Humeyra Bodur (2442606)
Berfin Kökmen (2443463)

07.01.2024

EE430 Term Project

Part 2

Introduction

In this part of the project, our focus centers on the exploration of dual-tone multi-frequency (DTMF) signaling. Specifically, we intend to implement signal processing techniques in MATLAB for the purpose of generating, transmitting, receiving, and decoding audio DTMF signals.For decoding algorithms, we will utilize the Spectrogram method and Goertzel method. According to tests conducted on the decoding methods by altering the period of tone duration and tone rest, we will compare these methods.

User Interface

There are two interfaces in this project: the Transmitter and the Receiver. In this section we will introduce both of the interfaces and explain how they are used.

The Transmitter Panel

In Figure 1, we see the Transmitter panel, which has six buttons, seven variable entries, and a keyboard. The keyboard is used to input numbers, symbols, or letters for encoding. Each key on the keyboard produces a signal with two tones: one high frequency and one low frequency.

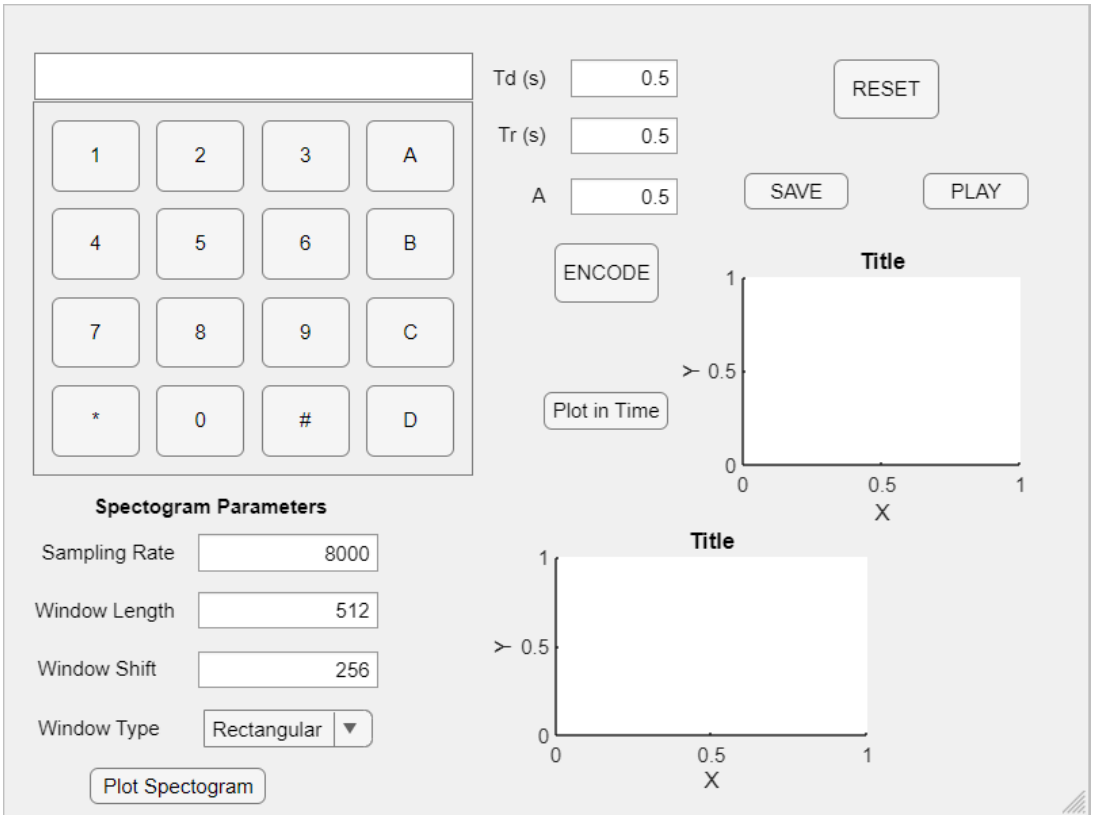


Figure 1. Transmitter Panel

Two inputs, Td and Tr, are used to set the duration of the dialing signal and the pause duration for each key, respectively. The variable 'A' represents the amplitude of the signal. Once these

inputs are entered, the system is ready to encode the dialing signal. Pressing the 'ENCODE' button generates the signal through the *'encoder'* function coded in Matlab. To store the generated time domain signal as a '.wav' or '.mp3' file, users can press the 'SAVE' button. Pressing the 'PLAY' button allows users to listen to the generated dialing signal. If users want to visualize the signal, the 'Plot in Time' button can be used for this purpose. Lastly, when pressed, the 'RESET' button will clear the keypad.

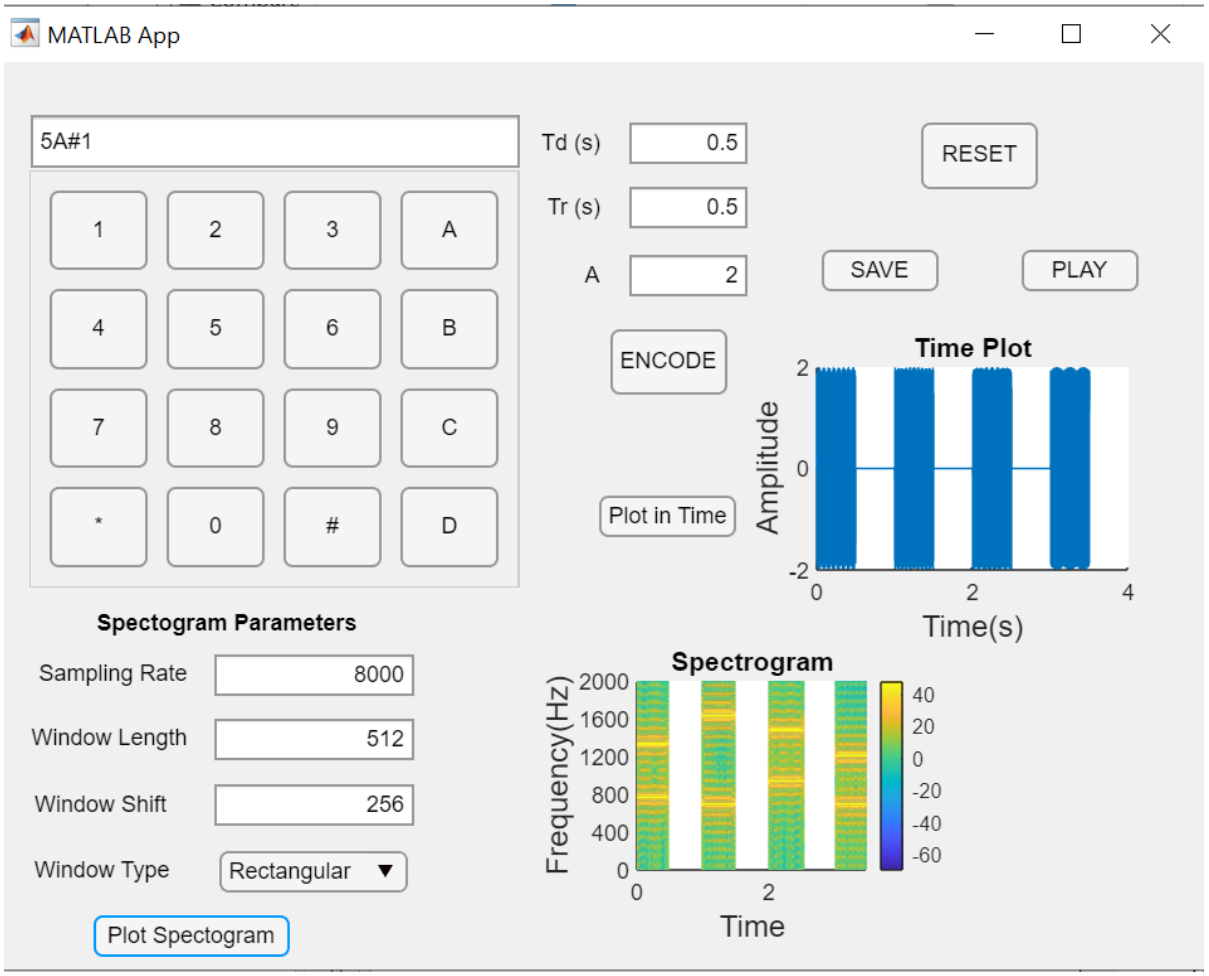


Figure 2. Displaying the encoded signal and its spectrogram

Figure 2 illustrates the operation of the 'Plot in Time' and 'Plot Spectrogram' buttons. The Time vs Amplitude plot displays the encoded signal. The Spectrogram displays the change of the frequency content of the signal in time.

The Encoder

The encoder function simply takes the input digits and generates a sum of 2 sine functions according to formula 1, where f_L and f_H correspond to the high and low frequencies of the k^th digit. Only a T_d long duration of this signal is taken. This is done for each digit given as input.

$$s^{(k)}(t;T_d) = \left(sin\left(f_L^{(k)}t\right) + sin\left(f_H^{(k)}t\right)\right) \cdot \left(u(t) - u(t-T_d)\right) \qquad (1)$$

Then, as in equation 2, we sum up these signals with a time shift, so that each consecutive digit is encoded in consecutive Td duration segments, where each segment is separated by Tr.

$$m(t;T_d,T_r) = \sum_{k=0}^{N-1} s^{(x[k])}(t - k(T_d + T_r)\,;\, T_d) \qquad (2)$$

The Receiver Panel

In Figure 3, we see the Receiver panel, which has 5 buttons, 5 variable entries, and a switch. The Td and Tr values are added so that the user can change the decoding parameters if desired. When the 'START' button is clicked the recording starts, and stops upon clicking the 'STOP' button. The user can choose one of the 2 decoding algorithms, and upon pressing the 'DECODE' key, the decoded signal will be displayed.

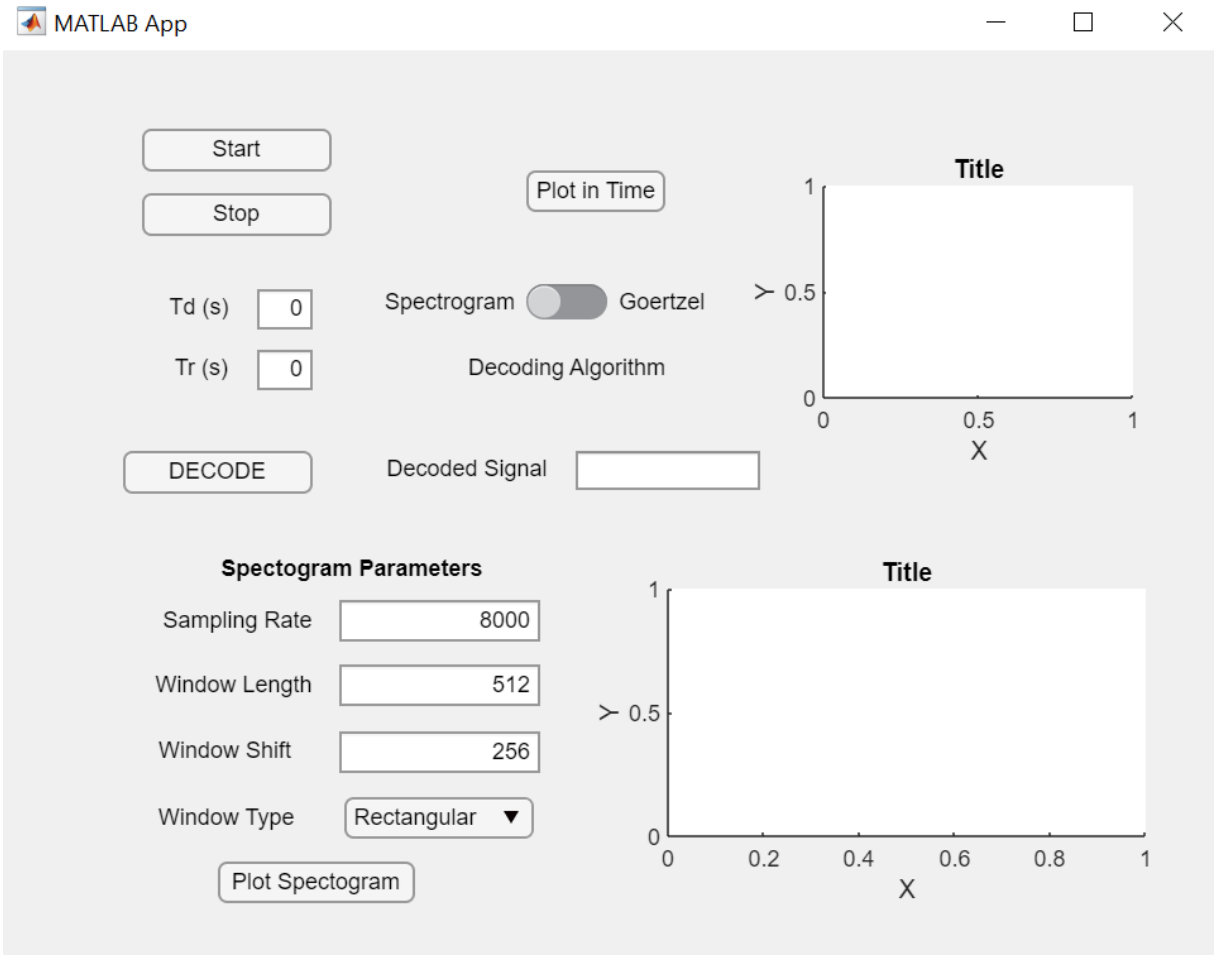


Figure 3. Receiver Panel

If users want to visualize the received signal, the 'Plot in Time' button can be used for this purpose. Lastly, when pressed, the 'RESET' button will clear the keypad.

Figure 4. Displaying the received signal and its spectrogram

Figure 4 illustrates the operation of the 'Plot in Time' and 'Plot Spectrogram' buttons. The Time vs Amplitude plot displays the received signal. The Spectrogram displays the change of the frequency content of that signal in time.

The Decoders

1-The Spectrogram Method

The spectrogram method analyzes the Fourier transform of the time-domain signal segment by segment, where each segment is of length tone_duration + tone_pause. The spectrogram analysis also requires user input for Td and Tr, although they could be obtained from a short time analysis, but our method assumed these values were already given. The signal goes through many filtering steps before its spectrogram analysis is done, in order to filter out noise that was captured during recording and to clip out any silent parts between pressing record and pause. The peak frequencies of each segment in the range of the defined “high frequencies” and “low frequencies” of DTMF are obtained for each segment. Then the obtained high frequency is compared to the DTMF High Frequencies in order to find out which of them is equal to it (within an error margin). The same is done for the low frequency component as well. Then the obtained high and low frequencies are mapped to the corresponding digit/symbol.

2-The Goertzel Method

The Goertzel algorithm implementation analyzes the strength of one of the two tones in an incoming signal across eight different Dual-Tone Multi-Frequency (DTMF) frequencies. The goal is to figure out which specific DTMF frequency is present. To achieve this, the input signal is converted to DTMF frequencies using a modified Goertzel algorithm. The matched filter concept is then applied separately to each DTMF frequency to identify the frequency where the incoming signal has the highest energy. Since the highest energy corresponds to the DTMF frequency, this process allows us to determine and detect the specific DTMF frequency present in the signal. Once the specific DTMF frequency is identified using this method, the decoder can readily associate it with the corresponding digit on the DTMF keypad.

Comparison of the Algorithms

The effect of changes in duration time and pause time has a consistent impact on the results of both algorithms. When these times are reduced, both algorithms encounter difficulty in accurately identifying the time intervals associated with encoded digits. Consequently, a decrease in duration and pause time leads to challenges in the detection process, causing the algorithms to potentially miss some digits during decoding.However, the Goertzel method is better at spotting encoded signals than the Spectrogram Method. It works well because it focuses only on the energy of specific frequencies, making it capable of identifying the strongest signal even when there are changes in timing.

Between these two algorithms, our observations revealed that the Goertzel Algorithm outperforms the Spectrogram Method in terms of speed. This is because the Goertzel Algorithm specifically targets certain frequency values, namely the high and low frequencies associated with the keypad's row and column numbers. In contrast, the spectrogram method calculates the Fourier transform for all frequency values, resulting in a slower execution.

Despite the Goertzel Algorithm's faster runtime, the Spectrogram Method demonstrates superior performance. This is particularly evident when dealing with slight variations in the received signal frequency. The Goertzel Method, constrained by its focus on limited frequency points, struggles to accurately detect encoded digits in such scenarios.

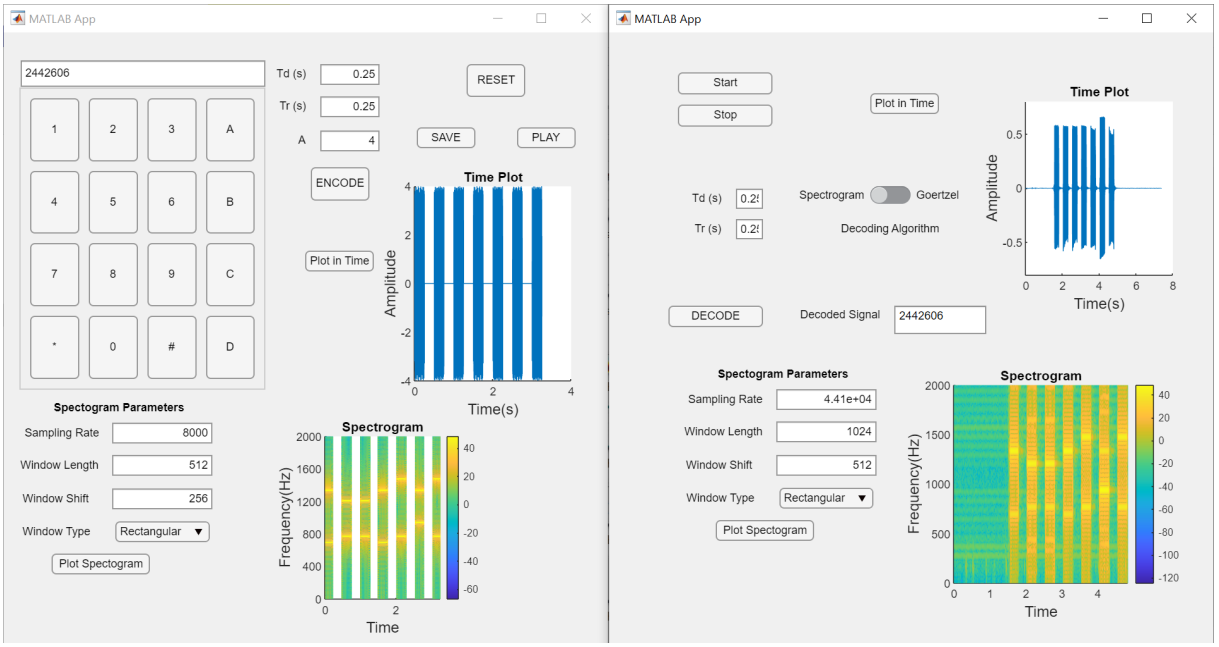
In summary, while the Goertzel Method is more time-efficient, the Spectrogram Method exhibits better overall performance. Considering our objectives prioritize accuracy over time efficiency, opting for the spectrogram method appears to be a reasonable choice for us.

Demonstration of the App with Examples

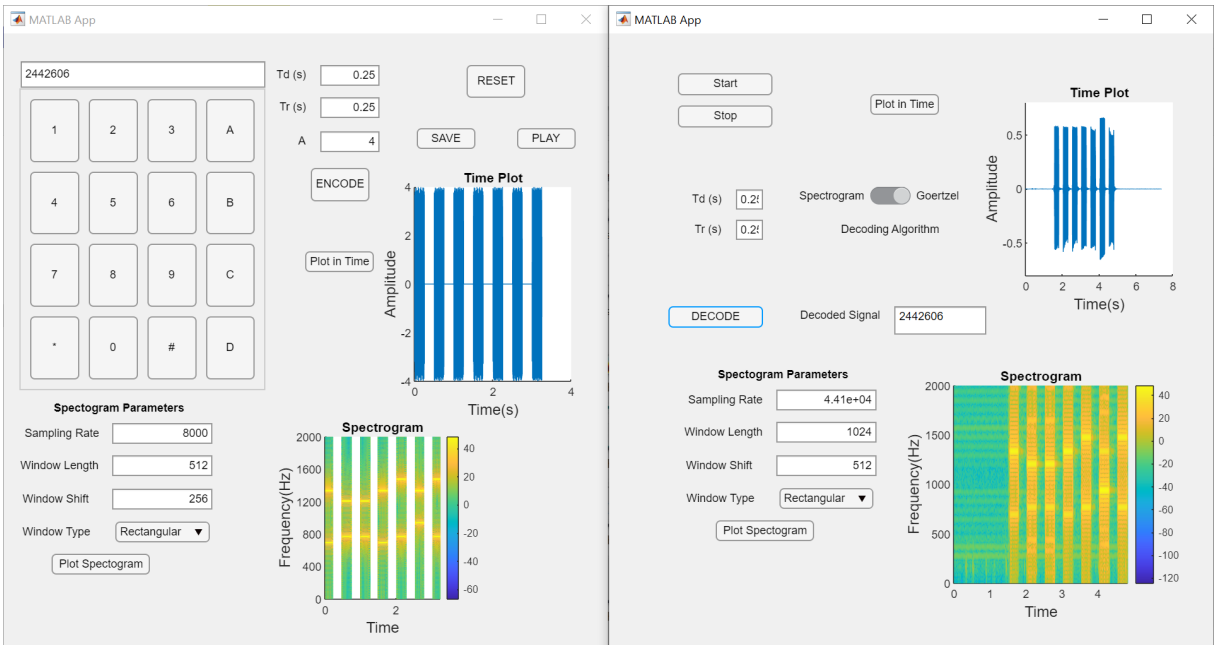
In this section we will analyze the application of both methods on 2 different keys but with different parameters. For the sake of demonstrating the app's functionality, the user interface will be used.

Input Code: 2442606

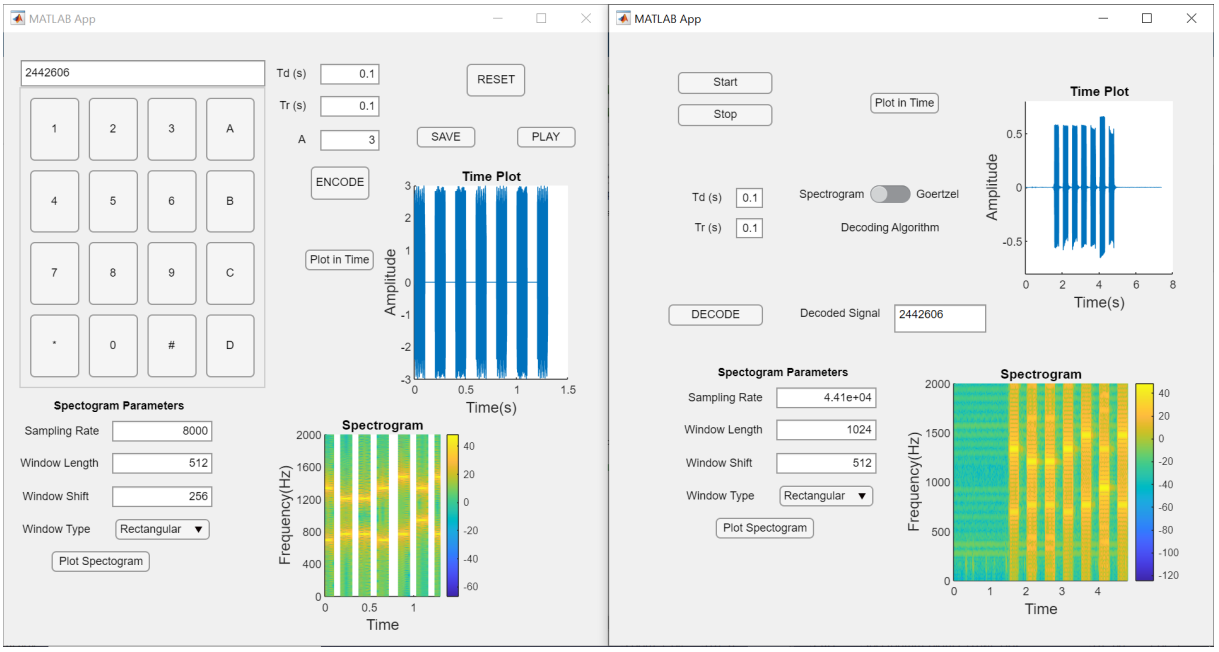
1)*Td=Tr = 0.25, Decoder = Spectrogram Method*



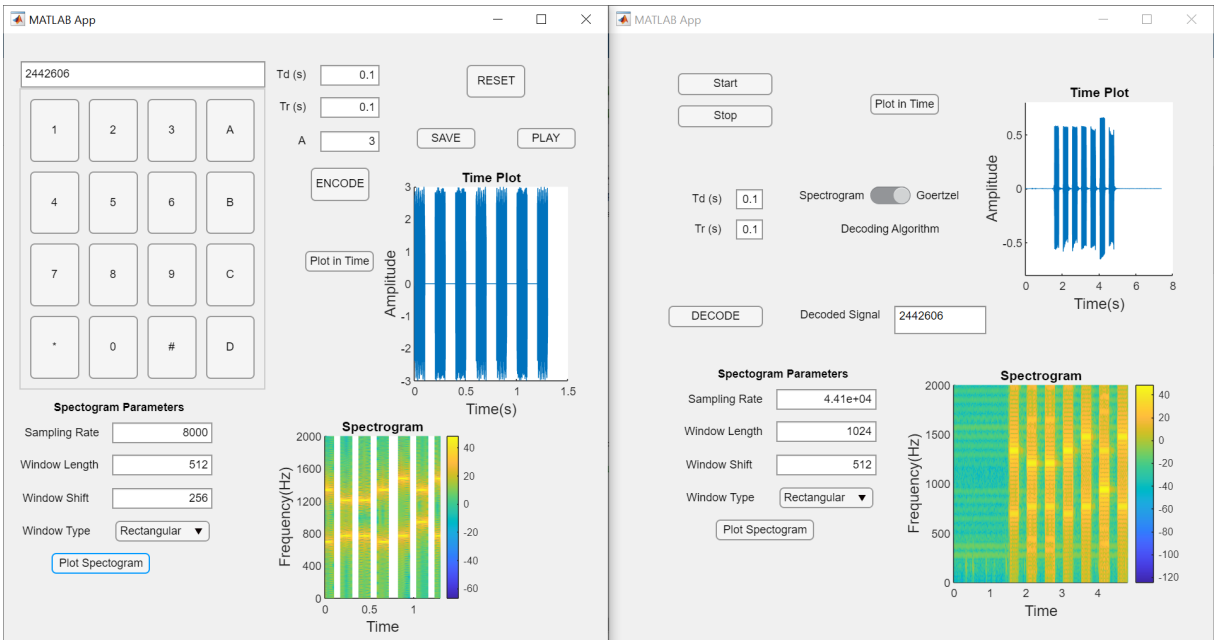
2)*Td=Tr = 0.25, Decoder = Goertzel Method*



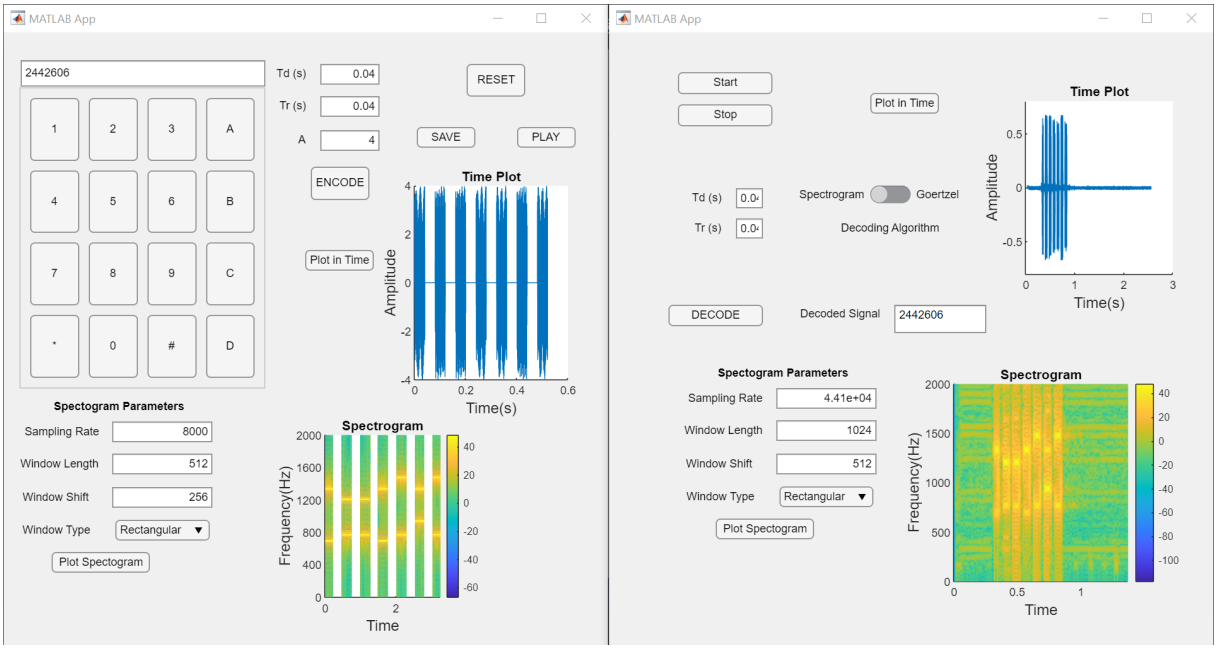
3)*Td=Tr = 0.1, Decoder = Spectrogram Method*



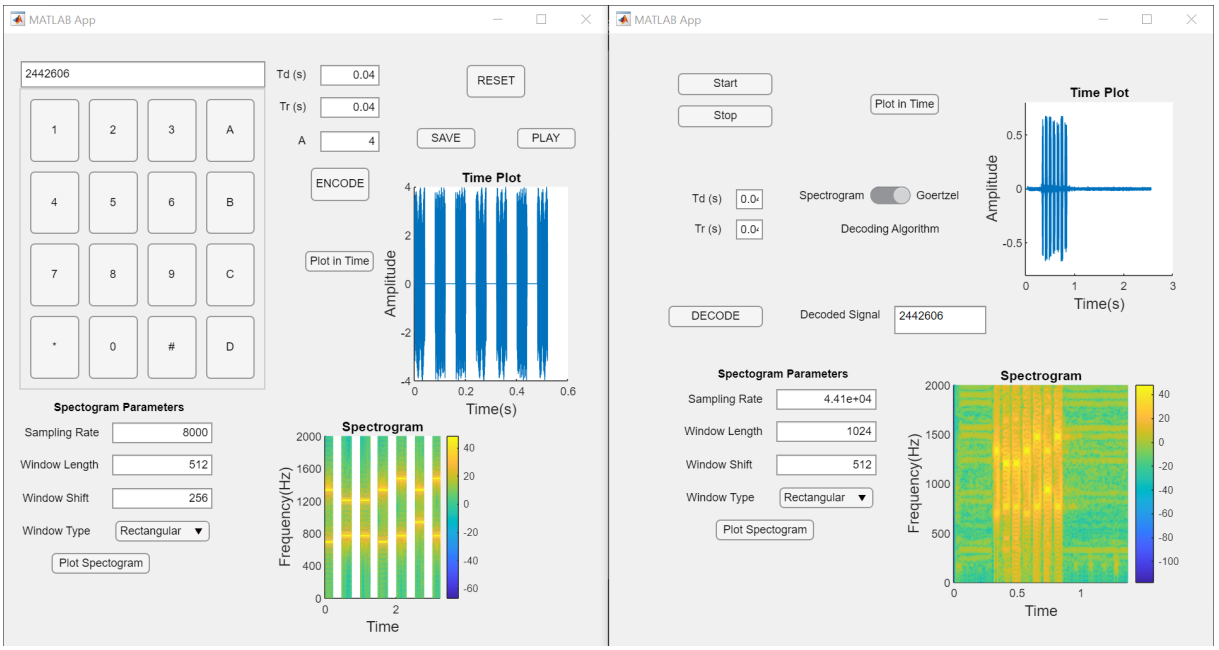
4)*Td=Tr = 0.1, Decoder = Goertzel Method*



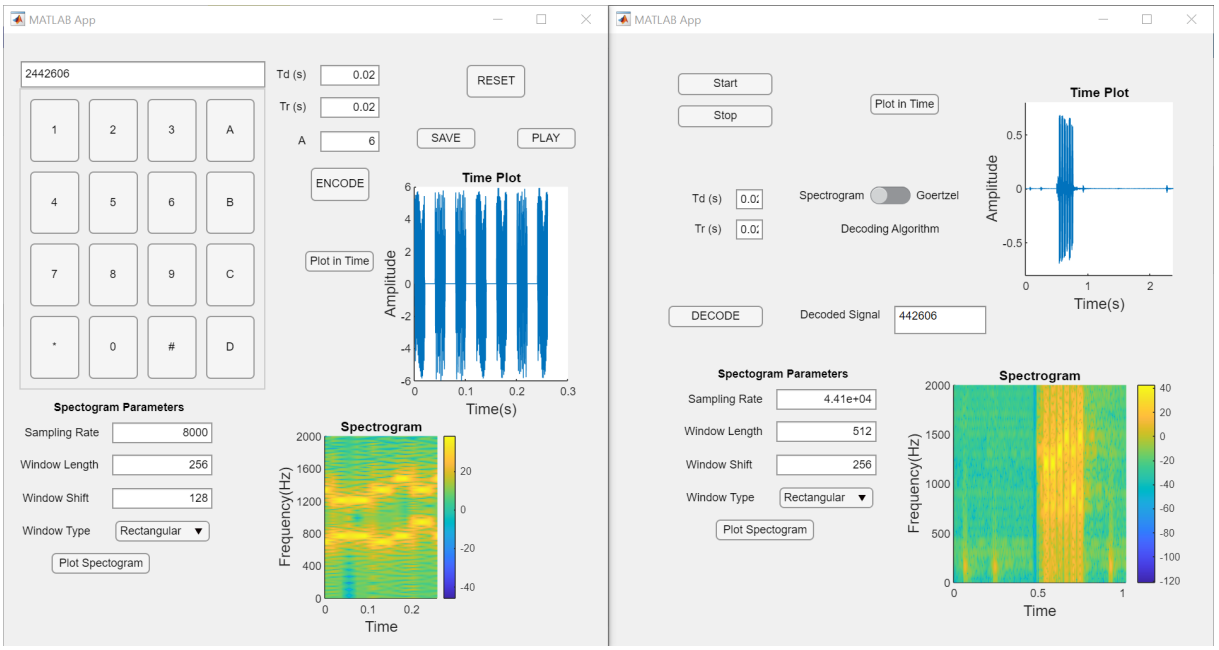
5)*Td=Tr = 0.04, Decoder = Spectrogram Method*



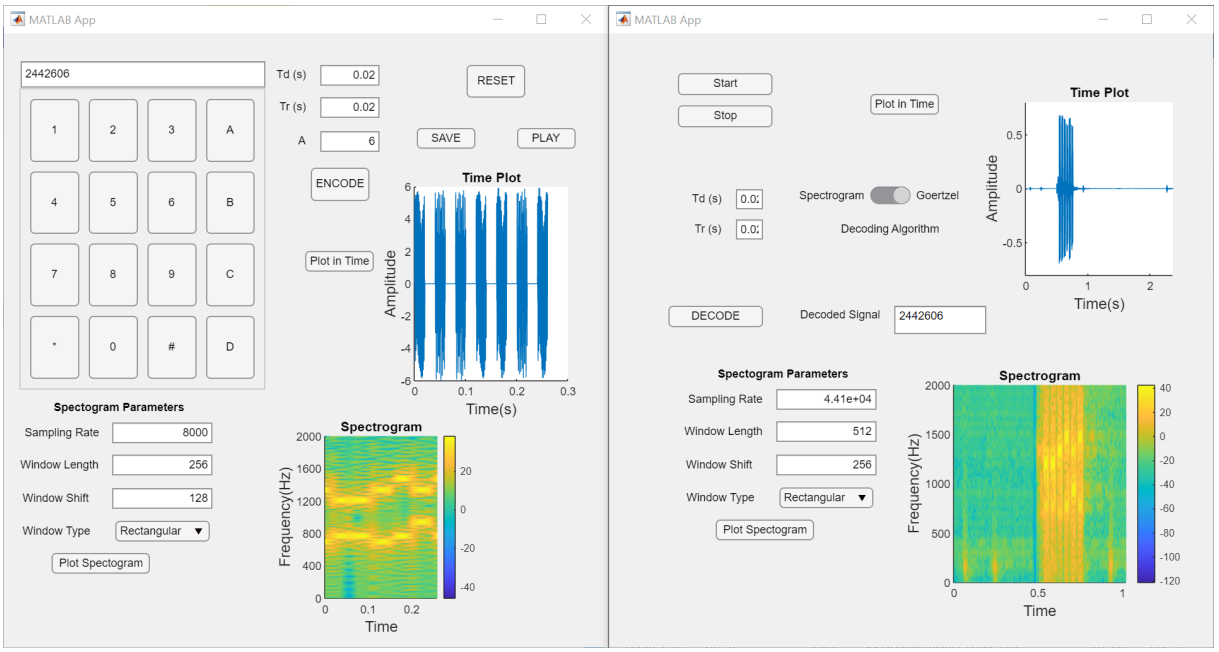
6)*Td=Tr = 0.04, Decoder = Goertzel Method*



7)*Td=Tr = 0.02, Decoder = Spectrogram Method*



8)*Td=Tr = 0.02, Decoder = Goertzel Method*

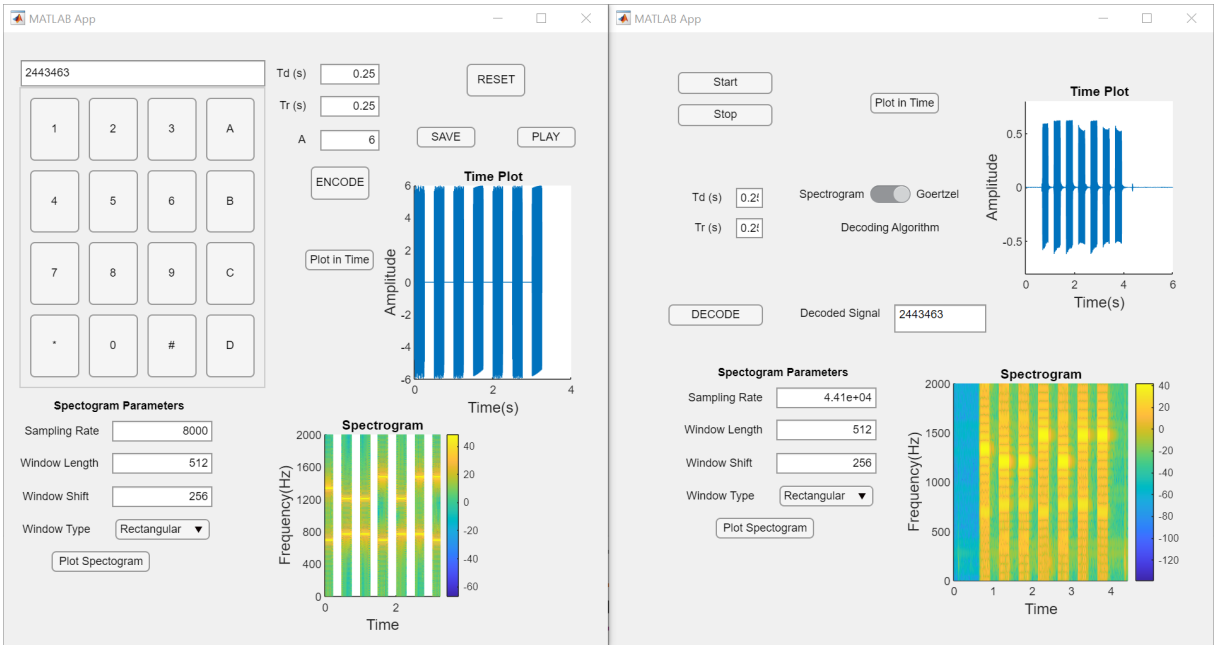


Humeyra Bodur (2442606)
Berfin Kökmen (2443463)

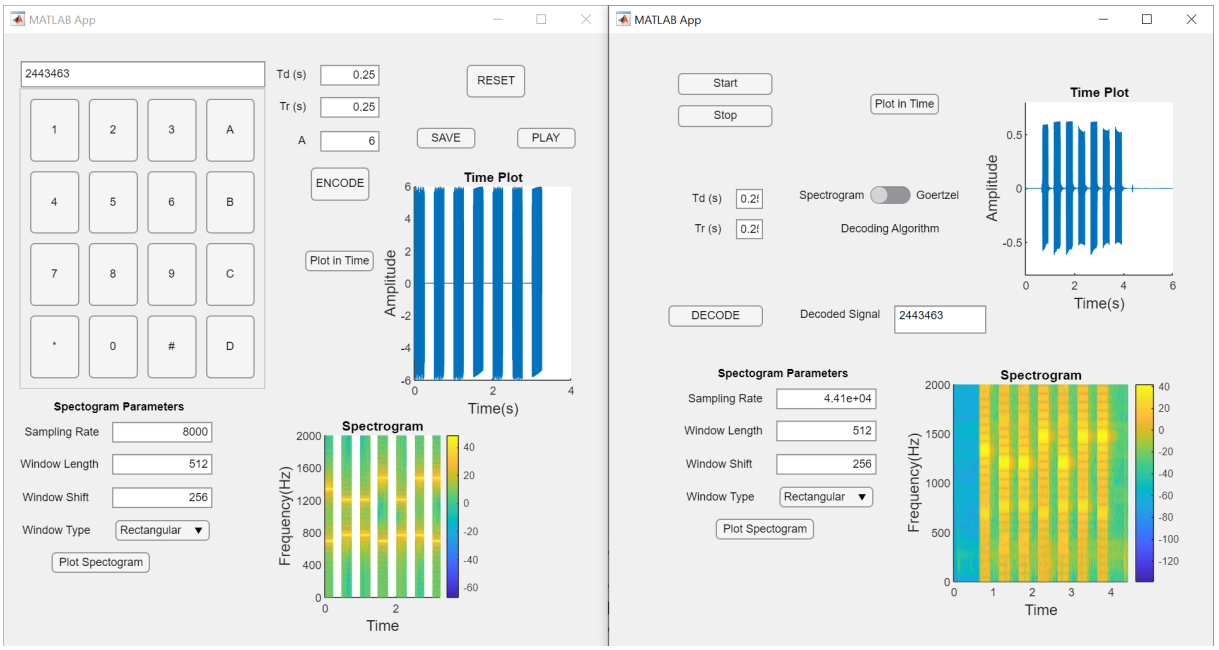
07.01.2024

Input Code: 2443463

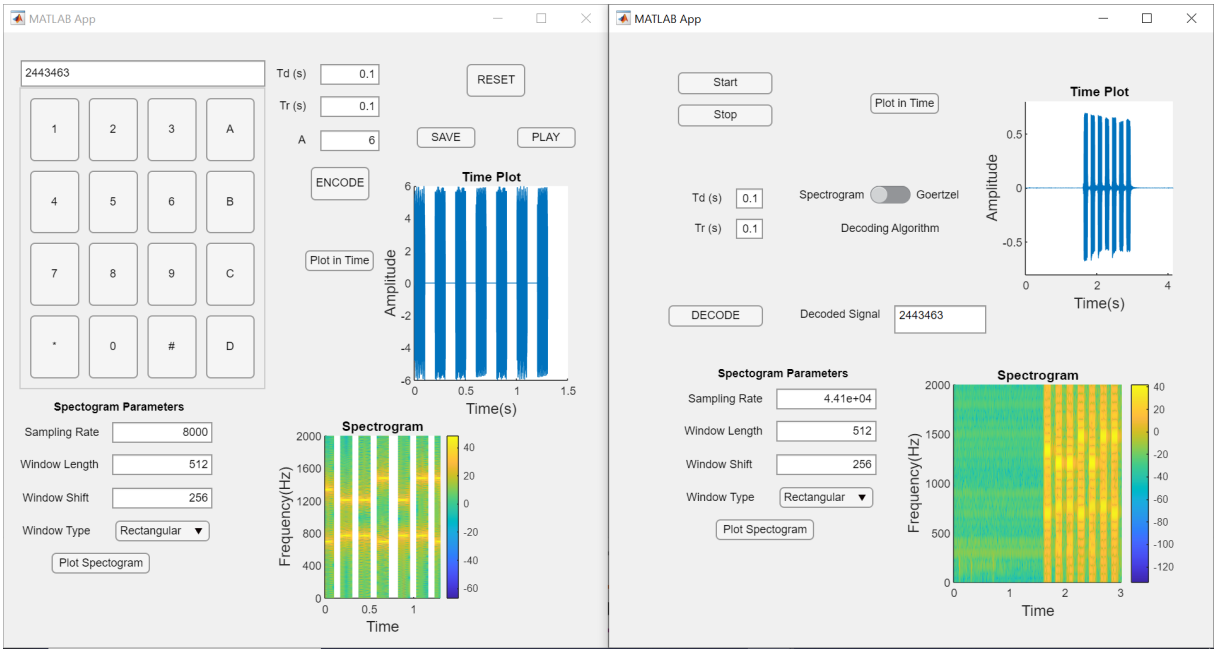
1)*Td=Tr = 0.25, Decoder = Spectrogram Method*



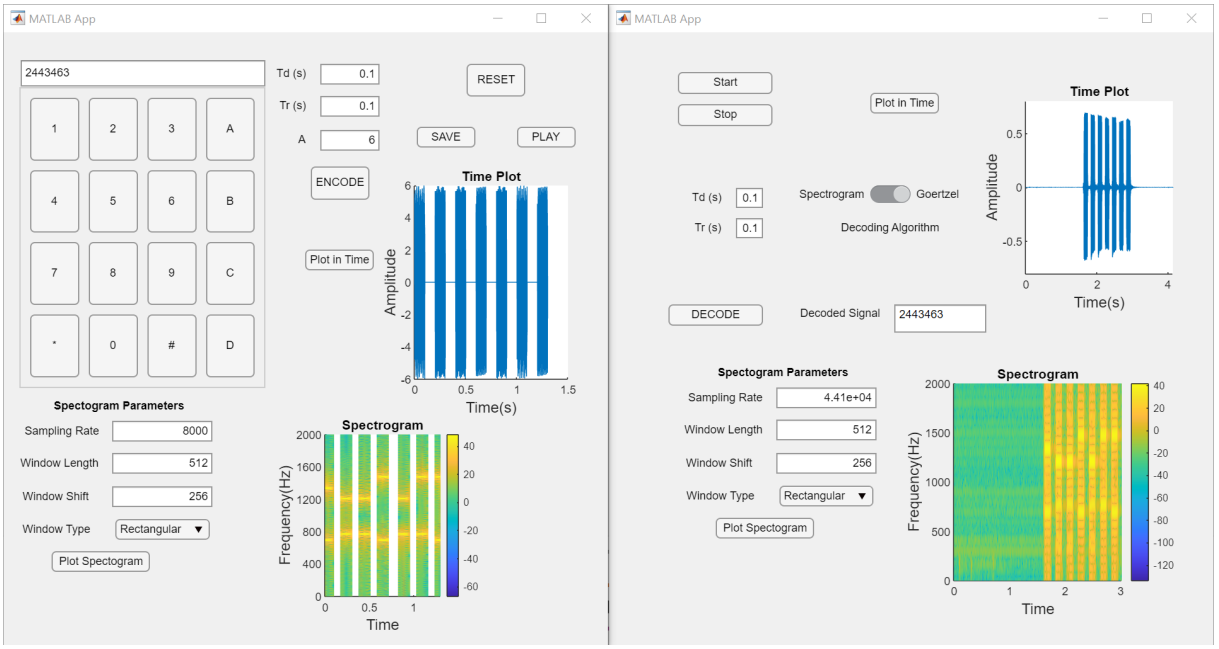
2)*Td=Tr = 0.25, Decoder = Goertzel Method*



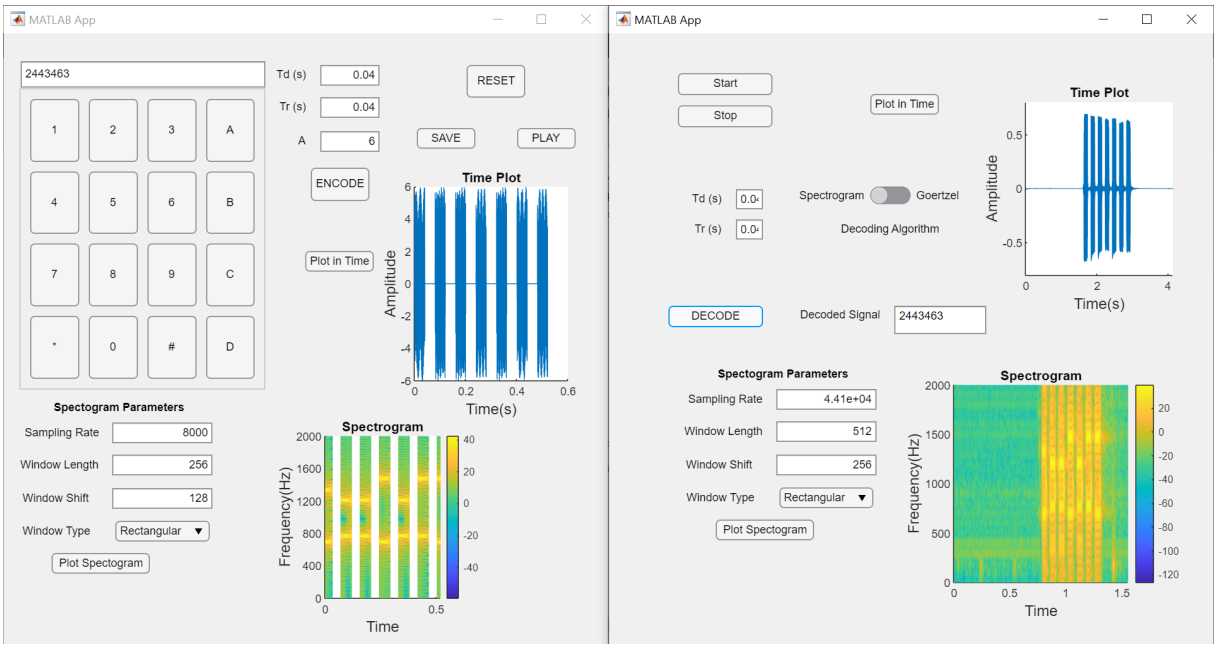
3)*Td=Tr = 0.1, Decoder = Spectrogram Method*



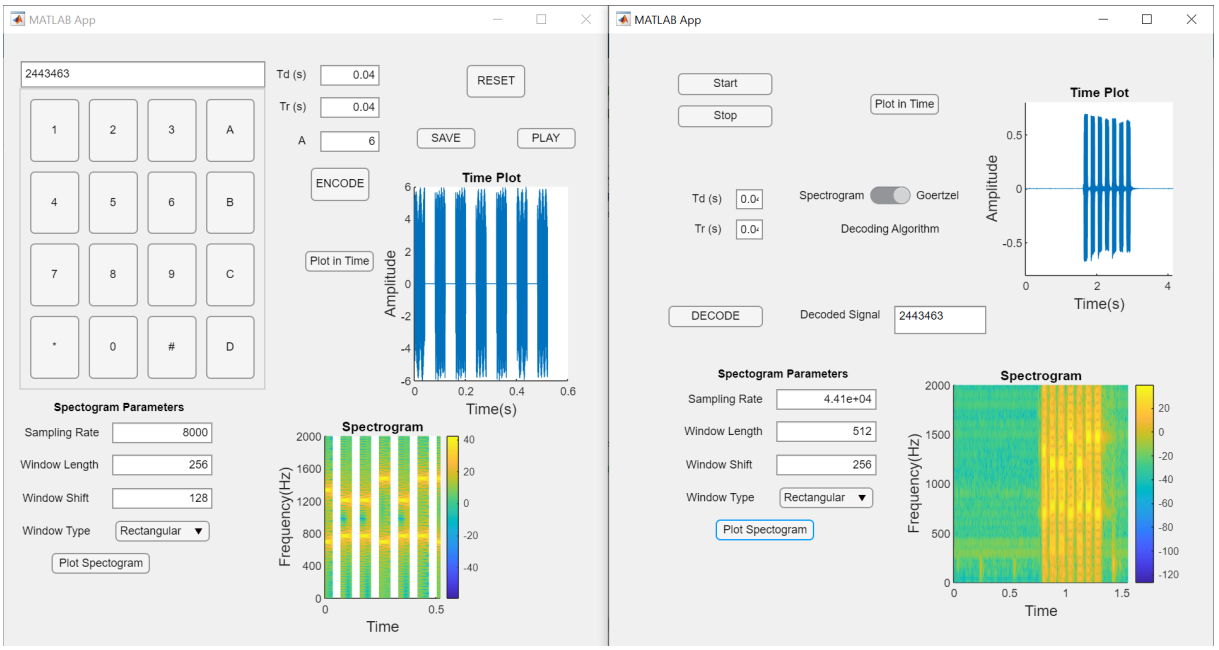
4)*Td=Tr = 0.1, Decoder = Goertzel Method*



5)Td=Tr = 0.04, Decoder = Spectrogram Method



6)Td=Tr = 0.04, Decoder = Goertzel Method



Comments on the Examples

Except for the case with key = 2442606, Td=Tr=0.02 with the spectrogram based decoder, we observe that each signal was decoded correctly, so our observations will be mostly comparing this result to the rest. The reason may be because it gets harder to differentiate the frequencies in a spectrogram with time when the changes in those frequencies happen quickly. So the frequency 1209 may have been detected as 1240 for instance, which is higher than our error tolerance in the decoding algorithm. The Goertzel algorithm however was able to do it since it looks only at the energy of certain frequencies, so even though such a change may appear, the power at that frequency will still be the highest.

Conclusion

In conclusion, our exploration into dual-tone multi-frequency (DTMF) signaling, implemented through MATLAB-based signal processing techniques, culminated in the development of a robust system for generating, transmitting, receiving, and decoding audio DTMF signals. The comparison between the Spectrogram method and the Goertzel method highlighted the consistent impact of changes in duration time and pause time on both algorithms, with the Spectrogram Method demonstrating superior overall performance despite the Goertzel Algorithm's faster execution. The detailed analysis of the Transmitter and Receiver interfaces, along with examples showcasing the application of both decoding methods, provided valuable insights into the system's functionality. Ultimately, our project successfully achieved its objectives, presenting a comprehensive and user-friendly platform for DTMF signal processing that prioritizes accuracy, making it a suitable choice for various applications.

MATLAB Codes

```
function dtmf_signal = encoder(digits, Fs, tone_duration, tone_pause)
  % Define DTMF Frequencies
  low_freqs = [697, 770, 852, 941];
  high_freqs = [1209, 1336, 1477, 1633];
  digit_map = ['1', '2', '3', 'A';
               '4', '5', '6', 'B';
               '7', '8', '9', 'C';
               '*', '0', '#', 'D'];
  % Initialize signal
  dtmf_signal = [];
  for i = 1:length(digits)
     % Find the row and column of the digit
     [row, col] = find(digit_map == digits(i));
     % Generate tones
     t = 0:1/Fs:tone_duration-1/Fs;
     tone1 = sin(2 * pi * low_freqs(row) * t);
     tone2 = sin(2 * pi * high_freqs(col) * t);
     % Combine tones and add to signal
     dtmf_signal = [dtmf_signal, tone1 + tone2];
     % Add pause between tones
     if i < length(digits)
        pause = zeros(1, Fs * tone_pause);
        dtmf_signal = [dtmf_signal, pause];
     end
  end
end
```

```
function spectogram_plotter_generated_signals_vol2(generated_signal,window_length,window_shift,window_type,sampling_frequency,tone_duration,tone_pause)
  sampling_frequency = sampling_frequency/2;
  % Transpose is taken if the signal is a column vector , instead of a
  % row vector

  if(size(generated_signal,1) > 5)
     generated_signal = generated_signal';
  end

  win = 0;    % initial blank window

  %% Controlling if we satisfy the length constraints:
  if( length(generated_signal) < window_length )

  disp(' Error: The signal is too short for this window length. Submit a new signal or choose a new window length.');
  end

  if( window_shift > window_length )
  disp(' Error: Enter a valid window length that is longer than the window shift.');
  end
  %% Assigning the window type:
  if( strcmp(window_type,'rectwin')) %#ok<*STCMP>
     win=ones(1,window_length);
  elseif(strcmp(window_type,'hamming'))
     win=hamming(window_length)';
  elseif(strcmp(window_type,'tukeywin'))
     win=tukeywin(window_length)';
```

```
  else
     win=ones(1,window_length);
  end
  %************************************************

  begin_point = 1;    % Initial point for thewindow
  vertical_line_no=1; % Counter for the vertical lines

  %% Calculating the STFT
  while( begin_point + window_length -1 <= length(generated_signal) )
     STFT(vertical_line_no,:) = abs(fft(generated_signal( begin_point :
begin_point+window_length-1 ).*win,window_length*100));
     begin_point = begin_point + window_shift;
     vertical_line_no = vertical_line_no+1 ;
  end
  signal_duration = length(generated_signal) / (2*sampling_frequency);
  time_values = 0 : size(STFT,1)-1; %Time values to be plotted
  t_values = linspace(0,signal_duration,length(time_values));
  time_values = t_values;
  frequency_values = linspace(0,sampling_frequency,window_length*50); %The default fs =
4000
%   frequency_values = (4000/sampling_frequency)*frequency_values
  STFT_in_DB=(20*log10(STFT))'; % Coverting the magnitudes to decibels for better analysis
  if (sampling_frequency > 1999)
     y_limit = 2000;
  else
     y_limit = sampling_frequency;
  end
  %% Plotting the STFT
  figure
```

```
  surf(time_values,frequency_values,STFT_in_DB(1:window_length*50,:));
  shading("interp");
  view(2); %The view should be from above since our plot is 2D in view
  xlabel('Time','Fontsize',12);
  ylabel('Frequency(Hz)','Fontsize',12);
  title('Spectrogram','Fontsize',14);
  colorbar;
  ylim([0 2000]);
  xlim([0 signal_duration]);
end


function
spectogram_plotter_DTMF_GUI(axes,generated_signal,window_length,window_shift,windo
w_type,sampling_frequency)


  disp("sampling freq is")
  sampling_frequency = sampling_frequency/2;
  % Transpose is taken if the signal is a column vector , instead of a
  % row vector


  if(size(generated_signal,1) > 5)
     generated_signal = generated_signal';
  end


  win = 0;    % initial blank window


  %% Controlling if we satisfy the length constraints:
  if( length(generated_signal) < window_length )
```

```
  disp(' Error: The signal is too short for this window length. Submit a new signal or choose a
new window length.');
  end

  if( window_shift > window_length )
  disp(' Error: Enter a valid window length that is longer than the window shift.');
  end
  %% Assigning the window type:
  if( strcmp(window_type,'rectwin')) %#ok<*STCMP>
     win=ones(1,window_length);
  elseif(strcmp(window_type,'hamming'))
     win=hamming(window_length)';
  elseif(strcmp(window_type,'tukeywin'))
     win=tukeywin(window_length)';
  else
     win=ones(1,window_length);
  end
  %************************************************

  begin_point = 1;    % Initial point for thewindow
  vertical_line_no=1; % Counter for the vertical lines

  %% Calculating the STFT

  while( begin_point + window_length -1 <= length(generated_signal) )

     STFT(vertical_line_no,:) = abs(fft(generated_signal( begin_point :
begin_point+window_length-1 ).*win,window_length*100));
     begin_point = begin_point + window_shift;
     vertical_line_no = vertical_line_no+1 ;
```

```
  end
  signal_duration = length(generated_signal) / (2*sampling_frequency);
  time_values = 0 : size(STFT,1)-1; %Time values to be plotted
  t_values = linspace(0,signal_duration,length(time_values));
  time_values = t_values;
  frequency_values = linspace(0,sampling_frequency,window_length*50) %The default fs =
4000
  frequency_values = (4000/sampling_frequency)*frequency_values
  STFT_in_DB=(20*log10(STFT))'; % Coverting the magnitudes to decibels for better analysis
  if (sampling_frequency > 1999)
     y_limit = 2000;
  else
     y_limit = sampling_frequency;
  end

  %% Plotting the STFT

  surf(axes,time_values,frequency_values,STFT_in_DB(1:window_length*50,:));
  shading(axes,"interp");
  view(axes,2); %The view should be from above since our plot is 2D in view
  xlabel(axes,'Time','Fontsize',12);
  ylabel(axes,'Frequency(Hz)','Fontsize',12);
  title(axes,'Spectrogram','Fontsize',14);
  colorbar(axes);
  ylim(axes,[0 y_limit]);
  xlim(axes,[0 signal_duration]);
  yticks(axes,linspace(0, sampling_frequency, 11));
end
```

```
function decoded_digits = dtmf_decoder_spectogram_GUI(signal, Fs, tone_duration,
tone_pause)
  %%  Signal Preprocessing begins
  % signal,Fs,Tr,Td are needed

    if(tone_duration<0.3)
    %***********************************
    % This might cause some problems, if it does just erase it.
    threshold = max(abs(signal))/5;
    indices_below_threshold = find(abs(signal)<threshold);
    signal(indices_below_threshold) = 0;
    %***********************************
    end
    Td = tone_duration;
    Tr = tone_pause;
    % Set the threshold duration (adjust as needed)
    segment_length_duration = Td*Fs; % Td*Fs yaparsın
    segment_length_rest = Tr*Fs;

    threshold = max(abs(signal))/20;

    % Find indices where amplitude exceeds the threshold
    highAmplitudeIndices = find(abs(signal) > threshold);

    % Identify the start and end indices of the region with higher amplitudes
    startIndex = min(highAmplitudeIndices);
    endIndex = max(highAmplitudeIndices);

    signal_clipped=signal(startIndex:endIndex);
    signal = signal_clipped;
```

```
    first_segment = signal(1:segment_length_duration);
    end_index = length(signal);

    rest_of_the_signal = signal(segment_length_duration+1:end_index);
    cut_this =
mod(length(rest_of_the_signal),segment_length_rest+segment_length_duration);

    if(cut_this<segment_length_rest/2)
       new_ending = length(rest_of_the_signal)-cut_this;
       rest_of_the_signal = rest_of_the_signal(1:new_ending);
    end

    iteration_time = length(rest_of_the_signal) /
(segment_length_duration+segment_length_rest);
    i=1;
    while(i<(iteration_time+1))

rest_of_the_signal((i-1)*(segment_length_duration+segment_length_rest)+1:(i-1)*(segment
_length_duration+segment_length_rest)+segment_length_rest) = 0;
       i=i+1;
    end

    new_signal = [first_segment; rest_of_the_signal];

    signal = new_signal;
    % Making up for the lost signal during recording
    signal= padarray(signal,4000,'post');
  % Signal Preprocessing ends.
%%  Define DTMF Frequencies
  low_frequencies = [697, 770, 852, 941];
```

```
high_frequencies = [1209, 1336, 1477, 1633];
digit_map = ['1', '2', '3', 'A';
        '4', '5', '6', 'B';
        '7', '8', '9', 'C';
        '*', '0', '#', 'D'];
% Initialize variables
decoded_digits = [];
segment_length = Fs*(tone_duration + tone_pause);
tone_samples = Fs * tone_duration;
% Processing each segment
for start_idx = 1:segment_length:length(signal) - tone_samples - 1
   % Extracting one segment
   segment = signal(start_idx:start_idx + tone_samples - 1);

   % Instead of finding the Spectogram by taking the FFT of segments
   % and then taking the segment and extracting the frequencies, we
   % chose to do the spectogram segment by segment.
   % FFT is performed here
   N = length(segment);
   f = Fs*(0:(N/2))/N;
   Y = fft(segment);
   P2 = abs(Y/N);
   P1_high = P2(1:N/2+1);
   P1_low = P1_high;
   % Set magnitudes to 0 for frequencies above the threshold
   indices_above_threshold = find(f>1000);
   P1_low(indices_above_threshold) = 0;
   indices_below_threshold = find(f<600);
   P1_low(indices_below_threshold) = 0;
```

```
    indices_below_threshold = find(f<1150);
    P1_high(indices_below_threshold) = 0;
    indices_above_threshold = find(f>1690);
    P1_high(indices_above_threshold) = 0;

    % We find where the peaks in the FFT magnitude are
    [pks_high, locs_high] = max(P1_high); %findpeaks(P1_high, 'MinPeakHeight', max(P1_high)/10, 'SortStr', 'descend');
    % We find where the peaks in the FFT magnitude are
    [pks_low, locs_low] = max(P1_low); %findpeaks(P1_low, 'MinPeakHeight', max(P1_low)/10, 'SortStr', 'descend');


% We choose the two highest peaks, which correspond to our low and high frequency magnitudes

    if(isempty(locs_low)|| isempty(locs_high))
       break
    end
    freqs_high = f(locs_high(1))
    freqs_low = f(locs_low(1))
    % We assign the smaller one to low frequency component and the larger
    % one to the high frequency component
    cur_low_freq = freqs_low;
    cur_high_freq = freqs_high;

    % We give an error margin of magnitude 10
    for i=1:4
       if(abs(low_frequencies(i)-cur_low_freq)<25)
          cur_low_freq = low_frequencies(i);
```

```
        end
    end
    for i=1:4
        if(abs(high_frequencies(i)-cur_high_freq)<35)
            cur_high_freq = high_frequencies(i);
        end
    end
    % We find the corresponding elements to the high and low
    % frequencies we found
    [col, row] = find(low_frequencies == cur_low_freq & high_frequencies' ==
cur_high_freq);


    if ~isempty(row) && ~isempty(col)
        digit = digit_map(row, col);
        decoded_digits = [decoded_digits, digit];
    end
  end
end



function decoded_digits=dtmf_goertzel_decoder_GUI(signal, fs,tone_duration,tone_pause)
  low_freq = [697, 770, 852, 941] ;
  high_freq = [1209, 1336, 1477,1633];
  dtmfKeys = ['1', '2', '3', 'A';
          '4', '5', '6', 'B';
          '7', '8', '9', 'C';
          '*', '0', '#', 'D'];
   %Signal Preprocessing begins:
  % signal,Fs,Tr,Td are needed
```

```
if(tone_duration<0.3)
    %************************************
    % This might cause some problems, if it does just erase it.
    threshold = max(abs(signal))/5;
    indices_below_threshold = find(abs(signal)<threshold);
    signal(indices_below_threshold) = 0;
    %************************************
end

    Fs=fs;
    Td = tone_duration;
    Tr = tone_pause;
    % Set the threshold duration (adjust as needed)
    segment_length_duration = Td*Fs; % Td*Fs yaparsın
    segment_length_rest = Tr*Fs;

    threshold = max(abs(signal))/20;

    % Find indices where amplitude exceeds the threshold
    highAmplitudeIndices = find(abs(signal) > threshold);

    % Identify the start and end indices of the region with higher amplitudes
    startIndex = min(highAmplitudeIndices);
    endIndex = max(highAmplitudeIndices);

    signal_clipped=signal(startIndex:endIndex);
    signal = signal_clipped;

    first_segment = signal(1:segment_length_duration);
```

```
    end_index = length(signal);


    rest_of_the_signal = signal(segment_length_duration+1:end_index);


    % Duruma göre çıkarabilirsin bunu
    cut_this =
mod(length(rest_of_the_signal),segment_length_rest+segment_length_duration);


    if(cut_this<segment_length_rest/2)
       new_ending = length(rest_of_the_signal)-cut_this;
       rest_of_the_signal = rest_of_the_signal(1:new_ending);
    end
    % buraya kadar


    iteration_time = length(rest_of_the_signal) /
(segment_length_duration+segment_length_rest);


    i=1;
    while(i<(iteration_time+1))

rest_of_the_signal((i-1)*(segment_length_duration+segment_length_rest)+1:(i-1)*(segment
_length_duration+segment_length_rest)+segment_length_rest) = 0;
       i=i+1;
    end


    new_signal = [first_segment; rest_of_the_signal];


    signal = new_signal;
  %Signal Preprocessing ends.


  N = length(signal);
```

```
  segment_lenght=(tone_duration+tone_pause)*fs;
  digit_number= ceil((N)/segment_lenght);
  row=zeros(1,digit_number);
  col=zeros(1,digit_number);
  power_high=zeros(1,4);
  power_low=zeros(1,4);
%    disp(digit_number);
%    disp(N);

  decodedKeys=zeros(1,digit_number);
  decoded_digits=zeros(1,digit_number);
  startIndex = 1;
  endIndex = length(signal);
  %Goertzel algorithm to detect frequencies
for i=1:digit_number

  if i==digit_number
     endIdx=endIndex-startIndex-1;
     startIdx = endIdx-tone_duration*fs;
%       disp(endIdx);
%       disp(startIdx);
  else
     startIdx = round((i - 1)*segment_lenght) + 1;
     endIdx = round(startIdx+tone_duration*fs);
%       disp(endIdx);
%       disp(startIdx);
  end
  for c = 1:length(low_freq)
     k = round((tone_duration*fs) * low_freq(c) / fs);
```

```
    omega = (2 * pi * k) / (tone_duration*fs);
    coeff = 2 * cos(omega);

    s_prev = 0;
    s_prev2 = 0;
    for j = startIdx:endIdx
        s = signal(j) + coeff * s_prev - s_prev2;
        s_prev2 = s_prev;
        s_prev = s;
    end
    power = s_prev2^2 + s_prev^2 - coeff * s_prev * s_prev2;
    %disp(power);
    power_low(c)=power;
    %disp(power_low)
  end
    for d = 1:length(high_freq)
    k = round((tone_duration*fs) * high_freq(d) / fs);
    omega = (2 * pi * k) / (tone_duration*fs);
    coeff = 2 * cos(omega);

    s_prev = 0;
    s_prev2 = 0;
    for j = startIdx:endIdx
        s = signal(j) + coeff * s_prev - s_prev2;
        s_prev2 = s_prev;
        s_prev = s;
    end
    power1 = s_prev2^2 + s_prev^2 - coeff * s_prev * s_prev2;
    %disp(power1);
```

```
    power_high(d)=power1;
     end
    [~,row(i)]=max(power_low);
%       disp(row(i));
    [~,col(i)]=max(power_high);
%       disp(col(i));
    power_low=zeros(1,4);
    power_high=zeros(1,4);
  decodedKeys(i)=dtmfKeys(row(i),col(i));
end
for r=1:digit_number
  decodedKeys(r)=dtmfKeys(row(r),col(r));
  decoded_digits=char(decodedKeys);
end
end
%-------------------------------------------------------------------------------------------------------
classdef TransmitterPanel < matlab.apps.AppBase
   % Properties that correspond to app components
   properties (Access = public)
       UIFigure                    matlab.ui.Figure
       PlotinTimeButton            matlab.ui.control.Button
       AEditField                  matlab.ui.control.NumericEditField
       AEditFieldLabel             matlab.ui.control.Label
       TrsEditField                matlab.ui.control.NumericEditField
       TrsEditFieldLabel           matlab.ui.control.Label
       EditField                   matlab.ui.control.EditField
       ENCODEButton                matlab.ui.control.Button
       PLAYButton                  matlab.ui.control.Button
       SAVEButton                  matlab.ui.control.Button
       SpectogramParametersLabel   matlab.ui.control.Label
       TdsEditField                matlab.ui.control.NumericEditField
       TdsEditFieldLabel           matlab.ui.control.Label
```

```
        RESETButton                 matlab.ui.control.Button
        KeyPadPanel                 matlab.ui.container.Panel
        GridLayout                  matlab.ui.container.GridLayout
        DButton                     matlab.ui.control.Button
        Button_13                   matlab.ui.control.Button
        Button_12                   matlab.ui.control.Button
        Button_11                   matlab.ui.control.Button
        CButton                     matlab.ui.control.Button
        Button_10                   matlab.ui.control.Button
        Button_9                    matlab.ui.control.Button
        Button_8                    matlab.ui.control.Button
        BButton                     matlab.ui.control.Button
        Button_7                    matlab.ui.control.Button
        Button_6                    matlab.ui.control.Button
        Button_5                    matlab.ui.control.Button
        AButton                     matlab.ui.control.Button
        Button_3                    matlab.ui.control.Button
        Button_2                    matlab.ui.control.Button
        Button                      matlab.ui.control.Button
        SamplingRateEditField       matlab.ui.control.NumericEditField
        SamplingRateEditFieldLabel  matlab.ui.control.Label
        WindowTypeDropDown          matlab.ui.control.DropDown
        WindowTypeDropDownLabel     matlab.ui.control.Label
        WindowShiftEditField        matlab.ui.control.NumericEditField
        WindowShiftEditFieldLabel   matlab.ui.control.Label
        WindowLengthEditField       matlab.ui.control.NumericEditField
        WindowLengthEditFieldLabel  matlab.ui.control.Label
        PlotSpectogramButton        matlab.ui.control.Button
        UIAxesTime                  matlab.ui.control.UIAxes
        UIAxesSpectogram            matlab.ui.control.UIAxes
    end


    properties (Access = public)
        % Lazım olursa variablelları gir buraya
        DTMF_signal
```

```
    end

    % Callbacks that handle component events
    methods (Access = private)
        % Button pushed function: Button
        function ButtonPushed(app, event)
            key_value = '1';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_2
        function Button_2Pushed(app, event)
            key_value = '2';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_3
        function Button_3Pushed(app, event)
            key_value = '3';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: AButton
        function AButtonPushed(app, event)
            key_value = 'A';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_5
        function Button_5Pushed(app, event)
            key_value = '4';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_6
        function Button_6Pushed(app, event)
            key_value = '5';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_7
```

```
        function Button_7Pushed(app, event)
            key_value = '6';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: BButton
        function BButtonPushed(app, event)
            key_value = 'B';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_8
        function Button_8Pushed(app, event)
            key_value = '7';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_9
        function Button_9Pushed(app, event)
            key_value = '8';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_10
        function Button_10Pushed(app, event)
            key_value = '9';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: CButton
        function CButtonPushed(app, event)
            key_value = 'C';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_11
        function Button_11Pushed(app, event)
            key_value = '*';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_12
```

```
        function Button_12Pushed(app, event)
            key_value = '0';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: Button_13
        function Button_13Pushed(app, event)
            key_value = '#';
            app.EditField.Value = [app.EditField.Value, key_value];
        end
        % Button pushed function: DButton
        function DButtonPushed(app, event)
            key_value = 'D';
            app.EditField.Value = [app.EditField.Value, key_value];
%             app.enetered_digits = fliplr(app.EditField.Value);
%             disp(app.enetered_digits)
        end
        % Button pushed function: RESETButton
        function RESETButtonPushed(app, event)
            app.EditField.Value = '';
        end
        % Button pushed function: ENCODEButton
        function ENCODEButtonPushed(app, event)
            Fs = 8000; % Sampling frequency
            tone_duration = app.TdsEditField.Value; % Duration of each tone
            tone_pause = app.TrsEditField.Value; % Pause between tones
            digits = app.EditField.Value; % Digits to encode

            % Encode the digits
            encoded_signal = encoder(digits, Fs, tone_duration, tone_pause);
            app.DTMF_signal = encoded_signal;
        end
        % Button pushed function: SAVEButton
        function SAVEButtonPushed(app, event)

            % The encoded signal:
```

```
            signal = app.DTMF_signal;

            % Specify the file name and path
            filename = 'last_saved_signal.wav';

            % Specify the sampling rate (replace this with your actual
sampling rate)
            Fs = 8000;

            % Save the signal to a .wav file
            audiowrite(filename, signal, Fs);
        end
        % Button pushed function: PLAYButton
        function PLAYButtonPushed(app, event)
            % The encoded signal
            signal = app.DTMF_signal;

            % Specify the sampling rate (replace this with your actual
sampling rate)
            Fs = 8000;

            % Play the signal as a sound
            sound(signal, Fs);
        end
        % Button pushed function: PlotSpectogramButton
        function PlotSpectogramButtonPushed(app, event)

            audioData = app.DTMF_signal;
            length(audioData)
            samplingRate = app.SamplingRateEditField.Value

            selectedWindow = app.WindowTypeDropDown.Value;

            switch selectedWindow
                    case 'Rectangular'
```

```
                        WindowType = 'rectwin';
                    case 'Tukey'
                        WindowType = 'tukeywin';
                    case 'Hamming'
                        WindowType = 'hamming';
                    otherwise
                        WindowType = 'rectwin';
                end


spectogram_plotter_DTMF_GUI(app.UIAxesSpectogram,audioData,app.WindowLengthEd
itField.Value,app.WindowShiftEditField.Value,WindowType,samplingRate) % Takes
~10 seconds for a 10 second signal
        end
        % Button pushed function: PlotinTimeButton
        function PlotinTimeButtonPushed(app, event)
            axes = app.UIAxesTime;
            tone_duration = app.TdsEditField.Value;
            tone_pause = app.TrsEditField.Value;
            digits = app.EditField.Value;
            amplitude = app.AEditField.Value/2;
            dtmf_signal = app.DTMF_signal*amplitude;

            signal_duration = (tone_duration+tone_pause)*length(digits) -
tone_pause;
            t_values = linspace(0,signal_duration,length(dtmf_signal));

            plot(axes,t_values,dtmf_signal);
            xlabel(axes,'Time(s)','Fontsize',12);
            ylabel(axes,'Amplitude','Fontsize',12);
            title(axes,'Time Plot','Fontsize',14);
%             ylim(axes,[0 2000]);
%             xlim(axes,[0 signal_duration]);
%             yticks(axes,linspace(0, sampling_frequency, 11));
        end
    end
```

```
    % Component initialization

    methods (Access = private)

        % Create UIFigure and components

        function createComponents(app)

            % Create UIFigure and hide until all components are created

            app.UIFigure = uifigure('Visible', 'off');

            app.UIFigure.Position = [100 100 640 480];

            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxesSpectogram

            app.UIAxesSpectogram = uiaxes(app.UIFigure);

            title(app.UIAxesSpectogram, 'Title')

            xlabel(app.UIAxesSpectogram, 'X')

            ylabel(app.UIAxesSpectogram, 'Y')

            zlabel(app.UIAxesSpectogram, 'Z')

            app.UIAxesSpectogram.Position = [287 14 232 160];

            % Create UIAxesTime

            app.UIAxesTime = uiaxes(app.UIFigure);

            title(app.UIAxesTime, 'Title')

            xlabel(app.UIAxesTime, 'X')

            ylabel(app.UIAxesTime, 'Y')

            zlabel(app.UIAxesTime, 'Z')

            app.UIAxesTime.Position = [398 173 208 167];

            % Create PlotSpectogramButton

            app.PlotSpectogramButton = uibutton(app.UIFigure, 'push');

            app.PlotSpectogramButton.ButtonPushedFcn = createCallbackFcn(app,
@PlotSpectogramButtonPushed, true);

            app.PlotSpectogramButton.Position = [52 8 104 22];

            app.PlotSpectogramButton.Text = 'Plot Spectogram';

            % Create WindowLengthEditFieldLabel

            app.WindowLengthEditFieldLabel = uilabel(app.UIFigure);

            app.WindowLengthEditFieldLabel.HorizontalAlignment = 'right';

            app.WindowLengthEditFieldLabel.Position = [14 112 88 22];

            app.WindowLengthEditFieldLabel.Text = 'Window Length';

            % Create WindowLengthEditField

            app.WindowLengthEditField = uieditfield(app.UIFigure, 'numeric');
```

```
            app.WindowLengthEditField.Position = [116 112 106 22];
            app.WindowLengthEditField.Value = 512;
            % Create WindowShiftEditFieldLabel
            app.WindowShiftEditFieldLabel = uilabel(app.UIFigure);
            app.WindowShiftEditFieldLabel.HorizontalAlignment = 'right';
            app.WindowShiftEditFieldLabel.Position = [15 77 76 22];
            app.WindowShiftEditFieldLabel.Text = 'Window Shift';
            % Create WindowShiftEditField
            app.WindowShiftEditField = uieditfield(app.UIFigure, 'numeric');
            app.WindowShiftEditField.Position = [116 77 106 22];
            app.WindowShiftEditField.Value = 256;
            % Create WindowTypeDropDownLabel
            app.WindowTypeDropDownLabel = uilabel(app.UIFigure);
            app.WindowTypeDropDownLabel.HorizontalAlignment = 'right';
            app.WindowTypeDropDownLabel.Position = [16 42 77 22];
            app.WindowTypeDropDownLabel.Text = 'Window Type';
            % Create WindowTypeDropDown
            app.WindowTypeDropDown = uidropdown(app.UIFigure);
            app.WindowTypeDropDown.Items = {'Rectangular', 'Tukey',
'Hamming'};
            app.WindowTypeDropDown.Position = [119 42 100 22];
            app.WindowTypeDropDown.Value = 'Rectangular';
            % Create SamplingRateEditFieldLabel
            app.SamplingRateEditFieldLabel = uilabel(app.UIFigure);
            app.SamplingRateEditFieldLabel.HorizontalAlignment = 'right';
            app.SamplingRateEditFieldLabel.Position = [18 146 84 22];
            app.SamplingRateEditFieldLabel.Text = 'Sampling Rate';
            % Create SamplingRateEditField
            app.SamplingRateEditField = uieditfield(app.UIFigure, 'numeric');
            app.SamplingRateEditField.Position = [116 146 106 22];
            app.SamplingRateEditField.Value = 8000;
            % Create KeyPadPanel
            app.KeyPadPanel = uipanel(app.UIFigure);
            app.KeyPadPanel.Position = [18 203 260 221];
            % Create GridLayout
```

```
            app.GridLayout = uigridlayout(app.KeyPadPanel);
            app.GridLayout.ColumnWidth = {'1x', '1x', '1x', '1x'};
            app.GridLayout.RowHeight = {'1x', '1x', '1x', '1x'};
            % Create Button
            app.Button = uibutton(app.GridLayout, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed,
true);
            app.Button.Layout.Row = 1;
            app.Button.Layout.Column = 1;
            app.Button.Text = '1';
            % Create Button_2
            app.Button_2 = uibutton(app.GridLayout, 'push');
            app.Button_2.ButtonPushedFcn = createCallbackFcn(app,
@Button_2Pushed, true);
            app.Button_2.Layout.Row = 1;
            app.Button_2.Layout.Column = 2;
            app.Button_2.Text = '2';
            % Create Button_3
            app.Button_3 = uibutton(app.GridLayout, 'push');
            app.Button_3.ButtonPushedFcn = createCallbackFcn(app,
@Button_3Pushed, true);
            app.Button_3.Layout.Row = 1;
            app.Button_3.Layout.Column = 3;
            app.Button_3.Text = '3';
            % Create AButton
            app.AButton = uibutton(app.GridLayout, 'push');
            app.AButton.ButtonPushedFcn = createCallbackFcn(app,
@AButtonPushed, true);
            app.AButton.Layout.Row = 1;
            app.AButton.Layout.Column = 4;
            app.AButton.Text = 'A';
            % Create Button_5
            app.Button_5 = uibutton(app.GridLayout, 'push');
            app.Button_5.ButtonPushedFcn = createCallbackFcn(app,
@Button_5Pushed, true);
            app.Button_5.Layout.Row = 2;
            app.Button_5.Layout.Column = 1;
```

```
            app.Button_5.Text = '4';

            % Create Button_6

            app.Button_6 = uibutton(app.GridLayout, 'push');

            app.Button_6.ButtonPushedFcn = createCallbackFcn(app,
@Button_6Pushed, true);

            app.Button_6.Layout.Row = 2;

            app.Button_6.Layout.Column = 2;

            app.Button_6.Text = '5';

            % Create Button_7

            app.Button_7 = uibutton(app.GridLayout, 'push');

            app.Button_7.ButtonPushedFcn = createCallbackFcn(app,
@Button_7Pushed, true);

            app.Button_7.Layout.Row = 2;

            app.Button_7.Layout.Column = 3;

            app.Button_7.Text = '6';

            % Create BButton

            app.BButton = uibutton(app.GridLayout, 'push');

            app.BButton.ButtonPushedFcn = createCallbackFcn(app,
@BButtonPushed, true);

            app.BButton.Layout.Row = 2;

            app.BButton.Layout.Column = 4;

            app.BButton.Text = 'B';

            % Create Button_8

            app.Button_8 = uibutton(app.GridLayout, 'push');

            app.Button_8.ButtonPushedFcn = createCallbackFcn(app,
@Button_8Pushed, true);

            app.Button_8.Layout.Row = 3;

            app.Button_8.Layout.Column = 1;

            app.Button_8.Text = '7';

            % Create Button_9

            app.Button_9 = uibutton(app.GridLayout, 'push');

            app.Button_9.ButtonPushedFcn = createCallbackFcn(app,
@Button_9Pushed, true);

            app.Button_9.Layout.Row = 3;

            app.Button_9.Layout.Column = 2;

            app.Button_9.Text = '8';

            % Create Button_10
```

```
            app.Button_10 = uibutton(app.GridLayout, 'push');

            app.Button_10.ButtonPushedFcn = createCallbackFcn(app,
@Button_10Pushed, true);

            app.Button_10.Layout.Row = 3;

            app.Button_10.Layout.Column = 3;

            app.Button_10.Text = '9';

            % Create CButton

            app.CButton = uibutton(app.GridLayout, 'push');

            app.CButton.ButtonPushedFcn = createCallbackFcn(app,
@CButtonPushed, true);

            app.CButton.Layout.Row = 3;

            app.CButton.Layout.Column = 4;

            app.CButton.Text = 'C';

            % Create Button_11

            app.Button_11 = uibutton(app.GridLayout, 'push');

            app.Button_11.ButtonPushedFcn = createCallbackFcn(app,
@Button_11Pushed, true);

            app.Button_11.Layout.Row = 4;

            app.Button_11.Layout.Column = 1;

            app.Button_11.Text = '*';

            % Create Button_12

            app.Button_12 = uibutton(app.GridLayout, 'push');

            app.Button_12.ButtonPushedFcn = createCallbackFcn(app,
@Button_12Pushed, true);

            app.Button_12.Layout.Row = 4;

            app.Button_12.Layout.Column = 2;

            app.Button_12.Text = '0';

            % Create Button_13

            app.Button_13 = uibutton(app.GridLayout, 'push');

            app.Button_13.ButtonPushedFcn = createCallbackFcn(app,
@Button_13Pushed, true);

            app.Button_13.Layout.Row = 4;

            app.Button_13.Layout.Column = 3;

            app.Button_13.Text = '#';

            % Create DButton

            app.DButton = uibutton(app.GridLayout, 'push');
```

```
            app.DButton.ButtonPushedFcn = createCallbackFcn(app,
@DButtonPushed, true);

            app.DButton.Layout.Row = 4;

            app.DButton.Layout.Column = 4;

            app.DButton.Text = 'D';

            % Create RESETButton

            app.RESETButton = uibutton(app.UIFigure, 'push');

            app.RESETButton.ButtonPushedFcn = createCallbackFcn(app,
@RESETButtonPushed, true);

            app.RESETButton.Position = [491 414 62 35];

            app.RESETButton.Text = 'RESET';

            % Create TdsEditFieldLabel

            app.TdsEditFieldLabel = uilabel(app.UIFigure);

            app.TdsEditFieldLabel.HorizontalAlignment = 'right';

            app.TdsEditFieldLabel.Position = [285 427 36 22];

            app.TdsEditFieldLabel.Text = 'Td (s)';

            % Create TdsEditField

            app.TdsEditField = uieditfield(app.UIFigure, 'numeric');

            app.TdsEditField.Position = [336 427 63 22];

            % Create SpectogramParametersLabel

            app.SpectogramParametersLabel = uilabel(app.UIFigure);

            app.SpectogramParametersLabel.FontWeight = 'bold';

            app.SpectogramParametersLabel.Position = [54 173 144 22];

            app.SpectogramParametersLabel.Text = 'Spectogram Parameters';

            % Create SAVEButton

            app.SAVEButton = uibutton(app.UIFigure, 'push');

            app.SAVEButton.ButtonPushedFcn = createCallbackFcn(app,
@SAVEButtonPushed, true);

            app.SAVEButton.Position = [438 360 62 22];

            app.SAVEButton.Text = 'SAVE';

            % Create PLAYButton

            app.PLAYButton = uibutton(app.UIFigure, 'push');

            app.PLAYButton.ButtonPushedFcn = createCallbackFcn(app,
@PLAYButtonPushed, true);

            app.PLAYButton.Position = [544 360 62 22];

            app.PLAYButton.Text = 'PLAY';
```

```
            % Create ENCODEButton
            app.ENCODEButton = uibutton(app.UIFigure, 'push');
            app.ENCODEButton.ButtonPushedFcn = createCallbackFcn(app,
@ENCODEButtonPushed, true);
            app.ENCODEButton.Position = [326 305 62 35];
            app.ENCODEButton.Text = 'ENCODE';
            % Create EditField
            app.EditField = uieditfield(app.UIFigure, 'text');
            app.EditField.Position = [19 425 259 28];
            % Create TrsEditFieldLabel
            app.TrsEditFieldLabel = uilabel(app.UIFigure);
            app.TrsEditFieldLabel.HorizontalAlignment = 'right';
            app.TrsEditFieldLabel.Position = [287 393 34 22];
            app.TrsEditFieldLabel.Text = 'Tr (s)';
            % Create TrsEditField
            app.TrsEditField = uieditfield(app.UIFigure, 'numeric');
            app.TrsEditField.Position = [336 393 63 22];
            app.TrsEditField.Value = 0.5;
            % Create AEditFieldLabel
            app.AEditFieldLabel = uilabel(app.UIFigure);
            app.AEditFieldLabel.HorizontalAlignment = 'right';
            app.AEditFieldLabel.Position = [296 357 25 22];
            app.AEditFieldLabel.Text = 'A';
            % Create AEditField
            app.AEditField = uieditfield(app.UIFigure, 'numeric');
            app.AEditField.Position = [336 357 63 22];
            app.AEditField.Value = 0.5;
            % Create PlotinTimeButton
            app.PlotinTimeButton = uibutton(app.UIFigure, 'push');
            app.PlotinTimeButton.ButtonPushedFcn = createCallbackFcn(app,
@PlotinTimeButtonPushed, true);
            app.PlotinTimeButton.Position = [320 230 73 22];
            app.PlotinTimeButton.Text = 'Plot in Time';
            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
```

```
        end
    end
    % App creation and deletion
    methods (Access = public)
        % Construct app
        function app = TransmitterPanel
            % Create UIFigure and components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
%--------------------------------------------------------------------------
classdef ReceiverPanel < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        DECODEButton                matlab.ui.control.Button
        DecodedSignalTextArea       matlab.ui.control.TextArea
        DecodedSignalTextAreaLabel  matlab.ui.control.Label
        DecodingAlgorithmSwitch     matlab.ui.control.Switch
        DecodingAlgorithmSwitchLabel  matlab.ui.control.Label
        SamplingRateEditField       matlab.ui.control.NumericEditField
        SamplingRateEditFieldLabel  matlab.ui.control.Label
        WindowTypeDropDown          matlab.ui.control.DropDown
        WindowTypeDropDownLabel     matlab.ui.control.Label
```

```
        WindowShiftEditField          matlab.ui.control.NumericEditField
        WindowShiftEditFieldLabel     matlab.ui.control.Label
        WindowLengthEditField         matlab.ui.control.NumericEditField
        WindowLengthEditFieldLabel    matlab.ui.control.Label
        SpectogramParametersLabel     matlab.ui.control.Label
        PlotSpectogramButton          matlab.ui.control.Button
        PlotinTimeButton              matlab.ui.control.Button
        TrsEditFieldLabel             matlab.ui.control.Label
        TrsEditField                  matlab.ui.control.NumericEditField
        TdsEditField                  matlab.ui.control.NumericEditField
        TdsEditFieldLabel             matlab.ui.control.Label
        StopButton                    matlab.ui.control.Button
        StartButton                   matlab.ui.control.Button
        UIAxes                        matlab.ui.control.UIAxes
        UIAxesTime                    matlab.ui.control.UIAxes
    end

    properties (Access = public)
         % Description
        DTMF_signal % Description
       % Description
    end

    % Callbacks that handle component events
    methods (Access = private)
        % Callback function
        function RecordAudioButtonPushed(app, event)
            samplingRate=44100;
            recordingDuration=app.RecordingTimeEditField.Value;
            captureAudio(samplingRate,recordingDuration);
        end
        % Callback function
        function PlayButtonPushed(app, event)
            signal = app.DTMF_signal;
```

```
            % Specify the sampling rate (replace this with your actual
sampling rate)



            % Play the signal as a sound
            playAudio(signal, 44100);
        end
        % Button pushed function: StartButton
        function StartButtonPushed(app, event)
            Fs = 44100;
            pause(0.5);
            recObj = audiorecorder(Fs, 16, 1); % 16 bits, 1 channel

            record(recObj);
            disp('started recording')
            % Store the audiorecorder object in the app data
            app.DTMF_signal = recObj;
        end
        % Button pushed function: StopButton
        function StopButtonPushed(app, event)
            % Check if the recorder object exists
            if isfield(app, 'DTMF_signal') && isvalid(app.DTMF_signal)
                stop(app.DTMF_signal);

                % Get the recorded audio data

                disp('Recording stopped.');
            else

                disp('No recording in progress.');
            end
           audioData = getaudiodata(app.DTMF_signal)



            % sampling rate when an audio is recorded is taken as human
```

```
            % signal as default, so the Fs should be 44100
           audiowrite('LastSaved.wav', audioData, 44100);


        end
        % Button pushed function: PlotinTimeButton
        function PlotinTimeButtonPushed(app, event)
            axes = app.UIAxesTime;

            dtmf_signal = getaudiodata(app.DTMF_signal);



            %signal_duration = (tone_duration+tone_pause)*length(digits) -
tone_pause;
            time = (0:(length(dtmf_signal)-1)) / app.DTMF_signal.SampleRate;

            plot(axes,time,dtmf_signal);
            xlabel(axes,'Time(s)','Fontsize',12);
            ylabel(axes,'Amplitude','Fontsize',12);
            title(axes,'Time Plot','Fontsize',14);
        end
        % Value changed function: DecodingAlgorithmSwitch
        function DecodingAlgorithmSwitchValueChanged(app, event)

        end
        % Button pushed function: DECODEButton
        function DECODEButtonPushed(app, event)
            switchValue = app.DecodingAlgorithmSwitch.Value;
            %dtmf_signal = getaudiodata(app.DTMF_signal);
            [audioData,samplingRate] = audioread('LastSaved.wav');

            %disp(Fs);
            tone_duration = app.TdsEditField.Value;
            tone_pause = app.TrsEditField.Value;
            % Decode the audio signal based on the switch value
            if switchValue == "Spectrogram"
```

```
                decodedSignal =
dtmf_decoder_spectogram_GUI(audioData,samplingRate,tone_duration,tone_pause);
            else
                decodedSignal =
dtmf_goertzel_decoder_GUI(audioData,samplingRate,tone_duration,tone_pause);
            end
            %app.DTMF_signal=decodedSignal;
            % Display the decoded signal in the text area
            app.DecodedSignalTextArea.Value = char(decodedSignal);



        end
        % Button pushed function: PlotSpectogramButton
        function PlotSpectogramButtonPushed(app, event)

            %audioData = getaudiodata(app.DTMF_signal);
            [audioData,samplingRate]=audioread('LastSaved.wav');

            %samplingRate = app.SamplingRateEditField;

            selectedWindow = app.WindowTypeDropDown.Value;

            switch selectedWindow
                    case 'Rectangular'
                        WindowType = 'rectwin';
                    case 'Tukey'
                        WindowType = 'tukeywin';
                    case 'Hamming'
                        WindowType = 'hamming';
                    otherwise
                        WindowType = 'rectwin';
            end
            tone_duration = app.TdsEditField.Value;
            tone_pause = app.TrsEditField.Value;
```

```
generateSpectrogram_last_version_GUI(app.UIAxes,audioData,app.WindowLengthEdi
tField.Value,app.WindowShiftEditField.Value,WindowType,samplingRate,tone_dura
tion,tone_pause);

%
spectogram_plotter_generated_signals_vol2(audioData,app.WindowLengthEditField
.Value,app.WindowShiftEditField.Value,WindowType,samplingRate,tone_duration,t
one_pause)

            % Takes ~10 seconds for a 10 second signal

        end

        % Value changed function: WindowLengthEditField

        function WindowLengthEditFieldValueChanged(app, event)


        end

        % Value changed function: WindowShiftEditField

        function WindowShiftEditFieldValueChanged(app, event)




        end

    end

    % Component initialization

    methods (Access = private)

        % Create UIFigure and components

        function createComponents(app)

            % Create UIFigure and hide until all components are created

            app.UIFigure = uifigure('Visible', 'off');

            app.UIFigure.Position = [100 100 640 480];

            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxesTime

            app.UIAxesTime = uiaxes(app.UIFigure);

            title(app.UIAxesTime, 'Title')

            xlabel(app.UIAxesTime, 'X')

            ylabel(app.UIAxesTime, 'Y')

            zlabel(app.UIAxesTime, 'Z')

            app.UIAxesTime.Position = [394 262 208 167];

            % Create UIAxes

            app.UIAxes = uiaxes(app.UIFigure);
```

```
            title(app.UIAxes, 'Title')
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [312 31 300 185];
            % Create StartButton
            app.StartButton = uibutton(app.UIFigure, 'push');
            app.StartButton.ButtonPushedFcn = createCallbackFcn(app,
@StartButtonPushed, true);
            app.StartButton.Position = [74 417 100 23];
            app.StartButton.Text = 'Start';
            % Create StopButton
            app.StopButton = uibutton(app.UIFigure, 'push');
            app.StopButton.ButtonPushedFcn = createCallbackFcn(app,
@StopButtonPushed, true);
            app.StopButton.Position = [74 383 100 23];
            app.StopButton.Text = 'Stop';
            % Create TdsEditFieldLabel
            app.TdsEditFieldLabel = uilabel(app.UIFigure);
            app.TdsEditFieldLabel.HorizontalAlignment = 'right';
            app.TdsEditFieldLabel.Position = [84 334 36 22];
            app.TdsEditFieldLabel.Text = 'Td (s)';
            % Create TdsEditField
            app.TdsEditField = uieditfield(app.UIFigure, 'numeric');
            app.TdsEditField.Position = [135 334 29 21];
            % Create TrsEditField
            app.TrsEditField = uieditfield(app.UIFigure, 'numeric');
            app.TrsEditField.Position = [135 302 29 21];
            % Create TrsEditFieldLabel
            app.TrsEditFieldLabel = uilabel(app.UIFigure);
            app.TrsEditFieldLabel.HorizontalAlignment = 'right';
            app.TrsEditFieldLabel.Position = [87 302 33 22];
            app.TrsEditFieldLabel.Text = 'Tr (s)';
            % Create PlotinTimeButton
            app.PlotinTimeButton = uibutton(app.UIFigure, 'push');
```

```
            app.PlotinTimeButton.ButtonPushedFcn = createCallbackFcn(app,
@PlotinTimeButtonPushed, true);

            app.PlotinTimeButton.Position = [277 396 73 22];

            app.PlotinTimeButton.Text = 'Plot in Time';

            % Create PlotSpectogramButton

            app.PlotSpectogramButton = uibutton(app.UIFigure, 'push');

            app.PlotSpectogramButton.ButtonPushedFcn = createCallbackFcn(app,
@PlotSpectogramButtonPushed, true);

            app.PlotSpectogramButton.Position = [114 31 104 22];

            app.PlotSpectogramButton.Text = 'Plot Spectogram';

            % Create SpectogramParametersLabel

            app.SpectogramParametersLabel = uilabel(app.UIFigure);

            app.SpectogramParametersLabel.FontWeight = 'bold';

            app.SpectogramParametersLabel.Position = [116 196 144 22];

            app.SpectogramParametersLabel.Text = 'Spectogram Parameters';

            % Create WindowLengthEditFieldLabel

            app.WindowLengthEditFieldLabel = uilabel(app.UIFigure);

            app.WindowLengthEditFieldLabel.HorizontalAlignment = 'right';

            app.WindowLengthEditFieldLabel.Position = [76 135 88 22];

            app.WindowLengthEditFieldLabel.Text = 'Window Length';

            % Create WindowLengthEditField

            app.WindowLengthEditField = uieditfield(app.UIFigure, 'numeric');

            app.WindowLengthEditField.ValueChangedFcn = createCallbackFcn(app,
@WindowLengthEditFieldValueChanged, true);

            app.WindowLengthEditField.Position = [178 135 106 22];

            app.WindowLengthEditField.Value = 512;

            % Create WindowShiftEditFieldLabel

            app.WindowShiftEditFieldLabel = uilabel(app.UIFigure);

            app.WindowShiftEditFieldLabel.HorizontalAlignment = 'right';

            app.WindowShiftEditFieldLabel.Position = [77 100 76 22];

            app.WindowShiftEditFieldLabel.Text = 'Window Shift';

            % Create WindowShiftEditField

            app.WindowShiftEditField = uieditfield(app.UIFigure, 'numeric');

            app.WindowShiftEditField.ValueChangedFcn = createCallbackFcn(app,
@WindowShiftEditFieldValueChanged, true);

            app.WindowShiftEditField.Position = [178 100 106 22];
```

```
            app.WindowShiftEditField.Value = 256;
            % Create WindowTypeDropDownLabel
            app.WindowTypeDropDownLabel = uilabel(app.UIFigure);
            app.WindowTypeDropDownLabel.HorizontalAlignment = 'right';
            app.WindowTypeDropDownLabel.Position = [78 65 77 22];
            app.WindowTypeDropDownLabel.Text = 'Window Type';
            % Create WindowTypeDropDown
            app.WindowTypeDropDown = uidropdown(app.UIFigure);
            app.WindowTypeDropDown.Items = {'Rectangular', 'Tukey',
'Hamming'};
            app.WindowTypeDropDown.Position = [181 65 100 22];
            app.WindowTypeDropDown.Value = 'Rectangular';
            % Create SamplingRateEditFieldLabel
            app.SamplingRateEditFieldLabel = uilabel(app.UIFigure);
            app.SamplingRateEditFieldLabel.HorizontalAlignment = 'right';
            app.SamplingRateEditFieldLabel.Position = [80 169 84 22];
            app.SamplingRateEditFieldLabel.Text = 'Sampling Rate';
            % Create SamplingRateEditField
            app.SamplingRateEditField = uieditfield(app.UIFigure, 'numeric');
            app.SamplingRateEditField.Position = [178 169 106 22];
            app.SamplingRateEditField.Value = 8000;
            % Create DecodingAlgorithmSwitchLabel
            app.DecodingAlgorithmSwitchLabel = uilabel(app.UIFigure);
            app.DecodingAlgorithmSwitchLabel.HorizontalAlignment = 'center';
            app.DecodingAlgorithmSwitchLabel.Position = [244 302 109 22];
            app.DecodingAlgorithmSwitchLabel.Text = 'Decoding Algorithm';
            % Create DecodingAlgorithmSwitch
            app.DecodingAlgorithmSwitch = uiswitch(app.UIFigure, 'slider');
            app.DecodingAlgorithmSwitch.Items = {'Spectrogram', 'Goertzel'};
            app.DecodingAlgorithmSwitch.ValueChangedFcn =
createCallbackFcn(app, @DecodingAlgorithmSwitchValueChanged, true);
            app.DecodingAlgorithmSwitch.Position = [277 339 44 19];
            app.DecodingAlgorithmSwitch.Value = 'Spectrogram';
            % Create DecodedSignalTextAreaLabel
            app.DecodedSignalTextAreaLabel = uilabel(app.UIFigure);
```

```
            app.DecodedSignalTextAreaLabel.HorizontalAlignment = 'right';
            app.DecodedSignalTextAreaLabel.Position = [198 249 90 22];
            app.DecodedSignalTextAreaLabel.Text = 'Decoded Signal';
            % Create DecodedSignalTextArea
            app.DecodedSignalTextArea = uitextarea(app.UIFigure);
            app.DecodedSignalTextArea.Position = [303 249 97 21];
            % Create DECODEButton
            app.DECODEButton = uibutton(app.UIFigure, 'push');
            app.DECODEButton.ButtonPushedFcn = createCallbackFcn(app,
@DECODEButtonPushed, true);
            app.DECODEButton.Position = [64 247 100 23];
            app.DECODEButton.Text = 'DECODE';
            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
    % App creation and deletion
    methods (Access = public)
        % Construct app
        function app = ReceiverPanel
            % Create UIFigure and components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```