop') && strcmp(src.Enable, 'on')

        stop(player);
        cla(ax_signal); % stergem axele
        cla(ax_spectrum);
        src.String = 'Start'; % după ce apăs pe buton, modific textul la 'Start'

    % semnalul e oprit in acest moment, dacă apăs acum pe buton, vreau să
    % îl repornesc
    elseif strcmp(src.String, 'Start') && strcmp(src.Enable, 'on')

        hold(ax_signal, 'on');  % mențin xlabel, ylabel, etc.
        hold(ax_spectrum, 'on');

            inner_data = data{1};
            tone = inner_data{1};

            t = inner_data{2}/components.sampleRate.Value;

            plot(ax_signal, t, tone, 'b', 'LineWidth', 1.5);

            [Yfft, f] = spectrum_analyzer(tone, Fs);
            plot(ax_spectrum, f, Yfft, 'r', 'LineWidth', 1.5);

            % revenire la comportament inițial pentru a nu suprapune
            % plot-urile
            hold(ax_signal, 'off');
            hold(ax_spectrum, 'off');

            play(player);
            src.String = 'Stop'; % modifică textul butonului în 'Stop' după apăsare

    end
end
```

Figura 1.23: Implementare funcție startStop

Această funcție este destul de asemănătoare cu updatePlot(...), diferența constă în logica prin care se poate folosi acest buton start/stop:

- Dacă tonul nu a fost generat în prealabil, nu voi putea folosi acest buton, el fiind implicit inactiv.
- Dacă tonul a fost deja generat, înseamnă că pot doar să îl opresc, deci butonul va avea funcția de stop. În momentul în care butonul este acționat, axele se vor șterge și player-ul se va opri.
- Dacă semnalul a fost oprit cu acest buton, înseamnă că prin reapăsarea lui vreau să îl repornesc.

1.4 Interpretare rezultate experimentale

1.4.1 Interpretare rezultate ton de disc

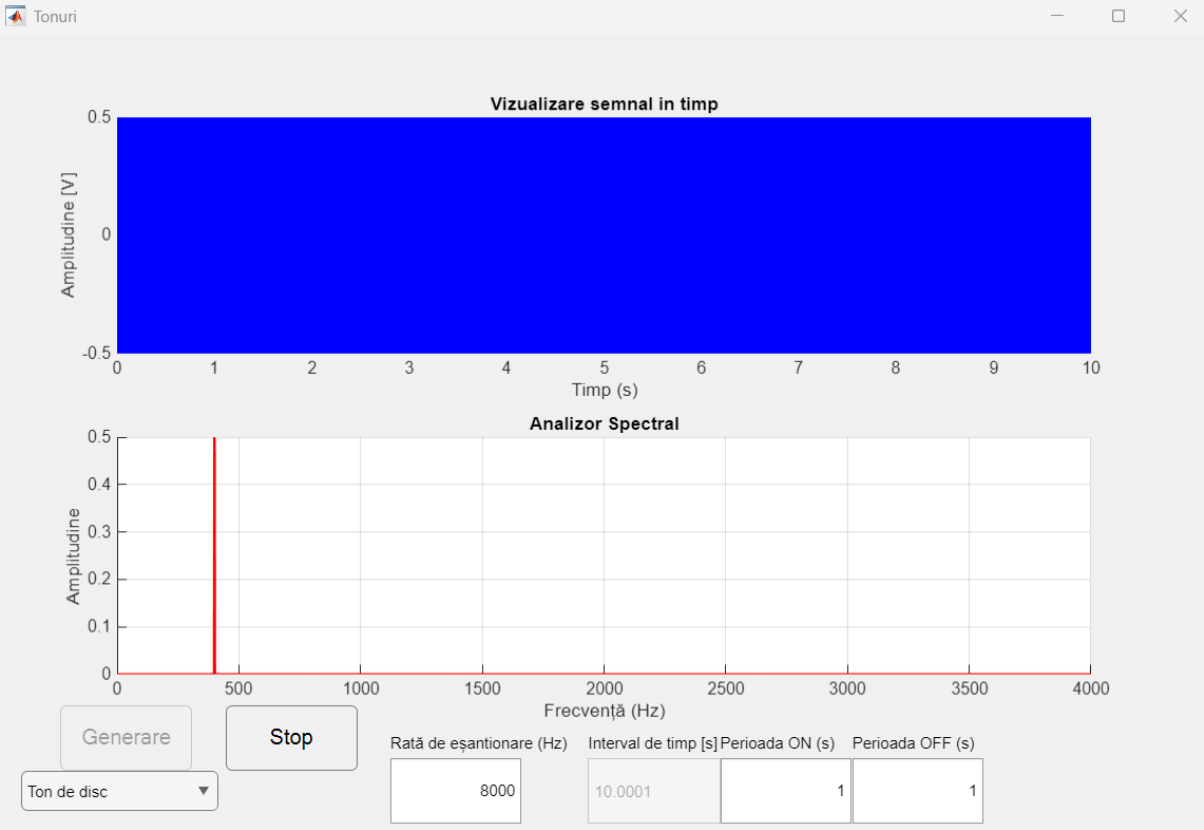

Figura 1.24: Intepretare rezultate ton de disc

Se poate observa tonul de disc de o durată de 10 secunde, având o rată de eșantionare de 8kHz. Fiind vorba de un semnal sinusoidal, avem desigur o singură componentă spectrală de amplitudine 0.5V, la frecvența de 400Hz.

1.4.2 Interpretare rezultate ton de ocupat

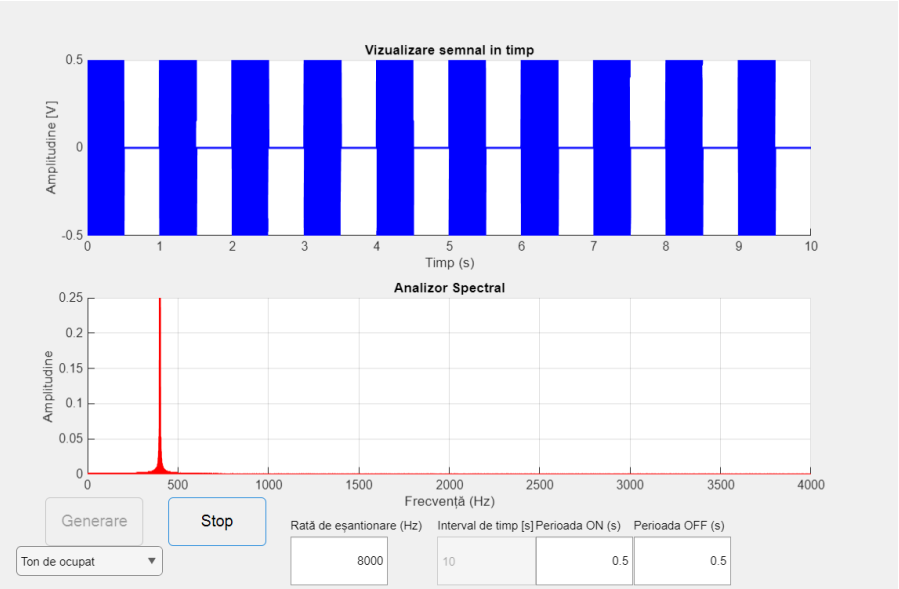

Figura 1.25: Intepretare rezultate ton de ocupat

Caracteristica tonului de ocupat este simetria perioadelor de on/off, iar din acest grafic se poate observa cum într-o perioadă de 1s, semnalul este activ 0.5s, respectiv inactiv timp de 0.5s. Intervalul de timp al acestui ton este de 10s, iar rata de eșantionare implicită este de 8kHz. Se observă o componentă spectrală de amplitudine 0.25V la frecvența de 400Hz.

Să variem rata de eșantionare la 1kHz:

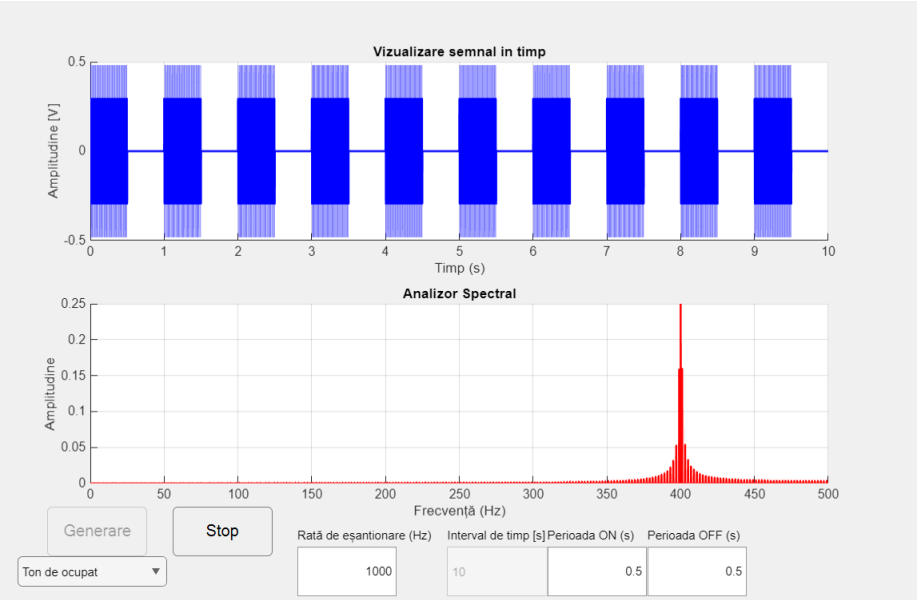

Figura 1.26: Intepretare rezultate ton de ocupat, rată de eșantionare 1 kHz

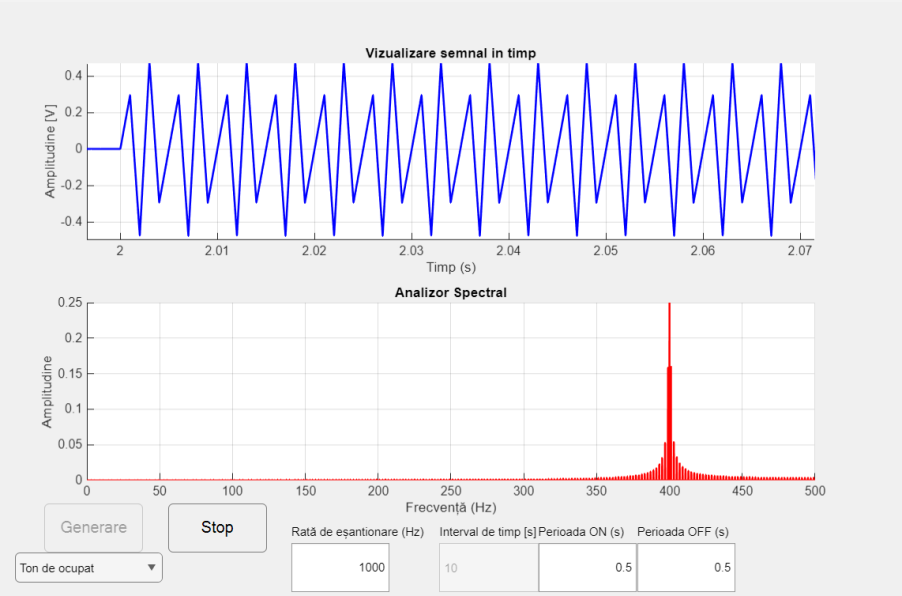

Figura 1.27: Intepretare rezultate ton de ocupat, rată de eșantionare 1 kHz, zoom-in

Din figurile 1.26, respectiv 1.27 se observă că semnalul începe să devină ușor distorsionat, acesta nu mai seamănă atât de mult cu un sinus clasic. În analizorul spectral, apar niște componente in jurul armonicei fundamentale, fapt care denotă prezența zgomotului. Acest lucru poate fi datorat faptului că suntem la limita teoremei lui Nyquist, frecvența semnalului fiind de 400Hz. Totuși, durata acestuia rămâne neschimbată.

Să variem și perioadele on/off:

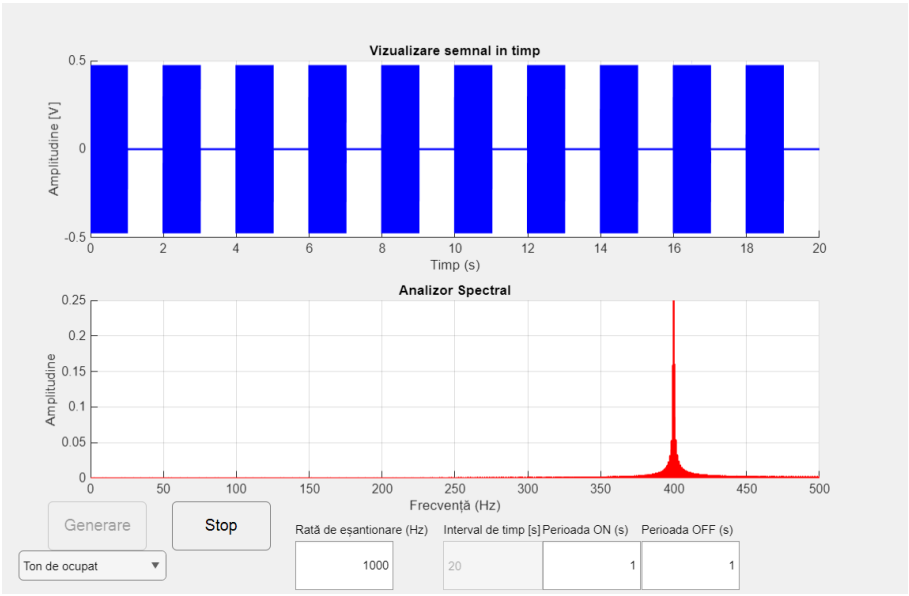

Figura 1.28: Intepretare ton de ocupat, rată de eșantionare 1 kHz, perioade on/off modificate

În figura de mai sus, se observă că perioadele de on/off au fost schimbate ambele la câte 1s, rezultând ca semnalul să dureze un timp total de 20s.

1.4.3 **Interpretare rezultate ton de revers apel**

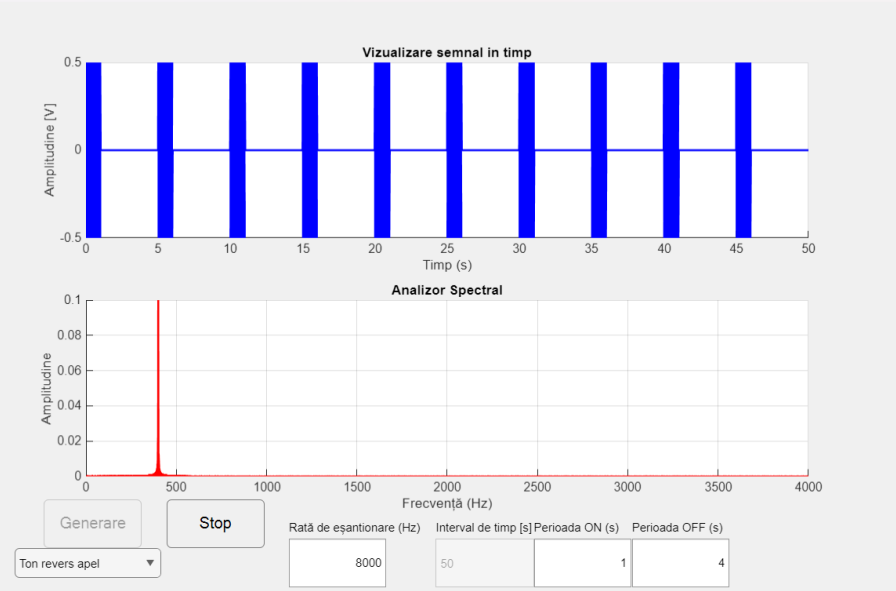

Figura 1.29: Intepretare ton de revers apel, rată de eșantionare 8 kHz

Pentru tonul de revers apel, amplitudinea componentei spectrale a fost explicată în subcapitolul 1.2.3. Se observă că intervalul de timp este de 50s, deoarece fiecare perioadă a semnalului este de 5s, iar acesta a fost repetat de 10 ori în funcția getTonReversApel(...).

Să variem rata de eșantionare la 1kHz:

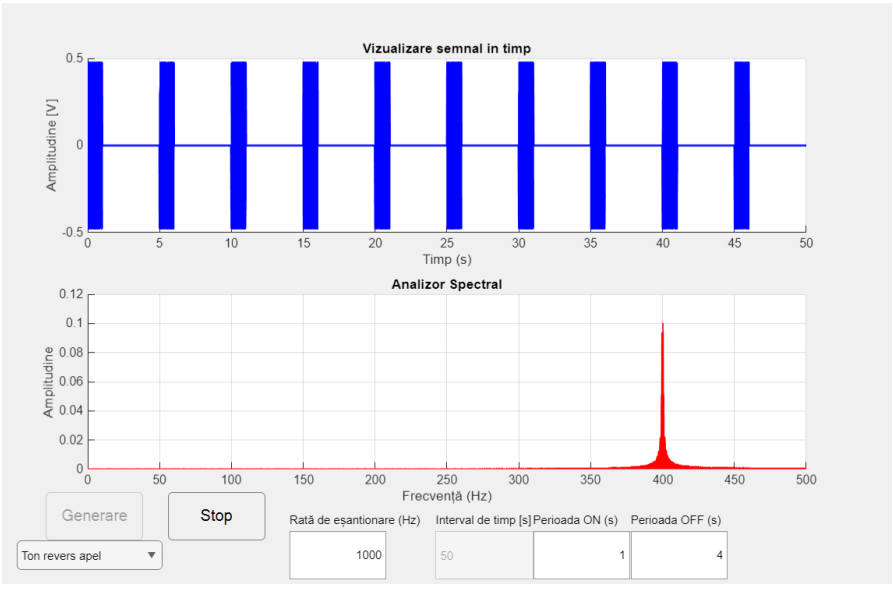

Figura 1.30: Intepretare ton de revers apel, rată de eșantionare 1 kHz

Odată ce am schimbat frecvența de eșantionare și ne apropiem de dublul frecvenței semnalului, se observă cum componenta spectrală începe să fie afectată de zgomot. Din nou, intervalul de timp rămâne același.

Să variem și perioadele on/off:

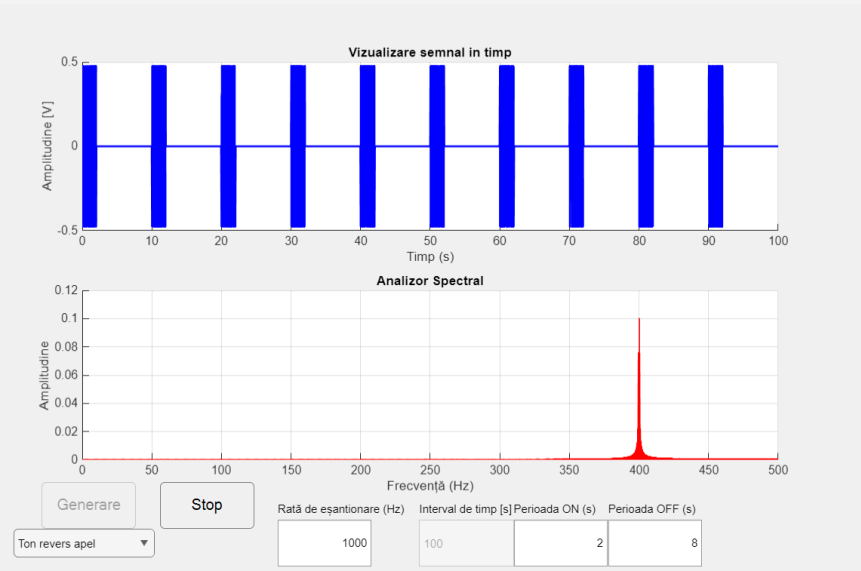

Figura 1.31: Intepretare ton de revers apel, perioade on/off dublate

Dublând astfel perioadele on/off, rezultă că tonul de revers apel va avea de 2 ori durata sa inițială, adică 100s. Pentru 2 secunde acesta va fi activ, iar pentru 8 secunde inactiv. O perioadă durează 10 secunde, repetată de 10 ori rezulta exact 100s, durata tonului.

1.4.4 Interpretare rezultate ton de sonerie

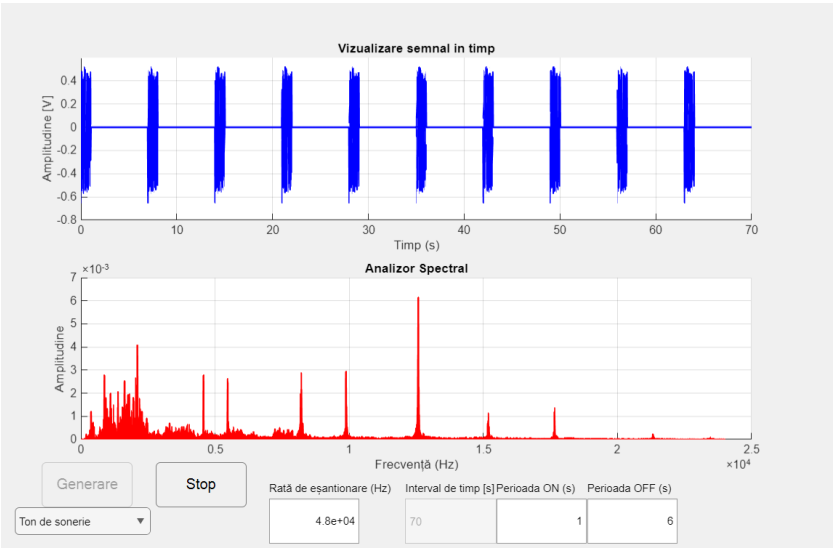

Figura 1.32: Intepretare ton de sonerie, rată de eșantionare 48kHz

În figura de mai sus, se observă tonul de sonerie în timp și în frecvență, acesta având componente spectrale la frecvențe mai mici de aproximativ 12kHz. Implicit, frecvența de eșantionare este setată la 48kHz, iar perioadele on/off sunt de 1s, respectiv 6s. Perioada semnalului este de 7 secunde, repetată de 10 ori rezultă durata totală de 70 secunde.

Să modificăm rata de eșantionare la 30kHz:

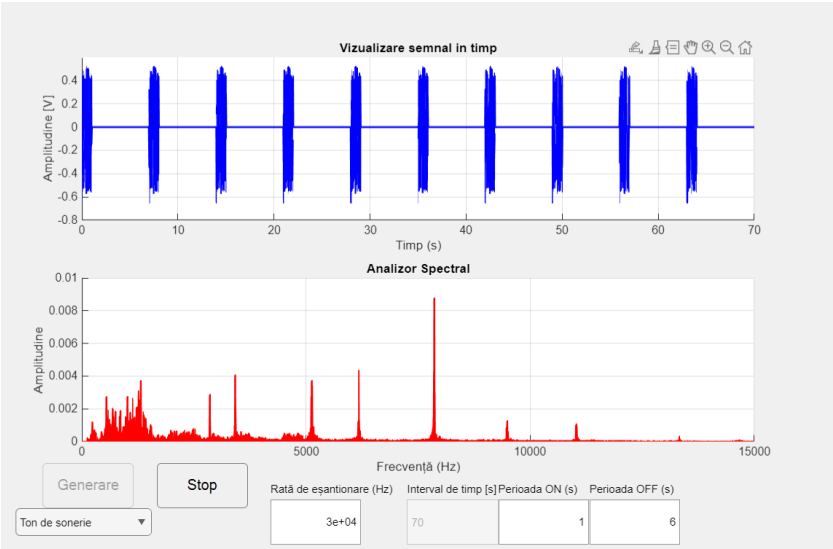

Figura 1.33: Intepretare ton de sonerie, rată de eșantionare 30kHz

Frecvența maximă a tonului de disc era situată la aproximativ 12kHz. Apropiindu-ne de $2*f_{max}$ cu frecvența de eșantionare, observăm cum semnalul se distorsionează, avem componente de amplitudini diferite la alte frecvențe decât cum era inițial.

Să modificăm și perioadele on/off ale semnalului:

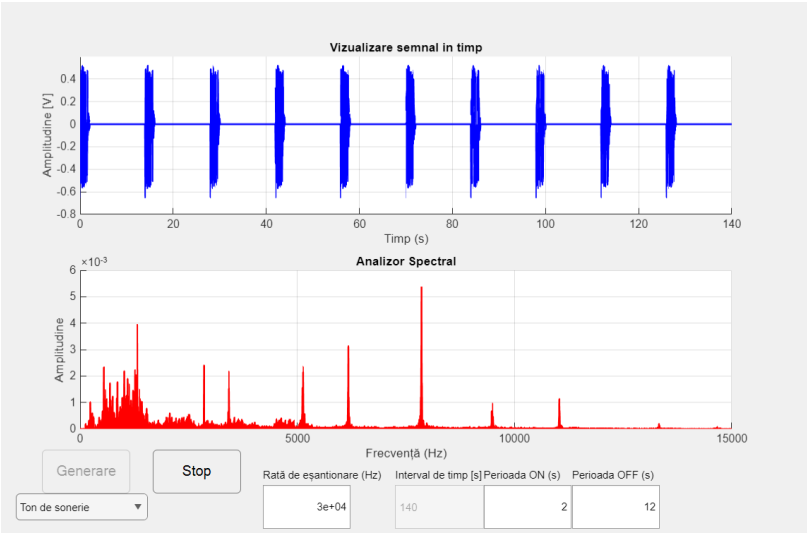

Figura 1.34: Intepretare ton de sonerie, perioade on/off modificate

Odată ce am dublat valorile perioadelor on/off ale semnalului, timpul total s-a dublat și el. Semnalul este activ 2 secunde din perioadă, inactiv 12s. O perioadă este de 14 secunde, repetată de 10 ori rezultă durata tonului de sonerie este acum de 140 secunde.

2 Script Doi. Semnalizare de adresă (Impuls și tonuri DTMF)

2.1 Fundamentare teoretică

2.1.1 Semnalizarea de adresă

Semnalizarea de adresă presupune codificarea și transmiterea, într-un format specific, a numărului abonatului apelat către centrala locală. În cazul conexiunilor analogice, cifrele sau simbolurile unui număr de telefon (sau ale unui identificator de echipament, terminal ori serviciu) pot fi reprezentate fie printr-o succesiune de impulsuri – cunoscută ca metoda prin impulsuri sau apel prin puls – fie printr-o combinație de tonuri, utilizând metoda DTMF (Dual Tone Multi Frequency).

2.1.2 Semnalizare prin impuls

Fiecare cifră din numărul format este transmisă sub forma unui anumit număr de impulsuri. Aceste impulsuri sunt generate prin întreruperea temporară a curentului din bucla abonatului. Numărul de impulsuri corespunde valorii fiecărei cifre, cu excepția cifrei 0, care este reprezentată prin 10 impulsuri.

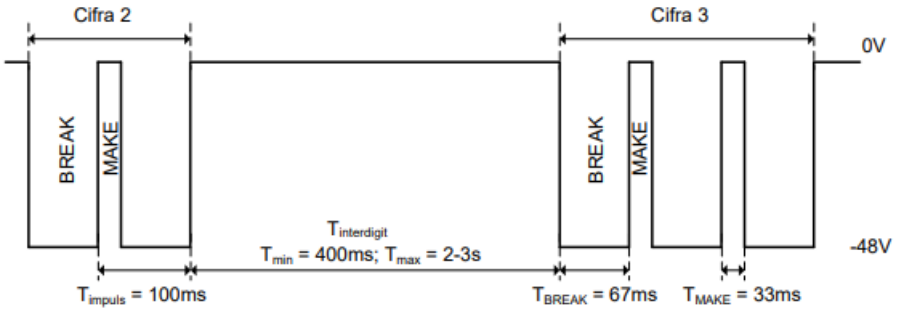

Figura 2.1 Transmiterea numărului prin impulsuri [1]

- Observație: întreruperile scurte ale curentului de buclă, în ordine de zeci sau sute de milisecunde, nu conduc la deconectarea telefonului. Pentru deconectare este necesară o întrerupere de câteva secunde.

Durata întreruperii buclei (perioada BREAK) este de aproximativ 67 ms (TBREAK ≈ 67 ms), iar durata de restabilire a conexiunii între două întreruperi consecutive (perioada MAKE) este de circa 33 ms (TMAKE ≈ 33 ms). Rata nominală de transmitere a impulsurilor este de 10 impulsuri pe secundă (10 imp/s). În practică, centralele telefonice acceptă o anumită marjă față de această rată nominală, cu valori minime acceptate de 8–9 imp/s și maxime de 11–12 imp/s. Timpul de așteptare între transmiterea a două cifre consecutive (timpul interdigit) variază între câteva sute de milisecunde și câteva secunde.

2.1.3 Semnalizare prin tonuri DTMF (Dual-tone multi-frequency)

Pentru fiecare cifră sau simbol transmis prin apelarea DTMF (Dual Tone Multi Frequency), se utilizează o combinație unică de două frecvențe: una dintr-un set inferior (F_{infi}) și una dintr-un set superior (F_{supj}), unde i și j aparțin mulțimii {1, 2, 3, 4} – vezi figura 2.2. Fiecare ton DTMF are o durată aproximativă de 100 ms. Spre deosebire de metoda de apelare prin impulsuri, sistemul DTMF permite transmiterea nu doar a celor 10 cifre (0–9), ci și a unor simboluri suplimentare, cele mai comune fiind „*” și „#”, disponibile pe majoritatea telefoanelor electronice moderne.

Frecvențele utilizate pentru generarea tonurilor DTMF se încadrează în intervalul 700 – 1700 Hz, care corespunde aproximativ benzii inferioare a spectrului telefonic. Această bandă este, prin convenție, rezervată pentru transmisia datelor sau a semnalelor de apel provenite de la un modem sau un echipament terminal, în contextul unei transmisiuni de tip FDD (Frequency Division Duplexing), adică duplex integral pe linia telefonică prin separarea frecvențelor de transmisie și recepție.

Distorsiunile de neliniaritate prezente pe linia telefonică pot genera componente armonice nedorite în semnalul DTMF, ceea ce poate duce la erori în procesul de decodare a tonurilor – vezi exemplul de mai jos. Pentru a preveni aceste efecte, frecvențele DTMF sunt alese astfel încât să nu existe relații armonice între frecvențele din setul inferior (F_{inf}) și cele din setul superior (F_{sup}), conform figurii 2.2.

În plus, pentru a minimiza influența acestor distorsiuni, tonurile din setul inferior – care au un potențial mai mare de a genera armonici care se suprapun cu frecvențele din setul superior – sunt emise cu o amplitudine mai redusă, conform specificațiilor din figura 2.3. Această diferență de amplitudine contribuie la o protecție sporită împotriva efectelor negative ale neliniarităților.

$F_{inf}[Hz]$ \ $F_{sup}[Hz]$	1209	1339	1447	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

Figura 2.2: Frecvențele tonurilor DTMF [1]

	$A_{min}[mV]$	$A_{nominal}[mV]$	$A_{max}[mV]$
F_{sup}	160	190	200
F_{inf}	130	150	160

Figura 2.3: Amplitudinile tonurilor DTMF [1]

2.2 Implementare tehnici de semnalizare în Matlab

2.2.1 Implementare semnalizare prin impuls

```
fe=1000; % frecvența de eșantionare
TM=0.033; % timp make (s)
TB=0.066; % timp break (s)
TI=0.4; % timp interdigit (s)
SM = zeros(1,floor(TM*fe)); % o perioadă semnnal make
SB= ones(1,floor(TB*fe));% o perioadă semnal break
SI=zeros(1,floor(TI*fe));%o perioadă semnal interdigit
sBM=[SB SM]; % o perioada break+make
sCapat = [SB SI]; % terminație simbol (break+interdigit)
digit=0; % am ales cifra 0

semnal=SI; % incepem cu perioada semnal interdigit
if digit ==0
    digit=10; % cifra 0, deci avem nevoie de 10 întreruperi break
end

semnal = [semnal repmat(sBM, 1, digit-1)]; % vom avea la inceput interdigit + 9*(break+make) perioade
semnal = [semnal sCapat]; % la final încă o perioadă interdigit pentru a semnaliza terminația cifrei
timp = (0:length(semnal)-1)/fe; % calculăm timpul total al cifrei
plot(timp,semnal);
[S,f] = spectrum_analyzer(semnal, fe);
figure
plot(f,S);
```

Figura 2.4: Implementare semnalizare prin impuls pentru cifra 0

După cum am spus în subcapitolul 2.1.2, cifra 0 ar avea nevoie de 10 impulsuri break. După fiecare astfel de impuls urmează un alt impuls make, pentru a putea fi diferențiate între ele. La final, se va adăuga o perioadă break+interdigit, pentru a semnaliza terminația cifrei.

Funcția repmat(sBM, 1, digit-1) va repeta perioada break+make de digit-1 ori, adică de 9 ori, după care la semnal va fi concatenat sCapat, care reprezintă o perioadă break+interdigit, rezultând astfel 10 impulsuri break pentru cifra 0, așa cum spune și teoria.

Analiza în domeniul timp:

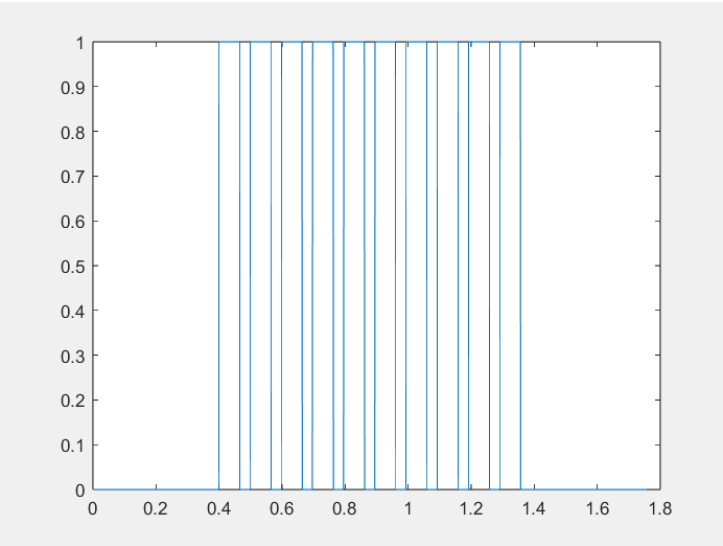

Figura 2.5: Analiză în domeniul timp semnalizare prin impuls pentru cifra 0

În figura de mai sus, observăm că există o perioadă interdigit la început de 400ms, urmată de 10 impulsuri BREAK, fiecare dintre acestea urmată la rândul ei de o perioadă MAKE, pentru diferențiere. La final a fost adăugată încă o perioadă interdigit, pentru a semnaliza posibilitatea apariției unei alte cifre.

Analiza în domeniul frecvență:

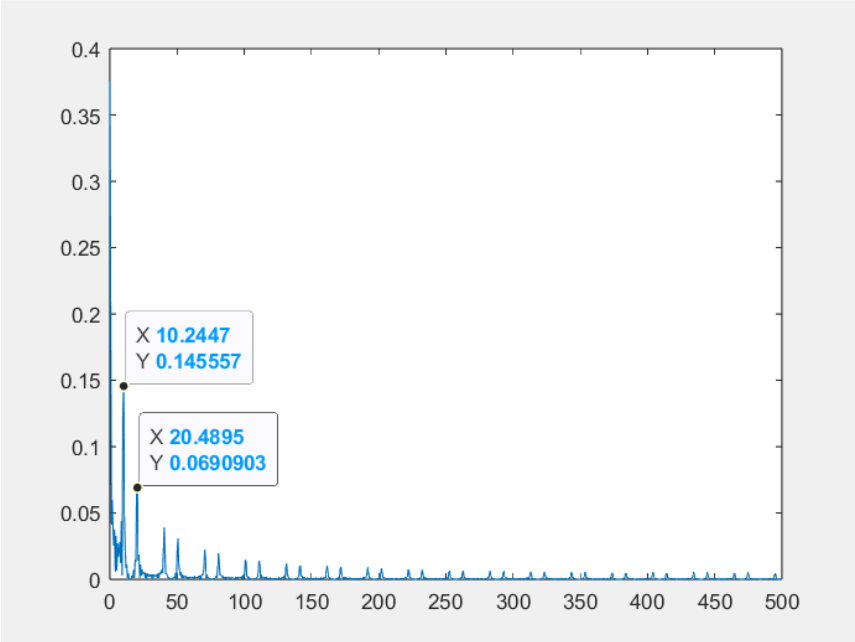

Figura 2.6: Analiză în domeniul frecvență semnalizare prin impuls pentru cifra 0

Analizorul spectral evidențiază un spectru caracteristic semnalizării prin impulsuri pentru cifra 0, constând într-o secvență de 10 întreruperi ale curentului de buclă. În spectru se observă componente semnificative la multiplii frecvenței fundamentale de aproximativ 10 Hz, corespunzătoare structurii de semnal tip pătrat (break + make). Amplitudinea armonicilor scade progresiv, iar spectrul este concentrat în zona joasă de frecvențe, ceea ce reflectă natura lentă și discretă a semnalului generat. Componentele de frecvență mai ridicată sunt neglijabile și rezultă din caracterul finit (neinfinit) al semnalului analizat.

2.2.2 Implementare semnalizare prin tonuri DTMF

```
fe = 6000; % frecv eșantionare
A1 = 0.2; % amplitudine fi
A2 = 0.3; % amplitudine fs
fi = [697 770 852 941]; %frecv inferioare din tabel teams
fs = [1209 1339 1447 1633]; % frecv superioare

Tton = 0.09; % perioada ton
Tpauza = 0.01; % per pauza între 2 cifre

spause = zeros(1, floor(Tpauza*fe)); % semnal de pauză
t =  0 :1/fe : Tton - 1/fe; % perioadă ton
finf = fi(1); % 697 Hz
fsup = fs(2); % 1339 Hz
% rezultă cifra 2

signal_tone = A1*sin(2*pi*finf*t) + A2*sin(2*pi*fsup*t); % tonul format din 2 sinusuri pt cifra 2
signal = [signal_tone spause]; % tonul + semnalul de pauză

time=(0:length(signal) - 1/fe); % timpul total

figure;
plot(time, signal);

[Signal, f] = spectrum_analyzer(signal, fe);
figure;
plot(f, Signal);

sound(signal, fe);
```

Figura 2.7: Implementare ton DTMF pentru cifra 2

Pentru generearea acestui ton, am folosit frecvențele din tabelul din figura 2.2, și am ales pentru frecvența inferioară valoarea de 697 Hz, iar pentru cea superioară 1339 Hz, rezultând astfel cifra 2. Am construit semnalele sinusoidale cu aceste frecvențe iar apoi le-am adunat, pentru a forma tonul DTMF. La acest ton am concatenat și un semnal de pauză, pentru a semnaliza terminația cifrei. Am recalculat timpul total pentru acest ton, am plotat în timp și frecvență și l-am trimis la placa de sunet.

Analiza în domeniul timp:

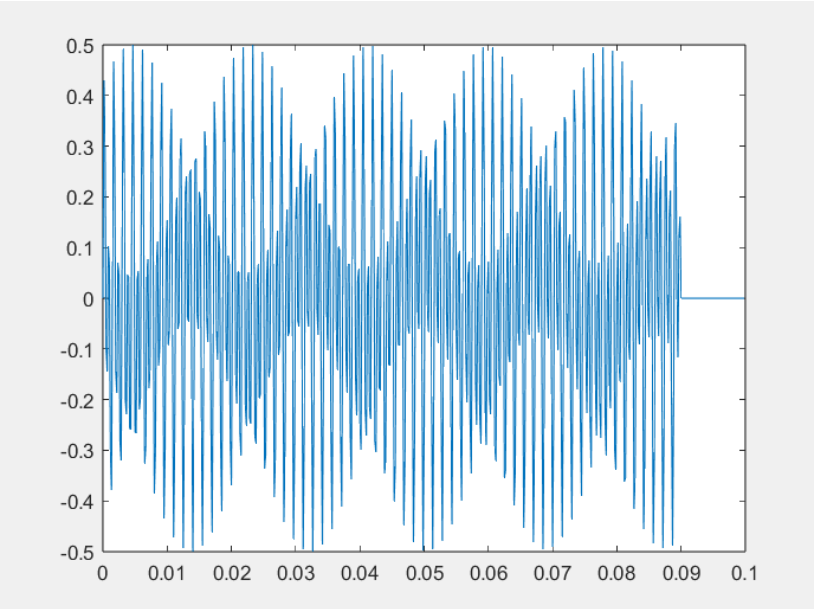

Figura 2.8: Analiză în domeniul timp ton DTMF pentru cifra 2

În figura de mai sus, pe axa orizontală avem timpul exprimat în secunde, iar pe axa verticală amplitudinea, exprimată în volți. Se poate observa cum tonul are o perioadă de 0.9 secunde, iar pauza o perioadă de 0.01 secunde.

Analiza în domeniul frecvență:

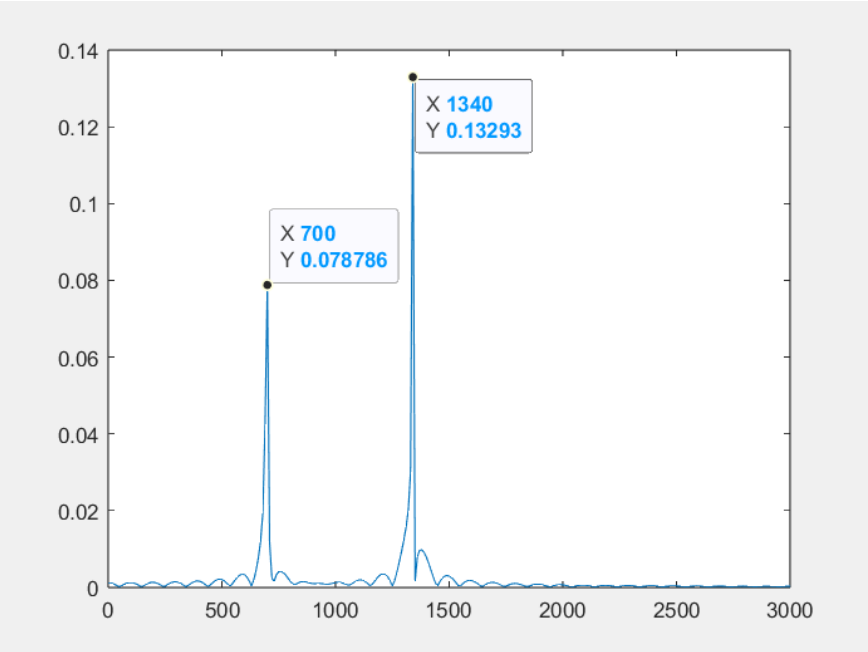

Figura 2.9: Analiză în domeniul frecvență ton DTMF pentru cifra 2

În acest analizor spectral, putem observa cele 2 armonici pe care le-am ales în cod pentru cifra 2. Acestea au aproximativ aceleași frecvențe, abaterile fiind de la modul în care se aproximează funcția sinus și operația de adunare între cele 2 semnale.

2.3 Implementare interfață grafică pentru semnalizare de adresă

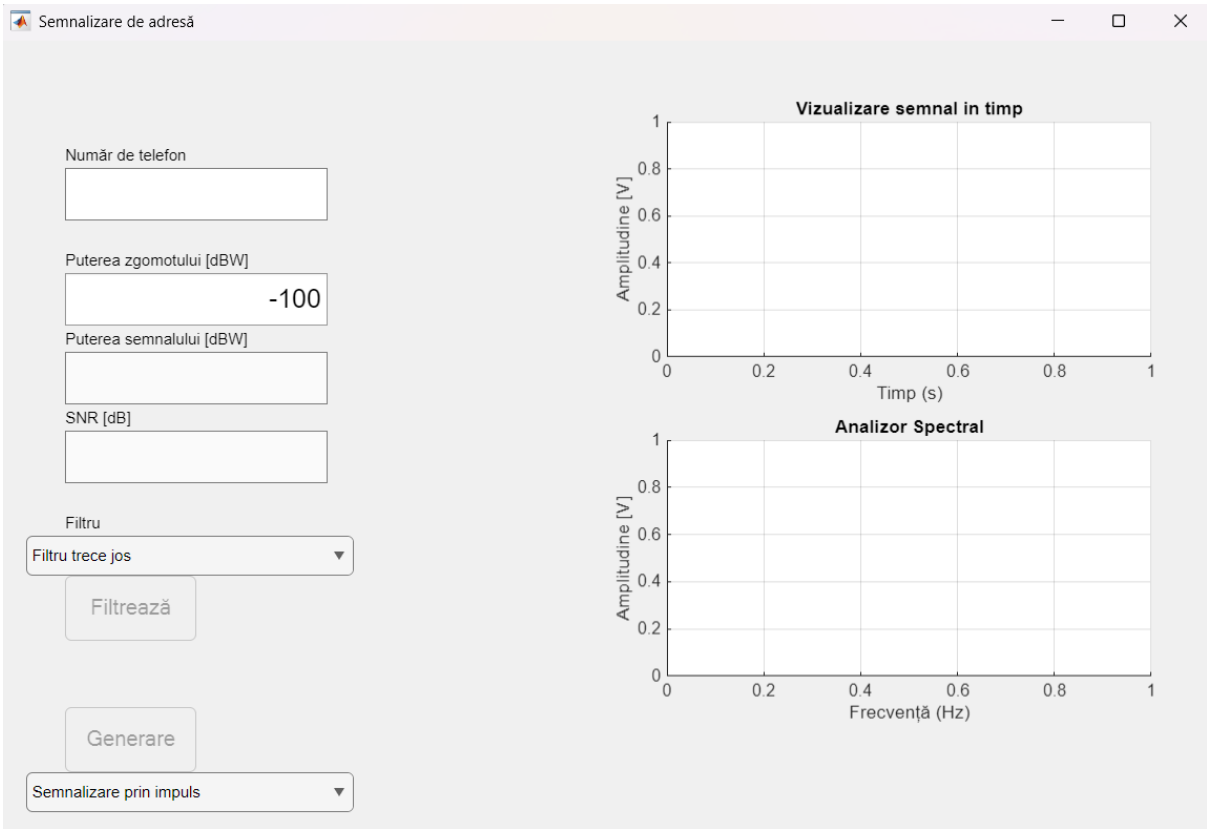

Figura 2.10: Interfață grafică pentru semnalizare de adresă

În această interfață au fost implementate următoarele componente:

- Un câmp pentru introducerea unui număr de telefon
- Un câmp din care se poate selecta una dintre metodele de semnalizare de adresă
- Vizualizarea semnalului în domeniul timp și frecvență
- Trimiterea tonurilor DTMF la placa de sunet, cu ajutorul funcției sound
- Un buton pentru generarea semnalizării selectate
- Un câmp pentru adăugarea zgomotului la semnalul generat
- Două câmpuri pentru calcularea puterii semnalului și a SNR-ului
- Un câmp pentru selectarea tipului de filtru și filtrarea semnalului + vizualizare caracteristică filtru

Codul prin care am implementat această interfață:

```
function GUI_semnalizare

fig_semnalizare = uifigure('Name', 'Semnalizare de adresă', ...
    'units', 'normalized', ...
    'position', [0.1 0.1 0.6 0.7]);

% figură pentru reprezentare semnal în timp
ax_signal = axes('Parent', fig_semnalizare, 'Position', [0.55, 0.6, 0.4, 0.3]);
xlabel(ax_signal, 'Timp (s)');
ylabel(ax_signal, 'Amplitudine [V]');
title(ax_signal, 'Vizualizare semnal in timp');
grid(ax_signal, 'on');

% figură pentru reprezentare semnal în frecvență
ax_spectrum = axes('Parent', fig_semnalizare, 'Position', [0.55, 0.2, 0.4, 0.3]);
xlabel(ax_spectrum, 'Frecvență (Hz)');
ylabel(ax_spectrum, 'Amplitudine [V]');
title(ax_spectrum, 'Analizor Spectral');
grid(ax_spectrum, 'on');

% inițializarea cheilor (cifrelor) într-un dicționar
keyDigits = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "*", "0" ,"#"];

% pentru fiecare cifră vom avea 2 frecvențe fi și fs
valueDigits = {...
    {697, 1209}, ... % 1
    {697, 1339}, ... % 2
    {697, 1477}, ... % 3
    {770, 1209}, ... % 4
    {770, 1339}, ... % 5
    {770, 1477}, ... % 6
    {852, 1209}, ... % 7
    {852, 1339}, ... % 8
    {852, 1477}, ... % 9
    {941, 1209}, ... % *
    {941, 1339}, ... % 0
    {941, 1477}, ... % #
};

% construim dicționarul
digitsDict = dictionary(keyDigits, valueDigits);

% meniu drop-down pentru selectare semnalizare
dd = uidropdown(fig_semnalizare, ...
    'Items', {'Semnalizare prin impuls', 'Semnalizare prin tonuri DTMF'}, ...
    'Position', [20, 20, 250, 30]);

% etichetă pentru câmpul de introducere a numărului de telefon
uilabel(fig_semnalizare, ...
    "Text", "Număr de telefon", ...
    "Position", [50 500 200 40], ...
    "FontSize", 12);

% câmp pentru introducerea numărului de telefon
afisaj = uieditfield(fig_semnalizare, 'text', ...
    'Editable', 'on', ...
    'Position', [50 470 200 40], ...
    'FontSize', 20, ...
    'ValueChangingFcn', @(src,event) valideaza_input_dinamic(src,event));

% etichetă pentru câmpul de introducere a nivelului de zgomot pentru
% semnalul generat
uilabel(fig_semnalizare, ...
    "Text", "Puterea zgomotului [dBW]", ...
    "Position", [50 420 200 40], ...
    "FontSize", 12);
```

```
% câmp pentru introducerea nivelului de zgomot
pZgomotField = uieditfield(fig_semnalizare, 'numeric', ...
    'Editable', 'on', ...
    'Value',-100,...
    'Position', [50 390 200 40], ...
    'FontSize', 20);

% etichetă pentru câmpul de calculare a puterii semnalului
uilabel(fig_semnalizare, ...
    "Text", "Puterea semnalului [dBW]", ...
    "Position", [50 360 200 40], ...
    "FontSize", 12);

% câmpul de calculare a puterii semnalului
pSemnalField = uieditfield(fig_semnalizare, 'text', ...
    'Editable', 'off', ...
    'Position', [50 330 200 40], ...
    'FontSize', 20);

% etichetă pentru câmpul de calculare a SNR-ului
uilabel(fig_semnalizare, ...
    "Text", "SNR [dB]", ...
    "Position", [50 300 200 40], ...
    "FontSize", 12);

% câmpul de calculare a SNR-ului
snr = uieditfield(fig_semnalizare, 'text', ...
    'Editable', 'off', ...
    'Position', [50 270 200 40], ...
    'FontSize', 20);

% etichetă pentru câmpul de selectare a filtrului
uilabel(fig_semnalizare, ...
    "Text", "Filtru", ...
    "Position", [50 220 200 40], ...
    "FontSize", 12);

% câmpul de selectare a filtrului
filters = uidropdown(fig_semnalizare, ...
    'Items', {'Filtru trece jos', 'Filtru trece sus', 'Filtru trece bandă'}, ...
    'Position', [20, 200, 250, 30]);

% buton filtrare semnal
bFiltrare = uicontrol(fig_semnalizare, ...
    'Style', 'pushbutton', ...
    'Enable', 'off', ...
    'String', 'Filtrează', ...
    'FontSize', 12, ...
    'Position', [50 150 100 50], ...
    'Callback', @(src, event) filtrareSemnal(src, ax_signal,ax_spectrum));

% buton generare semnal
bGenerare = uicontrol(fig_semnalizare, ...
    'Style', 'pushbutton', ...
    'Enable', 'off', ...
    'String', 'Generare', ...
    'FontSize', 12, ...
    'Position', [50 50 100 50], ...
    'Callback', @(src, event) updatePlot(src, ax_signal,ax_spectrum));

% structură în care salvez toate componentele de care am nevoie, butoane...
guidata(fig_semnalizare, struct('bGenerare', bGenerare,'meniuSemnalizare',dd,'campNumar',afisaj, ...
    'digitsDict',digitsDict,'pZgomot',pZgomotField,'pSemnal',pSemnalField,'snr',snr,'figSemnalizare', ...
    fig_semnalizare,'bFiltrare',bFiltrare,'filters',filters));

end
```

Figura 2.11: Implementare interfață grafică pentru semnalizare de adresă

Prima dată vom avea nevoie să creăm fereastra principală a aplicației, cu o componentă uifigure. Creăm încă 2 figuri pentru reprezentarea semnalului ales în timp și frecvență, iar după aceea construim un dicționar, pe baza căruia vom putea asigna fiecărei cifre câte 2 frecvențe pentru semnalizarea prin tonuri DTMF.

- **keyDigits** – cheile (cifrele), cu ajutorul cărora voi prelua frecvențele corespunzătoare fiecărei cifre
- **valueDigits** – pentru fiecare cheie din dictionar (adică cifra numărului), vom avea 2 frecvențe, una inferioară și una superioară, conform figurii 2.2.
- **dd** – meniu drop-down pentru selectare semnalizare, din acest meniu se pot selecta cele 2 tipuri de semnalizare de adresă.
- **afișaj** – câmp pentru introducerea numărului de telefon. În timp ce valoarea acestui câmp este modificată, se apelează funcția valideaza_input_dinamic, prin care validez câmpul respectiv.
- **pZgomotField** – câmp pentri introducerea nivelului de zgomot, în dBW. Acest nivel de zgomot va fi adăugat semnalului generat.
- **pSemnalField** – câmp pentru calcularea puterii semnalului generat.
- **snr** – câmp pentru calcularea raportului semnal-zgomot (SNR)
- **filters** – meniul drop-down, din care se poate selecta un anumit tip de filtru, mai apoi putându-se filtra semnalul generat.
- **bFiltrare** – buton cu ajutorul căruia se filtrează semnalul generat. La apăsare, se execută funcția filtrareSemnal
- **bGenerare** – buton prin care se generează semnalul, conform semnalizării alese.
- Guidata(...) – în această structură salvez toate componentele de care am nevoie, pentru a nu fi nevoit să le trimit ca și parametri la funcții

Mai departe, voi descrie succint funcțiile implementate în această interfață:

```
% funcția în care validez numărul de telefon introdus în timp real
function valideaza_input_dinamic(src, event)
    components = guidata(src); % preiau componente din structură
    val = event.Value; % preiau continutul in curs de introducere, deci și caracterul tastat.
    % cu src.Value ar fi trebuit să apăs enter pentru a prelua acea valoare

    % dacă acest câmp este gol, nu pot genera semnalizarea încă
    if ~isempty(event.Value)
        components.bGenerare.Enable = 'on';
    else
        components.bGenerare.Enable = 'off';
    end

    if isequal(components.meniuSemnalizare.Value, 'Semnalizare prin impuls')
        val_filtrat = regexprep(val, '[^0-9]', '');  % doar cifre dacă e semnalizare prin impuls
    else
        val_filtrat = regexprep(val, '[^0-9#*]', '');  % doar cifre, # și * pt ton DTMF
    end

    src.Value = val_filtrat; % filtrez valoarea câmpului
end
```

Figura 2.12: Implementare funcție pentru validarea numărului de telefon introdus

În funcția din figura de mai sus, voi valida numărul de telefon introdus în timp real, adică după fiecare tastă apăsată de către utilizator. Desigur, în momentul în care acest câmp este gol, nu pot să generez semnalizarea, deci butonul de generare va fi dezactivat, până la introducerea unei secvențe. Așa cum am explicat și la fundamentarea teoretică în subcapitolul 2.1, semnalizarea de impuls permite doar utilizarea cifrelor de la 0-9 în construcția numărului, dar semnalizarea prin tonuri DTMF permite, pe lângă aceste cifre, utilizarea simbolurilor * și #, deci câmpul va fi filtrat în corespondență cu tipul de semnalizare aleasă de către utilizator.

Funcția pentru generarea semnalului ales:

```
% funcția în care generez semnalizarea aleasă din meniul drop-down
function updatePlot(src, ax_signal,ax_spectrum)
    components = guidata(src); % preiau componente din structură

    vector_digits = double(components.campNumar.Value); % preiau textul din câmpul pt introducerea
    % numărului și îl transform într-un vector, în care fiecare element
    % devine codul ascii al caracterului din text

    pZgomot = components.pZgomot.Value; % puterea zgomotului în dBW
    switch components.meniuSemnalizare.Value % generare în funcție de semnalizarea aleasă
        case 'Semnalizare prin impuls'
            vector_digits = vector_digits - double('0'); % scad '0' (48d sau 30h) pentru a obține cifrele corecte
            fe=1000; % frecvența de eșantionare
            TM=0.033; % timp make
            TB=0.066; % timp break
            TI=0.4; % timp interdigit
            SM = zeros(1,floor(TM*fe)); %o perioada semnal make
            SB= ones(1,floor(TB*fe));% o perioada semnal break
            SI=zeros(1,floor(TI*fe));%o perioada semnal interdigit
            sBM=[SB SM]; % o perioada break+make
            sCapat = [SB SI];% semnal capat

            semnal=SI; % incepem cu perioada semnal interdigit
            for i=1:length(vector_digits)
                digit=vector_digits(i);

                if digit ==0
                    digit=10; % 10 impulsuri pt cifra 0
                end

                semnal = [semnal repmat(sBM, 1, digit-1)]; % concatenez la semnal pulsurile corespunzatoare tuturor
                % cifrelor de la 0-9. Vom avea la inceput interdigit + length(vector_digits)*(break+make) perioade
                semnal = [semnal sCapat]; % secventa de la capat
            end
            timp = (0:length(semnal)-1)/fe; % calculăm timpul total al semnalizării

            signal_power_measured = rms(semnal)^2;% puterea efectivă a semnalului generat
            components.pSemnal.Value = num2str(10*log10(signal_power_measured)); % valoarea puterii semnalului
            % în câmpul corespunzător

            noise = wgn(1, length(semnal), pZgomot);% definim vector de zgomot gaussian

            components.snr.Value = num2str(10*log10(signal_power_measured) - pZgomot); % SNR = Psemnal - Pzgomot
            semnal_cu_zgomot = semnal + noise; % adăugam zgomot gaussian peste semnal

            % dacă vreau să generez alt ton, dar deja a fost generat unul, trebuie
            % să șterg axele întâi pentru a nu suprapune plot-urile unul
            % peste altul
            cla(ax_signal);
            cla(ax_spectrum);

            % dacă nu am hold on, plot-ul de mai jos va recrea de fapt axa ca și
            % cum ar fi una nouă, deci aici salvez xlabel, ylabel, etc.
            hold(ax_signal, 'on');
            hold(ax_spectrum, 'on');

            % plot in timp si spectru
            plot(ax_signal, timp, semnal_cu_zgomot, 'b', 'LineWidth', 1.5);

            [Yfft, f] = spectrum_analyzer(semnal_cu_zgomot, fe);
            plot(ax_spectrum, f, Yfft, 'r', 'LineWidth', 1.5);

            setappdata(components.figSemnalizare, 'semnalImpuls', semnal_cu_zgomot);  % salvez semnalul în appdata
            setappdata(components.figSemnalizare, 'timpImpuls', timp);  % salvez timpul în appdata
            setappdata(components.figSemnalizare, 'feImpuls', fe);  % salvez fe în appdata
```

```
            components.bFiltrare.Enable = 'on'; % după generare se poate filtra

            % revenire la comportament inițial pentru a nu suprapune
            % plot-urile
            hold(ax_signal, 'off');
            hold(ax_spectrum, 'off');


        case 'Semnalizare prin tonuri DTMF'
            digitsDict = components.digitsDict; % preluăm dicționarul din structură
            fe = 6000; % frecv eșantionare
            A1 = 0.2; % amplitudine fi
            A2 = 0.3; % amplitudinde fs
            Tton = 0.09; % perioadă ton
            Tpauza = 0.01; % perioadă pauza

            spause = zeros(1, floor(Tpauza*fe)); % vector pt semnal de pauză
            t =  0 :1/fe : Tton - 1/fe; % timpul în care tonul e ON

            signal_all = 0;

            % dacă vreau să generez alt ton, dar deja a fost generat unul, trebuie
            % să șterg axele întâi pentru a nu suprapune plot-urile unul
            % peste altul
            cla(ax_signal);
            cla(ax_spectrum);

            % parcurg vectorul de cifre introduse
            for i=1:length(vector_digits)

                value = digitsDict(char(vector_digits(i))); % preiau codurile ascii ca și caractere
                % pt a putea accesa valorile din dicționar
                frecvente = value{1}; % cell array de valori
                fi = frecvente{1}; % frecv inferioară pe poz 1 din cell array
                fs = frecvente{2}; % frecv superioară pe poz 2 din cell array
                dtmf_ton = A1*sin(2*pi*fi*t) + A2*sin(2*pi*fs*t); % 1 ton dtmf
                signal = [dtmf_ton spause]; % concatenez cu semnal de pauză

                noise = wgn(1, length(signal), pZgomot);% definim vector de zgomot gaussian pt fiecare ton dtmf
                semnal_cu_zg = signal + noise; % adaug zgomot peste fiecare ton in parte
                signal_all = [signal_all semnal_cu_zg]; % concatenez toate tonurile dtmf

                % dacă nu am hold on, plot-ul de mai jos va recrea de fapt axa ca și
                % cum ar fi una nouă, deci aici salvez xlabel, ylabel, etc.
                hold(ax_signal, 'on');
                hold(ax_spectrum, 'on');

                sound(semnal_cu_zg,fe) % trimit la placa de sunet
                pause(Tton+ Tpauza); % delay intre tonuri
            end

            time=(0:length(signal_all) - 1/fe); % timpul total

            signal_power_measured = rms(signal_all)^2;% puterea efectiva a semnalului generat
            components.pSemnal.Value = num2str(10*log10(signal_power_measured)); % valoarea puterii semnalului
            % în câmpul corespunzător

            components.snr.Value = num2str(10*log10(signal_power_measured) - pZgomot); % SNR = Psemnal - Pzgomot

            % plot în timp si spectru
            plot(ax_signal, time, signal_all, 'b', 'LineWidth', 1.5);

            [Yfft, f] = spectrum_analyzer(signal_all, fe);
            plot(ax spectrum, f, Yfft, 'r', 'LineWidth', 1.5);
            setappdata(components.figSemnalizare, 'semnalDTMF', signal_all);  % salvez semnalul în appdata
            setappdata(components.figSemnalizare, 'timpDTMF', time);  % salvez timpul în appdata
            setappdata(components.figSemnalizare, 'feDTMF', fe);  % salvez fe în appdata

            components.bFiltrare.Enable = 'on'; % după generare se poate filtra

            % revenire la comportament inițial pentru a nu suprapune
            % plot-urile
            hold(ax_signal, 'off');
            hold(ax_spectrum, 'off');
    end
end
```

Figura 2.13: Implementare funcție pentru generarea semnalizării selectate

Această funcție va genera semnalizarea aleasă din meniul drop-down. Prima dată, se va prelua valoarea câmpului pentru introducerea numărului de telefon. Acest câmp este de tip text, iar prin aplicarea funcției double() asupra valorii lui, o vom transforma într-un vector, în care fiecare element este codul ascii al cifrei/simbolului introdus în câmp.

- Pentru semnalizarea prin impuls: să presupunem că avem introdus textul „123” în câmpul respectiv. Funcția double va genera un vector de forma [49 50 51], în care fiecare element este codul ascii al fiecărui caracter introdus (49 – caracter `1`; 50 – caracter `2`; 51 – caracter `3`;). Mai departe, pentru a converti aceste caractere în cifre, vom scădea din fiecare element din vector valoarea `0` (cod ascii 48), astfel vom avea cifrele 1, 2, 3, putând astfel accesa mai jos în cod elementele din vector ca și cifre și să generăm numărul de impulsuri corespunzător fiecărei cifre introduse.

- Pentru semnalizarea prin tonuri DTMF: dicționarul de frecvențe conține cheile fiecărei cifre și simboluri *, # reprezentate ca și caractere. Pentru a putea prelua aceste valori din dicționar, trebuie folosită funcția char() asupra vector_digits, chiar dacă acesta conține codurile ascii ale caracterelor introduse în câmpul numărului de telefon. În Matlab nu se poate face conversia direct de la cod ascii la caracter, de aceea trebuie folosită această funcție. Mai departe, pentru fiecare cheie voi avea un cell array, iar înăuntrul acelui array se vor regăsi frecvențele superioare și inferioare, corespunzătoare cifrei (cheii) folosite.

Modul prin care se adaugă zgomot semnalului generat și calcularea puterii semnalului, respectiv a raportului semnal-zgomot este detaliat în codul din figura 2.13.

Funcție pentru filtrarea semnalului generat:

```
% funcția pentru filtrarea semnalului generat
function filtrareSemnal(src, ax_signal,ax_spectrum)
    components = guidata(src); % preiau componentele din structură

    switch components.meniuSemnalizare.Value % logică diferită în funcție de tipul de semnalizare

        case 'Semnalizare prin impuls'

            semnal = getappdata(components.figSemnalizare,'semnalImpuls'); % preiau semnalul din appdata
            timp = getappdata(components.figSemnalizare,'timpImpuls'); % preiau timpul din appdata
            fe = getappdata(components.figSemnalizare,'feImpuls'); % preiau fe din appdata

            switch components.filters.Value % logică diferită pentru fiecare filtrare

                case 'Filtru trece jos'
                    f_pass = 300; % frecvența de tăiere;
                    [semnal_filtrat, digital_filter_obj] = lowpass(semnal, f_pass, fe, 'Steepness', 0.99);

                case 'Filtru trece sus'
                    f_pass = 10; % frecvența de tăiere;
                    [semnal_filtrat, digital_filter_obj] = highpass(semnal, f_pass, fe, 'Steepness', 0.99);

                case 'Filtru trece bandă'
                    f_pass = [10 300]; % frecvențele de tăiere low și high;
                    [semnal_filtrat, digital_filter_obj] = bandpass(semnal, f_pass, fe, 'Steepness', 0.99);
            end

            % dacă vreau să generez alt ton, dar deja a fost generat unul, trebuie
            % să șterg axele întâi pentru a nu suprapune plot-urile unul
            % peste altul
            cla(ax_signal);
            cla(ax_spectrum);
            % dacă nu am hold on, plot-ul de mai jos va recrea de fapt axa ca și
            % cum ar fi una nouă, deci aici salvez xlabel, ylabel, etc.
            hold(ax_signal, 'on');
            hold(ax_spectrum, 'on');

            % plot în timp și spectru
            plot(ax_signal, timp, semnal_filtrat, 'b', 'LineWidth', 1.5);

            [Yfft, f] = spectrum_analyzer(semnal_filtrat, fe);
            plot(ax_spectrum, f, Yfft, 'r', 'LineWidth', 1.5);

            % revenire la comportament inițial pentru a nu suprapune
            % plot-urile
            hold(ax_signal, 'off');
            hold(ax_spectrum, 'off');
            fvtool(digital_filter_obj); % caracteristica filtrului

        case 'Semnalizare prin tonuri DTMF'
            semnal = getappdata(components.figSemnalizare,'semnalDTMF'); % preiau semnalul din appdata
            timp = getappdata(components.figSemnalizare,'timpDTMF'); % preiau timpul din appdata
            fe = getappdata(components.figSemnalizare,'feDTMF'); % preiau fe din appdata

            switch components.filters.Value

                case 'Filtru trece jos'
                    f_pass = 1500; % frecvența de tăiere;
                    [semnal_filtrat, digital_filter_obj] = lowpass(semnal, f_pass, fe, 'Steepness', 0.99);

                case 'Filtru trece sus'
                    disp('aici')
                    f_pass = 100; % frecvența de tăiere;
                    [semnal_filtrat, digital_filter_obj] = highpass(semnal, f_pass, fe, 'Steepness', 0.99);
```

```
            case 'Filtru trece bandă'
                disp('aici')
                f_pass = [400 1900]; % frecvențele de tăiere low și high;
                [semnal_filtrat, digital_filter_obj] = bandpass(semnal, f_pass, fe, 'Steepness', 0.99);
        end

        % dacă vreau să generez alt ton, dar deja a fost generat unul, trebuie
        % să șterg axele întâi pentru a nu suprapune plot-urile unul
        % peste altul
        cla(ax_signal);
        cla(ax_spectrum);

        % dacă nu am hold on, plot-ul de mai jos va recrea de fapt axa ca și
        % cum ar fi una nouă, deci aici salvez xlabel, ylabel, etc.
        hold(ax_signal, 'on');
        hold(ax_spectrum, 'on');

        % plot in timp si spectru
        plot(ax_signal, timp, semnal_filtrat, 'b', 'LineWidth', 1.5);

        [Yfft, f] = spectrum_analyzer(semnal_filtrat, fe);
        plot(ax_spectrum, f, Yfft, 'r', 'LineWidth', 1.5);

        % revenire la comportament inițial pentru a nu suprapune
        % plot-urile
        hold(ax_signal, 'off');
        hold(ax_spectrum, 'off');
        fvtool(digital_filter_obj); % caracteristica filtrului
    end

end
```

Figura 2.14: Implementare funcție pentru filtrarea semnalului generat

Pentru ambele tipuri de semnalizări preiau semnalul, timpul și frecvența de eșantionare salvate în appdata, în funcția pentru generare a semnalizării selectate. Apoi, filtrez semnalul generat în funcție de tipul de filtru ales, cu frecvențele de tăiere corespunzătoare. După aceea, plotez în timp și in frecvență semnalul filtrat, iar cu ajutorul funcției fvtool și a obiectului digital_filter_obj afișez caracteristica filtrului.

2.4 Interpretare rezultate experimentale

2.4.1 Interpretare rezultate semnalizare prin impuls

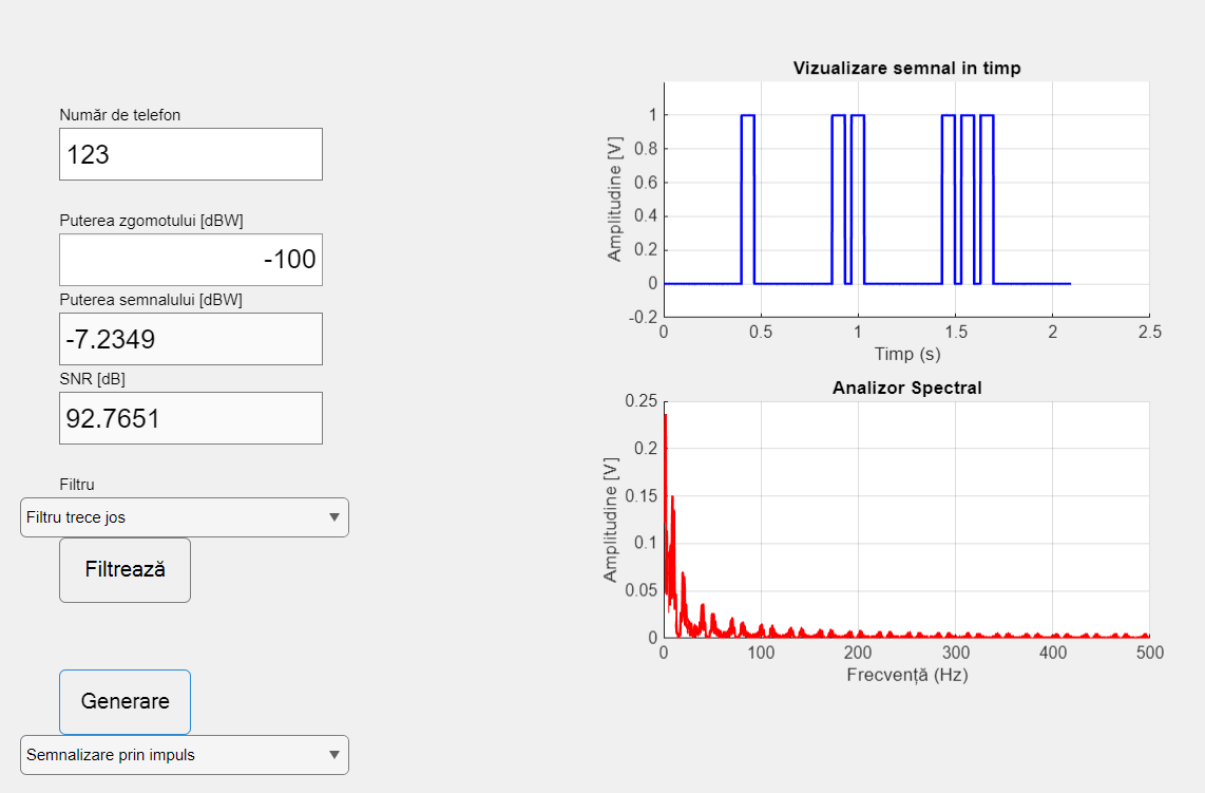

Figura 2.15: Interpretare rezultate semnalizare prin impuls pentru numărul 123

Pentru numărul de telefon `123`, și un raport semnal zgomot de 92dB (puterea semnalului generat este de aproximativ 10^9 ori mai mare decât puterea zgomotului), putem observa un caz ideal al semnalizării prin impuls, în care semnalul generat nu este afectat deloc de zgomot.

Analizorul în timp al semnalului ne indică faptul că fiecare cifră e reprezentată de un număr de impulsuri break, urmată de o perioadă interdigit, pentru diferențierea cifrelor între ele. De exemplu, pentru cifra 1 avem un impuls break, pentru cifra 2 două impulsuri break, șamd.

Spectrul este concentrat în zona joasă de frecvențe, ceea ce reflectă natura lentă și discretă a semnalului generat.

Să adăugăm zgomot gaussian peste semnalul generat:

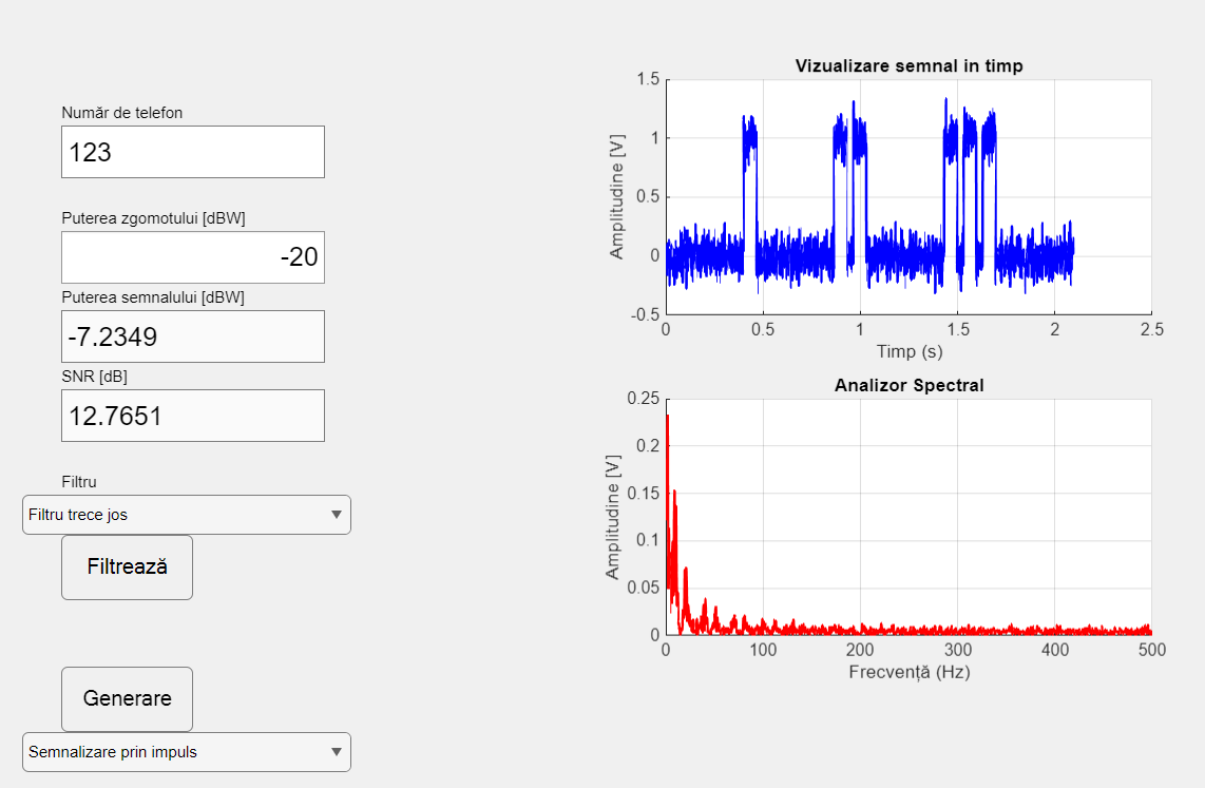

Figura 2.16: Interpretare rezultate semnalizare prin impuls pentru numărul 123, puterea zgomotului = -20 dBW

Evident, după adăugarea unui zgomot de putere -20 dBW peste un semnal de putere -7 dBW, raportul semnal zgomot s-a înrăutățit, iar acest lucru se poate observa și în analizorul în timp, semnalul acum este afectat de zgomot gaussian. Analizorul spectral prezintă componente de amplitudini moderate la frecvențe mai mari de ~100Hz, fapt care denotă prezența zgomotului.

Dacă filtrăm trece jos semnalul cu frecvența de tăiere la 100Hz (modificată în cod) obținem:

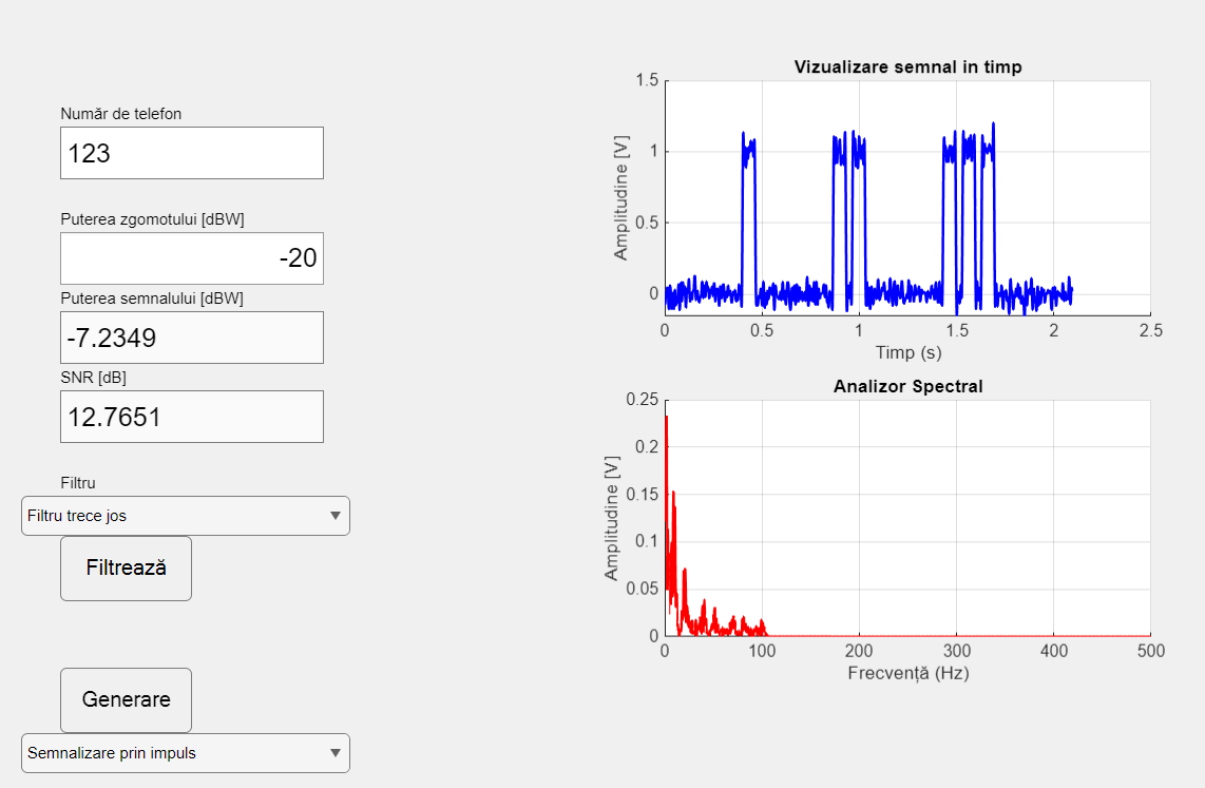

Figura 2.17: Interpretare rezultate semnalizare prin impuls pentru numărul 123, puterea zgomotului = -20 dBW, FTJ, frecvența de tăiere 100Hz

Se observă o îmbunătățire a calității semnalului după filtrare, în domeniul timp și în domeniul frecvență. Zgomotul nu mai este atât de proeminent, deoarece componentele de frecvență mai mare de 100Hz au fost atenuate complet.

Caracteristica filtrului trece jos:

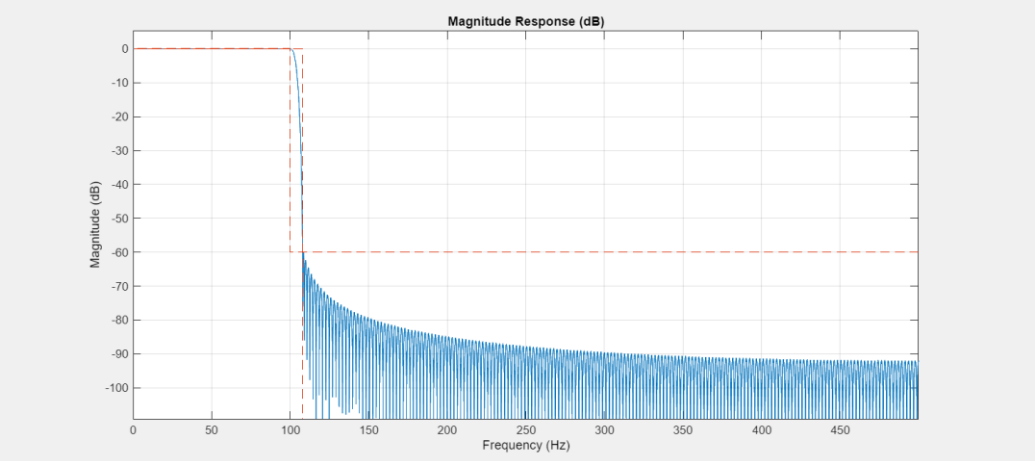

Figura 2.18: Interpretare rezultate semnalizare prin impuls pentru numărul 123, caracteristică FTJ

În figura de mai sus, se observă că frecvență de tăiere a filtrului este la 100Hz, ceea ce înseamnă că acesta va rejecta toate componentele spectrale cu frecvență mai mare decât aceasta, eliminând astfel o bună parte a zgomotului gaussian aplicat semnalului generat.

Să aplicăm și filtru trece sus asupra semnalului afectat de zgomot:

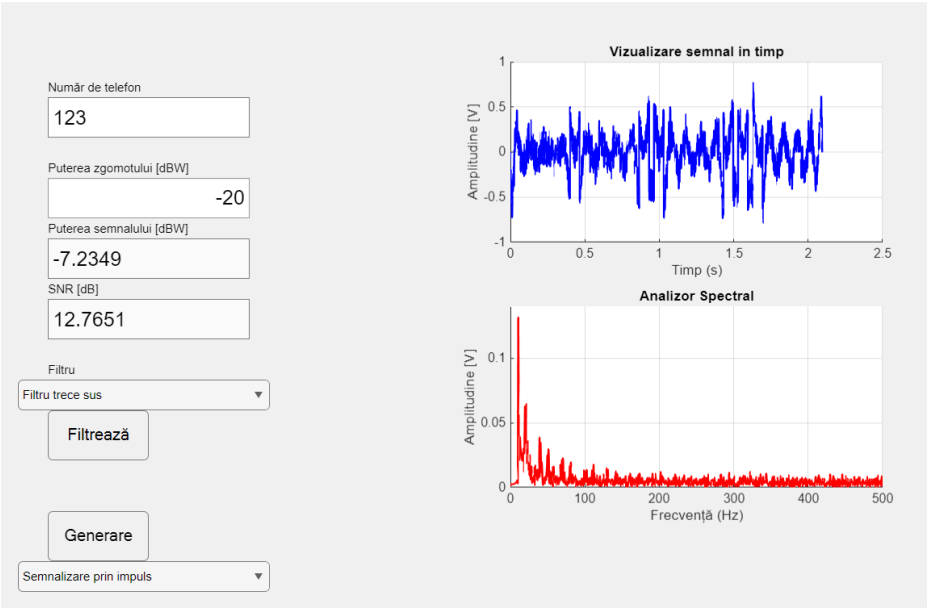

Figura 2.19: Interpretare rezultate semnalizare prin impuls pentru numărul 123, puterea zgomotului = -20 dBW, FTS, frecvența de tăiere 10Hz

Filtrând acest semnal afectat de zgomot cu un FTS la frecvența de tăiere de 10Hz, am înrăutățit de fapt calitatea semnalului, deoarece o mare parte a zgomotului este situat la frecvențe mai mari decât 10Hz. Acest lucru se observă și din analizorul spectral, avem componente la frecvențe de 100Hz, 200Hz, etc.

Caracteristica filtrului trece sus:

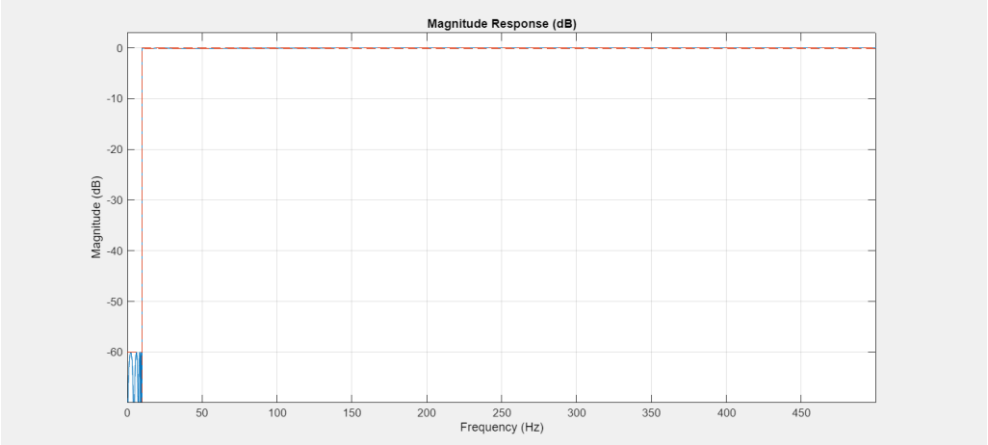

Figura 2.20: Interpretare rezultate semnalizare prin impuls pentru numărul 123, caracteristică FTS

În figura de mai sus, se observă că frecvență de tăiere a filtrului trece sus este la 10Hz, ceea ce înseamnă că acesta va rejecta toate componentele spectrale cu frecvență mai mică decât aceasta, eliminând o bună parte a semnalului util, lăsând astfel să treacă zgomotul situat la frecvențe de 100Hz, 200hz, etc.

Să aplicăm și filtru trece bandă asupra semnalului afectat de zgomot:

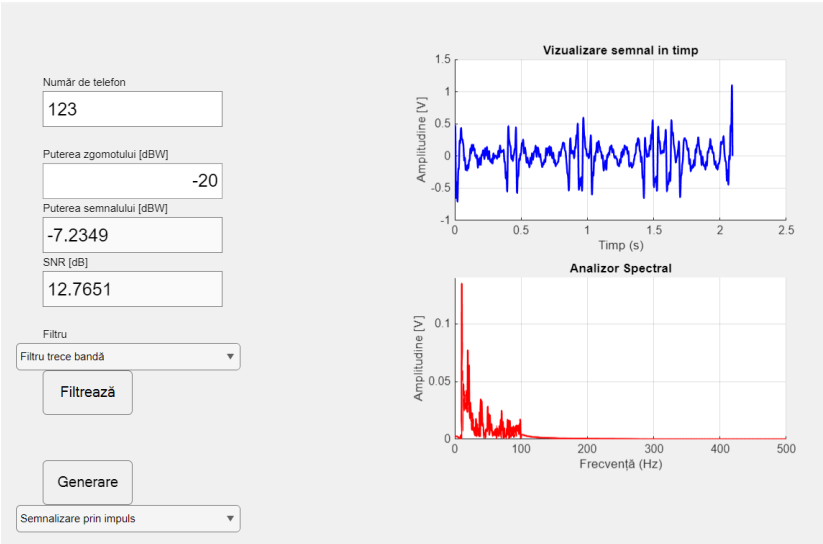

Figura 2.21: Interpretare rezultate semnalizare prin impuls pentru numărul 123, puterea zgomotului = -20 dBW, FTB, frecvențele de tăiere 10-100Hz

Filtrând acest semnal afectat de zgomot cu un FTB la frecvențele de tăiere de 10Hz (inferioară), respectiv 100Hz (superioară), avem o calitate mai bună a semnalului decât în cazul filtrării trece sus, dar semnalul tot este afectat de o cantitate semnificativă de zgomot. Se pot distinge impulsurile pentru fiecare cifră în parte, dar decodarea la recepție ar fi destul de anevoioasă.

Caracteristica filtrului trece bandă:

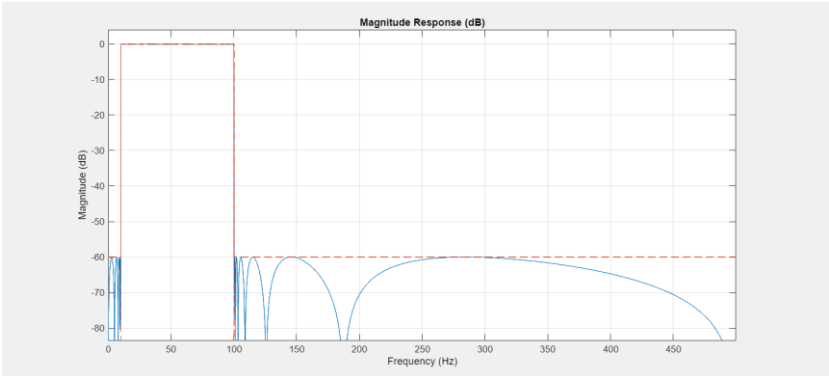

Figura 2.22: Interpretare rezultate semnalizare prin impuls pentru numărul 123, caracteristică FTS

În figura 2.22, filtrul trece bandă selectează banda semnalului între 10 și 100Hz, atenuând componentele care nu fac parte din acest interval.

2.4.2 Interpretare rezultate semnalizare prin tonuri DTMF

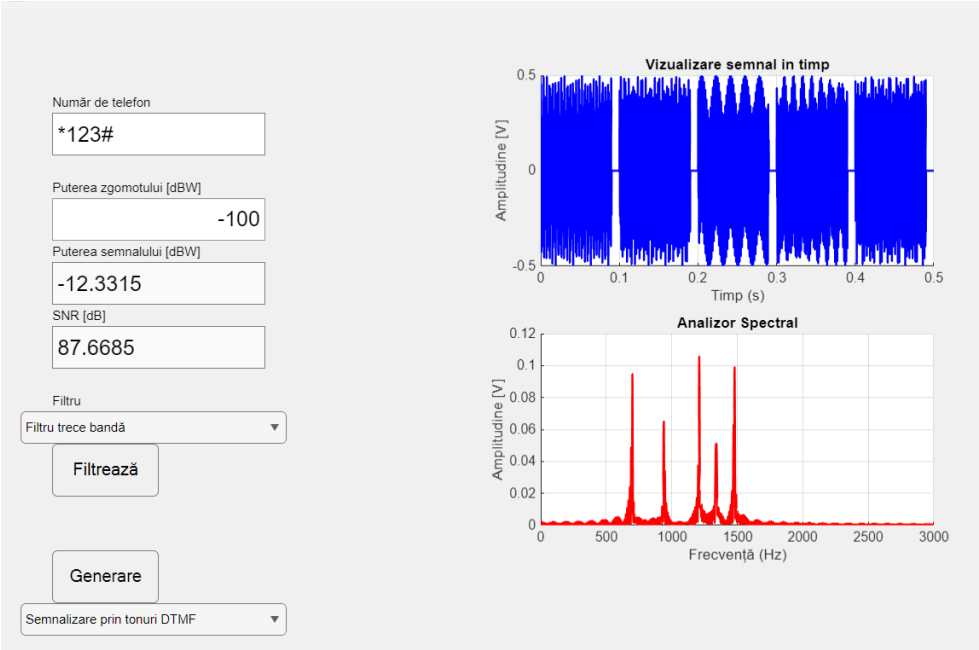

Figura 2.23: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#

Pentru numărul de telefon `*123#`, puterea zgomotului setată la -100 dBW și un raport semnal zgomot de 87dB (puterea semnalului generat este de aproximativ 10^9 ori mai mare decât puterea zgomotului), putem observa un caz ideal al semnalizării prin tonuri DTMF, în care semnalul generat nu este afectat deloc de zgomot.

Analizorul în timp al semnalului ne indică faptul că fiecare cifră e reprezentată de un ton DTMF, format din 2 semnale sinusoidale, iar timpul total al unui astfel de ton este setat în cod la 0.09 secunde (90ms). Fiecare astfel de ton este urmat de o perioadă de pauză de 0.01 secunde, pentru diferențierea între cifrele numărului de telefon.

Spectrul este concentrat în banda inferioară a spectrului telefonic (700 – 1700 Hz). Această bandă este, prin convenție, rezervată pentru transmisia datelor sau a semnalelor de apel provenite de la un modem sau un echipament terminal.

Să adăugăm zgomot gaussian peste semnalul generat:

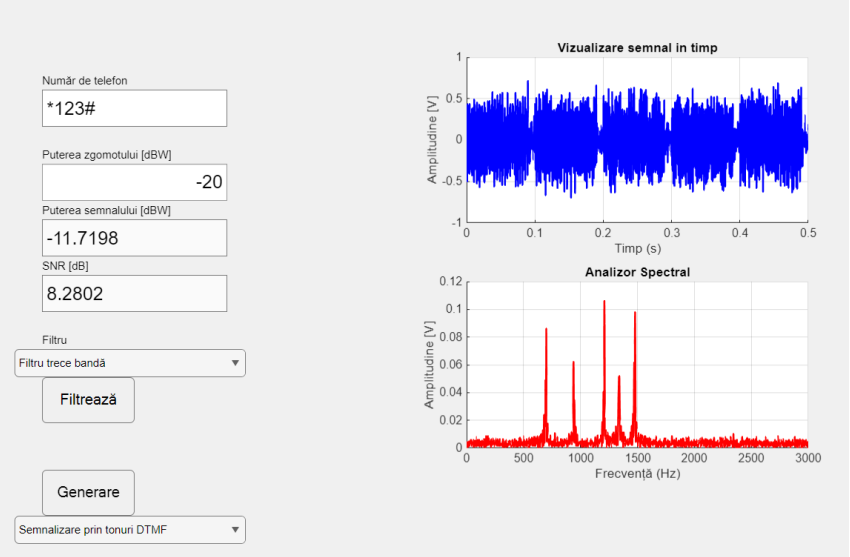

Figura 2.24: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#, puterea zgomotului = -20 dBW

Observăm că odată ce am adăugat zgomot peste semnalul generat, SNR-ul s-a înrăutățit, acest lucru se poate observa și din spectrul semnalului, afectat de zgomot gaussian.

Să filtrăm trece jos acest semnal:

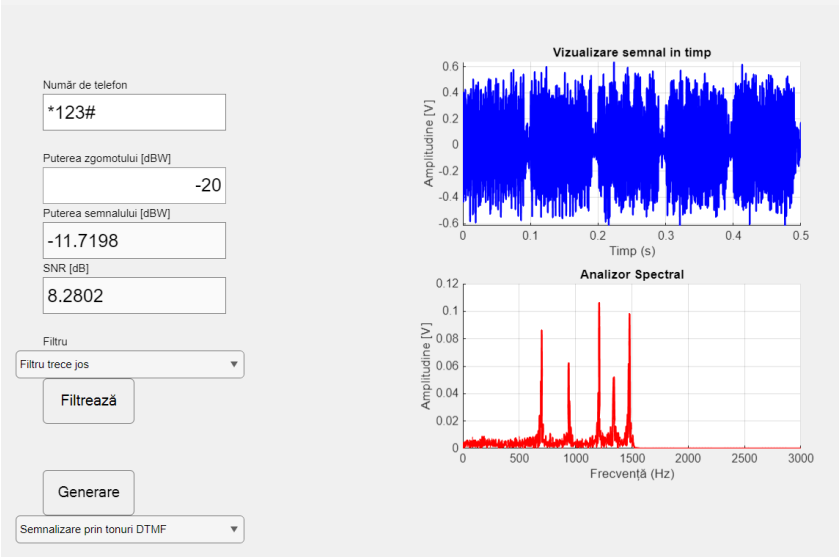

Figura 2.25: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#, puterea zgomotului = -20 dBW, FTJ frecvența de tăiere 1.5 kHz

Filtrul trece jos axat pe frecvența de 1.5kHz va elimina o parte din zgomotul de pe canal, dar semnalul este în continuare distorsionat.

Caracteristica filtrului trece jos:

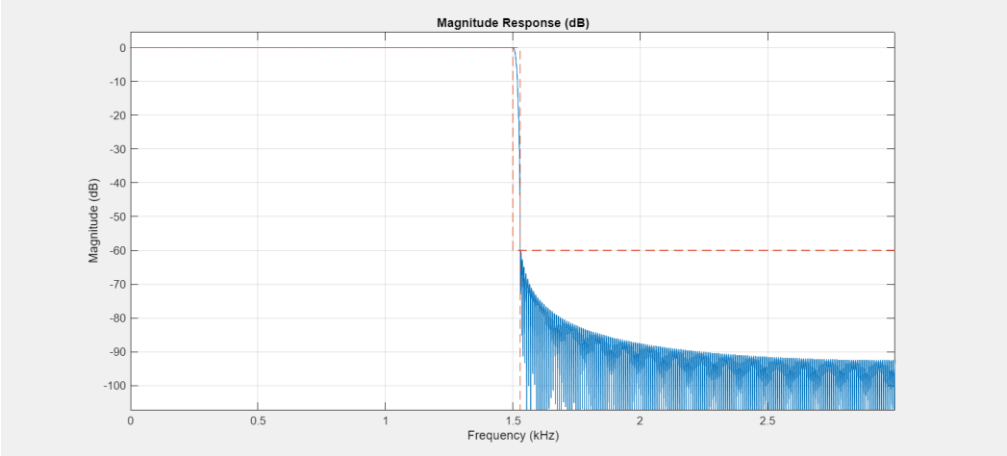

Figura 2.26: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#, caracteristică FTJ

Caracteristica filtrului trece jos arată faptul că acesta elimină componentele de frecvență mai mare decât 1.5 kHz.

Să filtrăm și trece sus semnalul afectat de zgomot:

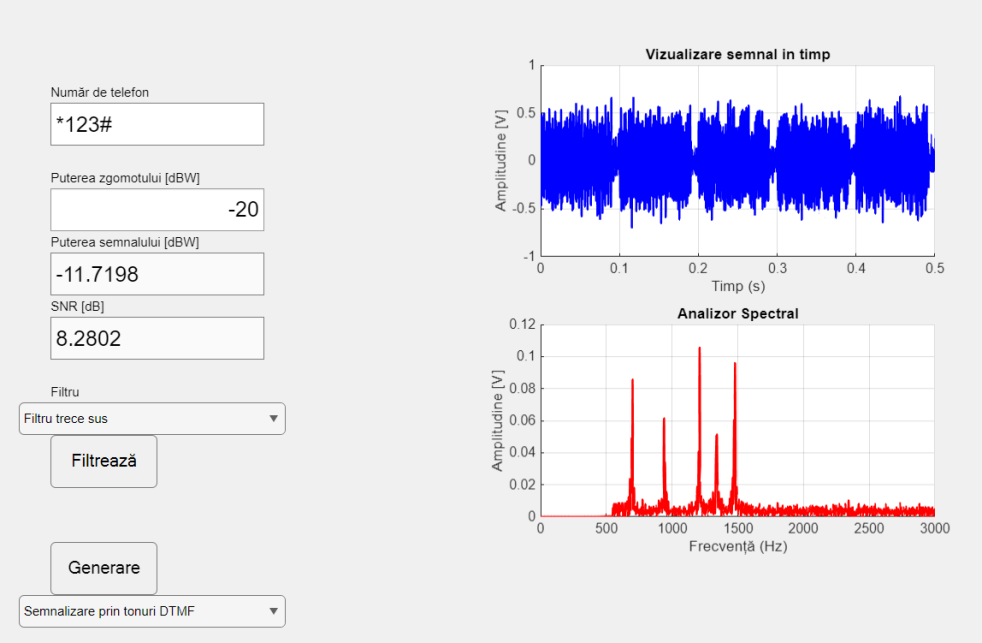

Figura 2.27: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#, puterea zgomotului = -20 dBW, FTS frecvența de tăiere 550 Hz

Semnalul fiind filtrat trece sus la frecvența de 550Hz a devenit mai calitativ, componentele de frecvență joasă fiind atenuate puternic (zgomotul).

Caracteristica filtrului trece sus:

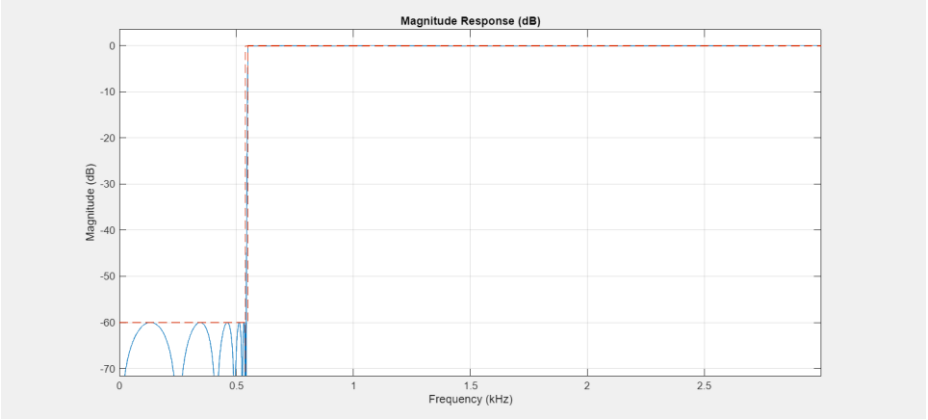

Figura 2.28: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#, caracteristică FTS

Caracteristica filtrului trece sus arată faptul că acesta elimină componentele de frecvență mai mici decât 550 Hz.

Să filtrăm și trece bandă semnalul afectat de zgomot:

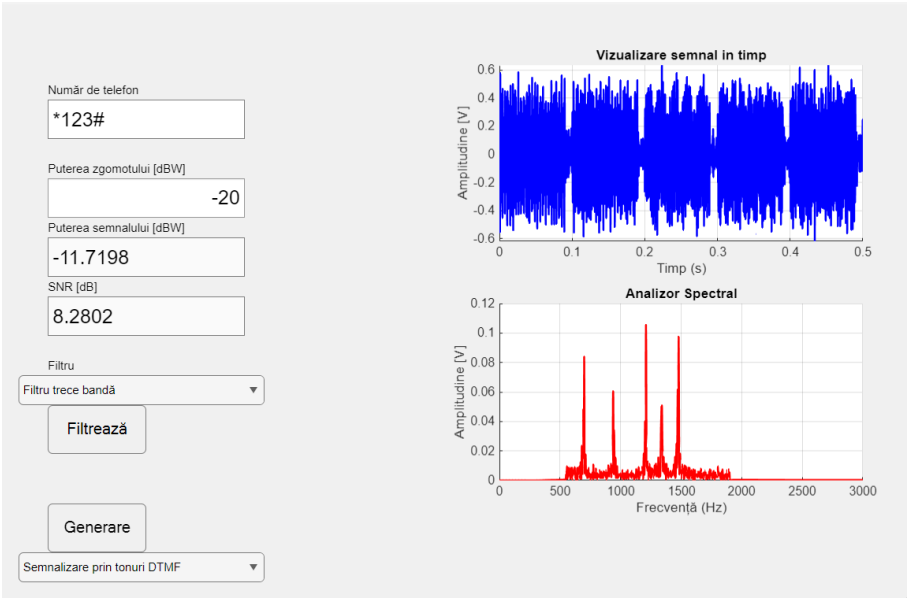

Figura 2.29: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#, puterea zgomotului = -20 dBW, FTB frecvențele de tăiere 550 - 1900 Hz

Filtrând trece bandă semnalul, acesta devine mai puțin afectat de zgomot, banda utilă a semnalului fiind mai proeminentă în spectru.

Caracteristica filtrului trece bandă:

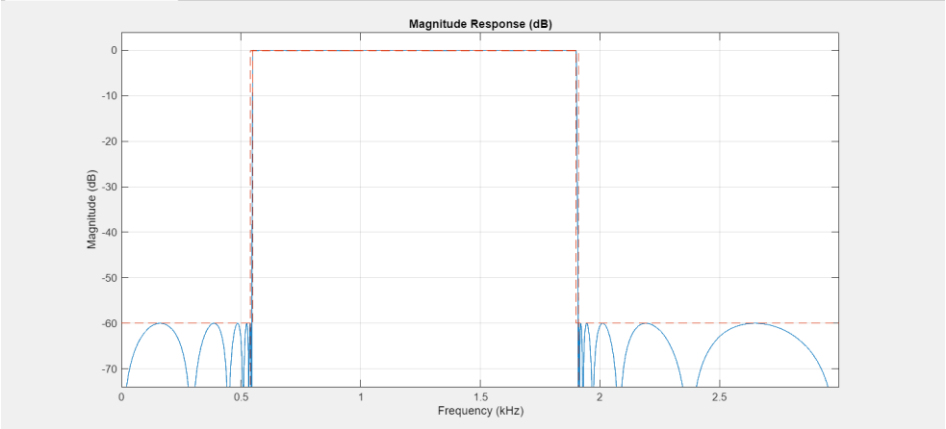

Figura 2.30: Interpretare rezultate semnalizare prin tonuri DTMF pentru numărul *123#, caracteristică FTB

Filtrul selectează banda între frecvențele de 550 Hz (inferioară) și 1900 Hz (superioară), atenuând componentele care nu se situează în acest interval.

Bibliografie

[1] Zsuzsanna Ilona Kiss, Zsolt Alfréd Polgár, *Telefonie, teorie și aplicații*, Mai 2020