Yunus Emre Tüysüz 2444032
Ata Oğuz Tanrıkulu 2443927

Date: 5.1.2023

EE430: Term Project Part 2

1. Introduction:

In this project, our focus is on the generation, encoding, and decoding of Dual-Tone Multi-Frequency (DTMF) signals. The project is structured into two primary sections: the transmitter and the receiver. The transmitter's role involves accepting user input in the form of dialled numbers, generating the corresponding DTMF signal, and then performing several functions: visualizing the signal in both time domain and spectrogram, playing the audio signal for verification, and storing the signal as an audio file. On the receiver's end, the system is designed to either record a DTMF audio signal using a microphone or to accept an uploaded audio file. It then plots the signal's time domain and spectrogram, decodes the signal using two distinct methods, and allows for adjustments in the decoding parameters. Detailed explanations of the application specifications, the algorithms employed, and various real-time application tests and results are provided throughout this report.

2. MATLAB Application

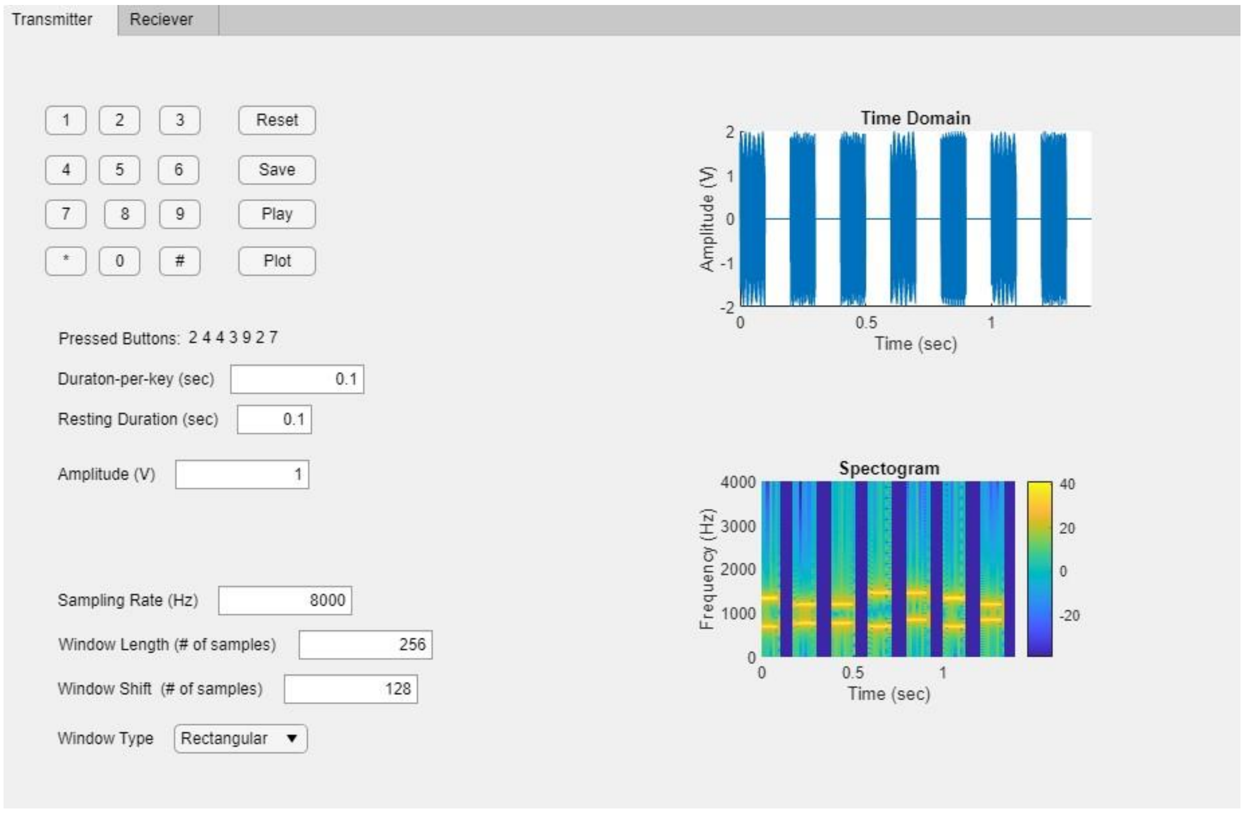
In this part, app specifications are explained in detail.

2.1. Transmitter

Transmitter is responsible from getting input from the user as dialled numbers, generating the DTMF signal, playing it if it is required, saving the audio file and lastly plotting the signal both in time domain and spectrogram. To get the user's dialled numbers, 12 buttons are used each corresponding one of the keypads. Every time user presses one of the buttons, it will be displayed on the screen. User can also reset his/her input by pressing "Reset" button. After that, all of the previous input will be cleared, and user can start over.

After he/she dial all the numbers that he/she wants, user can adjust the signal duration per key, resting duration and amplitude of the signal, and also, he/she can press the "Plot" button to display time domain version of the signal and its spectrogram. User also can press "Play" button to play the audio file and he/she can press "Save" button to save the audio file as the "dialing.wav" to the current directory of the Matlab.

User also can adjust the spectrogram parameters like window length, shift and type, and he/she can set the sampling rate. Application screen is shown in Figure-1.



Pressed Buttons: 2 4 4 3 9 2 7

Duraton-per-key (sec) 0.1

Resting Duration (sec) 0.1

Amplitude (V) 1

Sampling Rate (Hz) 8000

Window Length (# of samples) 256

Window Shift (# of samples) 128

Window Type Rectangular

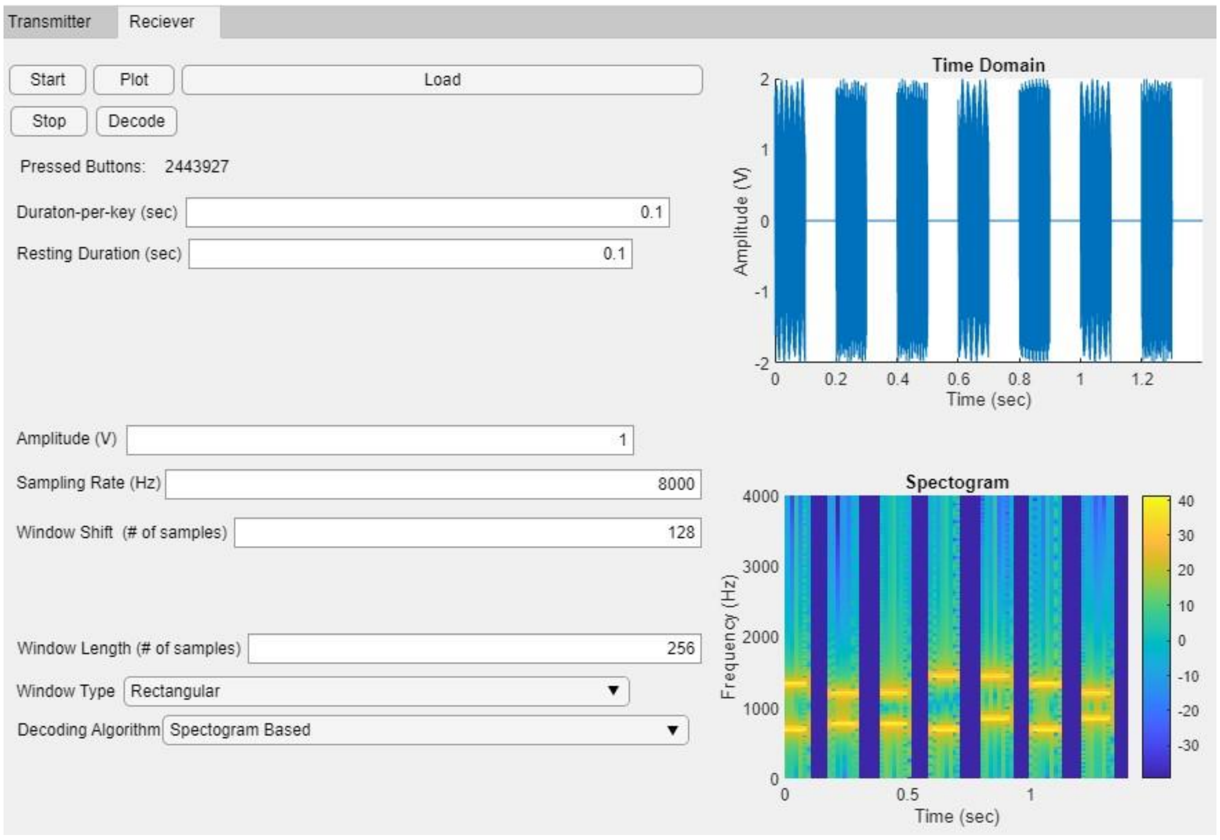
Figure 1: Transmitter Page

2.2. Receiver

Receiver is responsible from retrieving the DTMF signal and decoding it as dialled numbers. User can use a microphone to record and audio signal as input or he/she can upload an audio file from his/her device. To record an audio file, user should press “Start” button to start the recording and press “Stop” button to stop it. He/she can press “Load” button to upload an audio file.

After that user can press the “Plot” button to visualize the time domain version of the input signal and its spectrogram. User can adjust spectrogram parameters like window length, shift and type, and he/she can set the sampling rate, both for spectrogram plotting and for spectrogram-based decoding of the signal. User can adjust the signal duration per key, resting duration and amplitude of the signal for Goertzel Algorithm decoding method.

User can specify which decoding method that he/she would use and decode the input signal by pressing the “Decode” button. Application screen is shown in Figure-2.



Pressed Buttons: 2443927

Duraton-per-key (sec) 0.1

Resting Duration (sec) 0.1

Amplitude (V) 1

Sampling Rate (Hz) 8000

Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Spectogram Based

Figure 2: Receiver Page

3. Decoding Algorithms

There are two choices for the user to decode the input signal in the Receiver end: Spectrogram-based and Goertzel Algorithm. In this part, specifications of these algorithms are explained in detail.

3.1. Spectrogram-Based Decoding Algorithm

Spectrogram based decoding uses the spectrogram of the input signal. Generation of the spectrogram is made by our own algorithm which is explained in the first part of the project. It will take inputs as the window shift, length and type and also the sampling rate.

The decoding algorithm operates on the generated spectrogram by segmenting the spectrogram matrix into smaller matrices. This segmentation is based on periods of silence, identified as moments when the signal's power drops to zero across all frequencies for a certain duration. For each of these segmented matrices, the algorithm then identifies the frequency component with the highest power in two separate ranges: below and above 1 kHz. These peak frequency values are recorded.

Subsequently, the algorithm compares the lower frequency peak with the known low-frequency tones of DTMF signals, selecting the closest match as the identified low frequency. This process is similarly applied to the higher frequency peak. Once both the low and high frequency components are determined, the algorithm maps these frequencies to their corresponding dialled number.

This procedure is repeated for each segmented matrix. The decoded digits are compiled into a list, and upon completion of the process, this list of decoded numbers is displayed on the screen.

3.2. Goertzel Algorithm

The Goertzel algorithm is a digital signal processing technique that provides an efficient way to calculate the discrete Fourier transform (DFT) at specific frequencies. It's particularly useful for detecting predefined frequencies in a signal, which is why it's often used in applications like DTMF (Dual-Tone Multi-Frequency) tone decoding in telephone systems. And in this report we implement Goertzel algorithm for receiver part. We will explain the MATLAB function to decode DTMF signals using the Goertzel algorithm.

Here's a breakdown of how the provided code works:

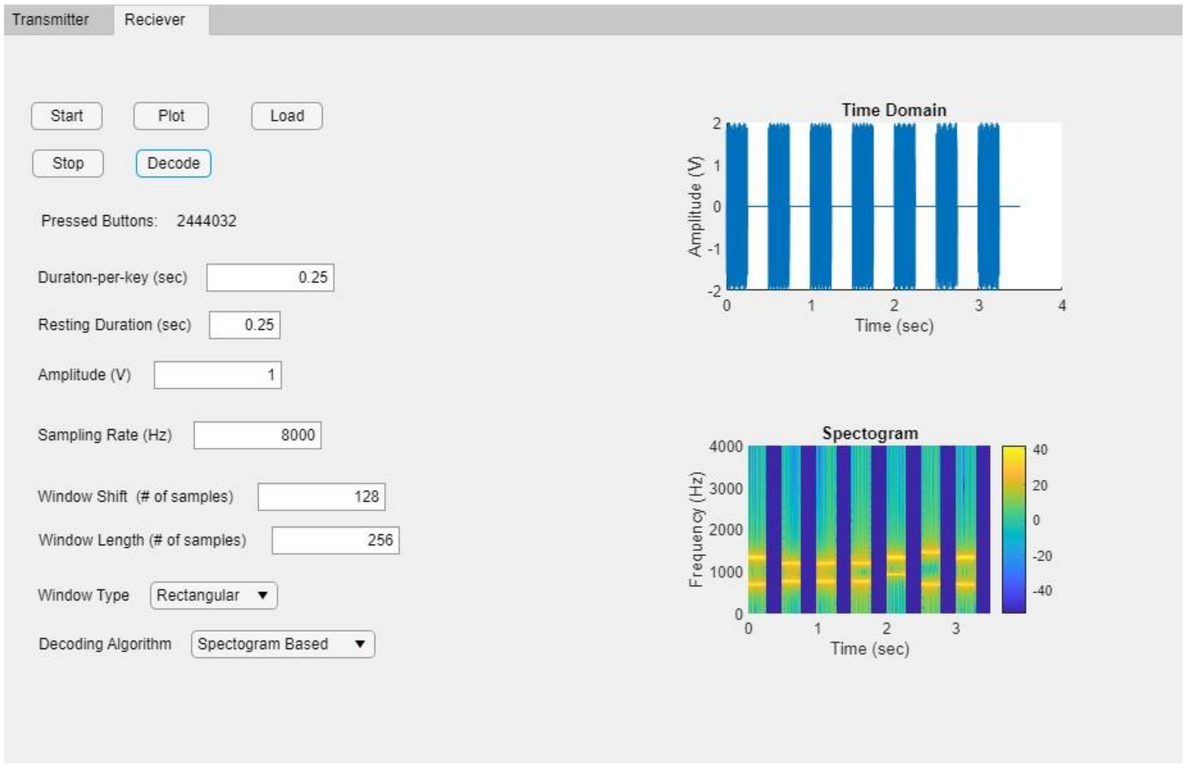
The function takes several parameters, including the input signal, the sampling frequency, the duration per key press, resting duration between key presses, minimum amplitude for detection, window shift and length, and the type of window to use (like Rectangular or Hamming). Then we define DTMF Frequencies: DTMF tones are composed of two frequencies: one from a low-frequency group (697, 770, 852, 941 Hz) and one from a high-frequency group (1209, 1336, 1477 Hz). These frequencies correspond to the keys on a telephone keypad. The signal is sampled according to the specified durations and a window function is applied. Windowing helps mitigate spectral leakage in the frequency analysis. The main part of the function involves applying the Goertzel algorithm to these windowed segments of the signal. The Goertzel algorithm computes the strength (magnitude) of each DTMF frequency in the segment. It uses an iterative approach with cosine functions to compute the DFT selectively for that frequency as we discussed in class. In more detailed way, the Goertzel function is an implementation of the Goertzel algorithm, designed to detect a specific frequency within a signal. The function iteratively updates two variables (prev1 and prev2) based on the current signal value and a cosine term related to the target frequency bin (binIndex). After iterating through the signal segment, it calculates the magnitude of the target frequency component using these updated values. Then the algorithm identifies the strongest frequencies in the low and high groups. If these frequencies exceed a specified amplitude threshold, the corresponding key is decoded using the frequency-to-key mapping defined earlier. After decoding a key, the algorithm skips over the resting period to the start of the next keypress according to window shift.

In summary, this code is a specialized implementation of the Goertzel algorithm tailored for decoding DTMF tones from a signal. It processes segments of the input signal to detect the presence of specific frequencies and decodes these frequencies into corresponding digits based on the standard DTMF keypad layout.

4. Test Results

Both decoding algorithms are tested with diverse types of inputs where per-key and resting durations are altered.

Test results are shown in the Figure 3,4,5,6,7,8 where decoded signal and decoding parameters are shown. For each case, dialled input signal is “2444032” and input signal taken as a loaded audio file.



Pressed Buttons: 2444032

Duraton-per-key (sec) 0.25

Resting Duration (sec) 0.25

Amplitude (V) 1

Sampling Rate (Hz) 8000

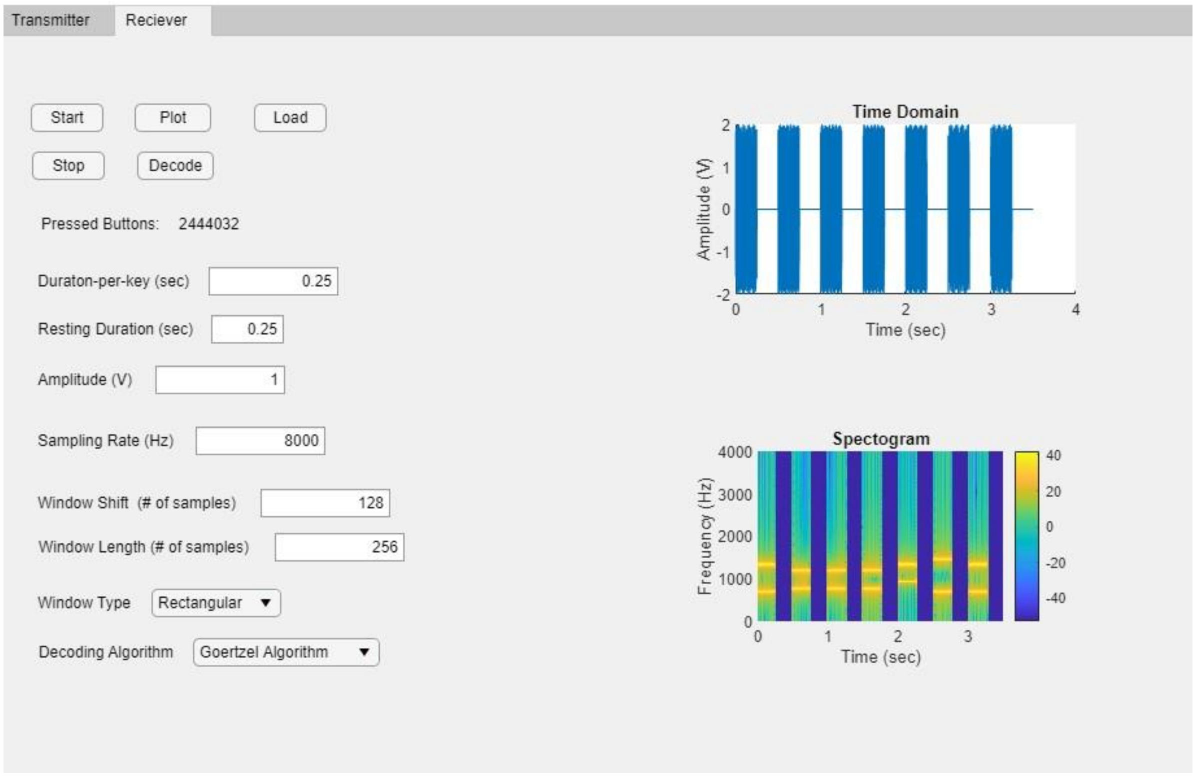
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Spectogram Based

Figure 3: $(T_d, T_r) = (250\ ms, 250\ ms)$*, Spectrogram-Based*



Pressed Buttons: 2444032

Duraton-per-key (sec) 0.25

Resting Duration (sec) 0.25

Amplitude (V) 1

Sampling Rate (Hz) 8000

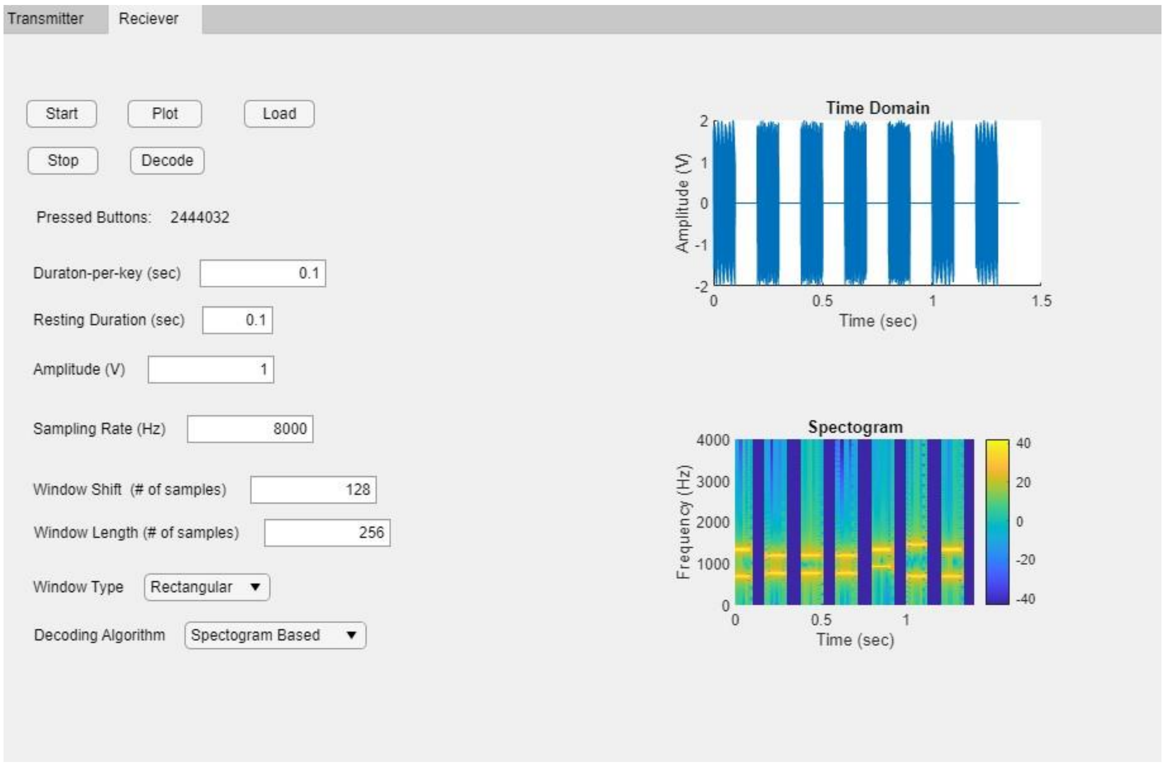
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Goertzel Algorithm

Figure 4: $(T_d, T_r) = (250\ ms, 250\ ms)$*, Goertzel*



Pressed Buttons: 2444032

Duraton-per-key (sec) 0.1

Resting Duration (sec) 0.1

Amplitude (V) 1

Sampling Rate (Hz) 8000

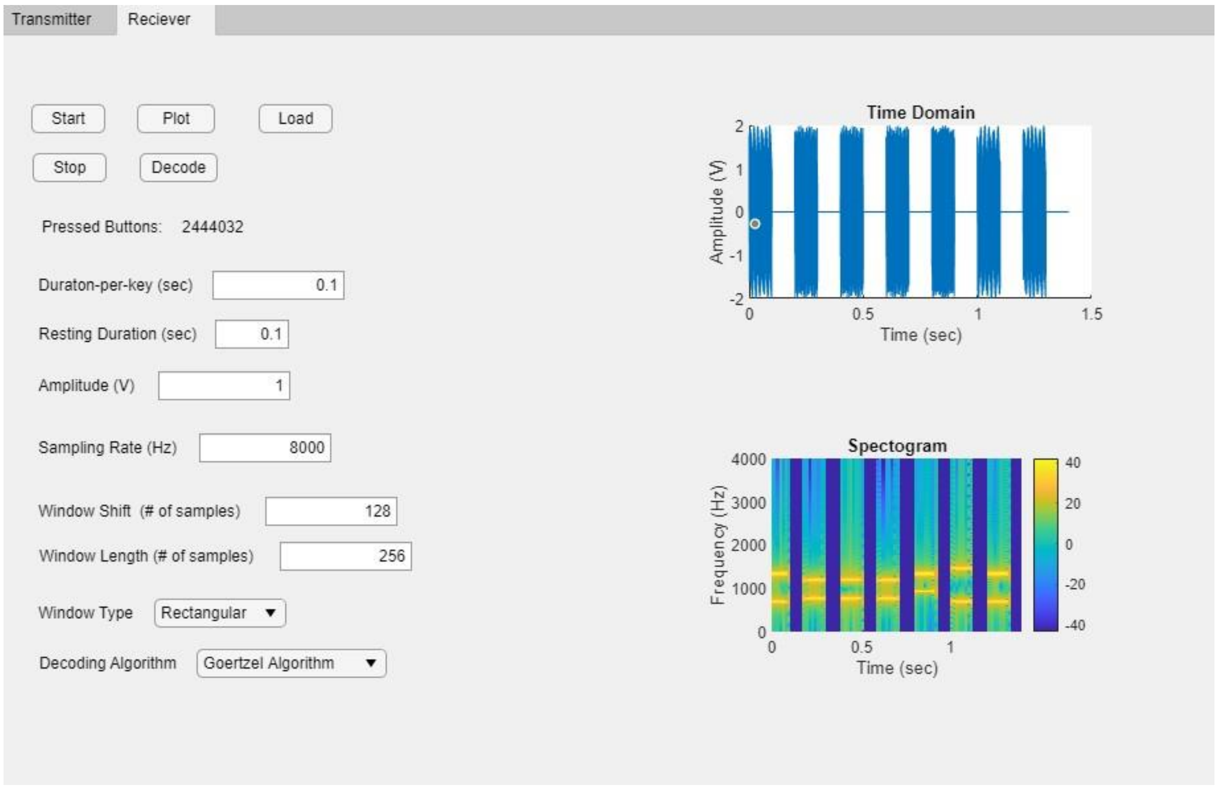
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Spectogram Based

Figure 5: $(T_d, T_r) = (100\ ms, 100\ ms)$*, Spectrogram-Based*



Pressed Buttons: 2444032

Duraton-per-key (sec) 0.1

Resting Duration (sec) 0.1

Amplitude (V) 1

Sampling Rate (Hz) 8000

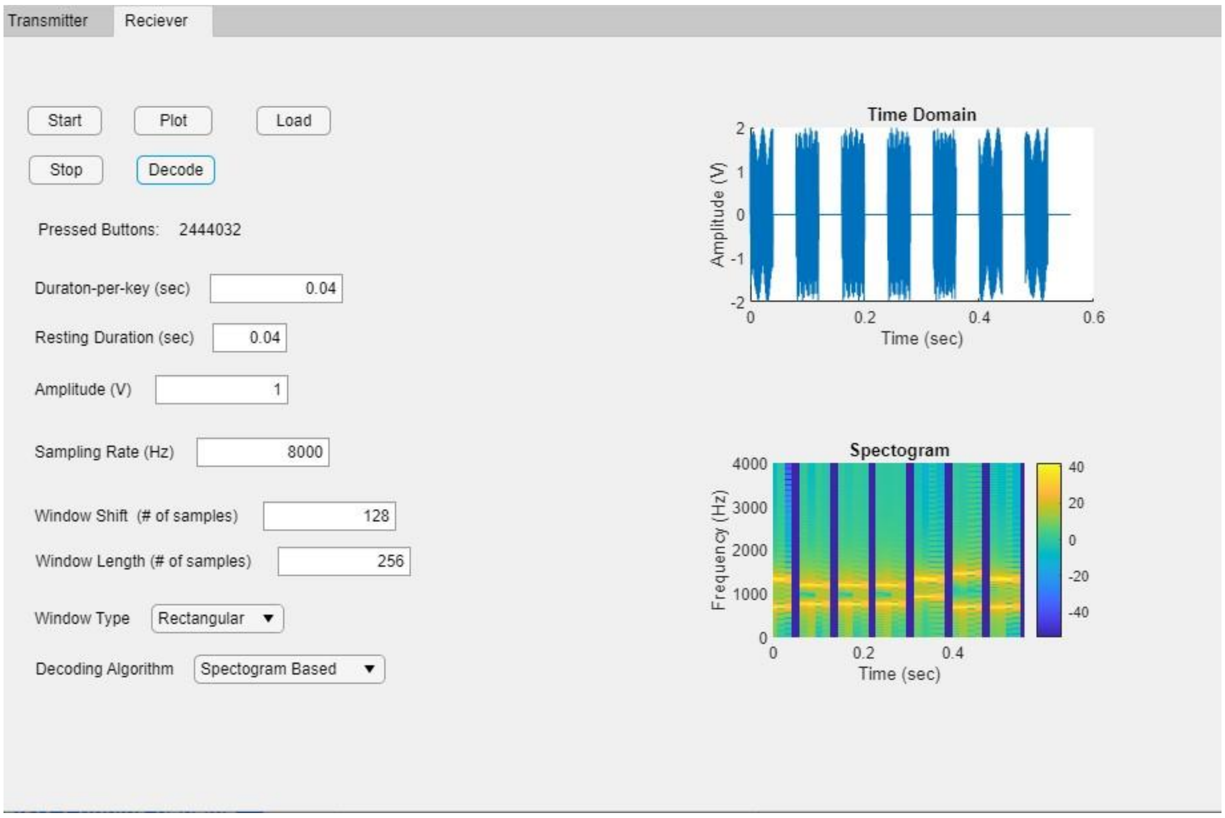
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Goertzel Algorithm

Figure 6: $(T_d, T_r) = (100\, ms, 100\, ms)$*, Goertzel*



Pressed Buttons: 2444032

Duraton-per-key (sec) 0.04

Resting Duration (sec) 0.04

Amplitude (V) 1

Sampling Rate (Hz) 8000

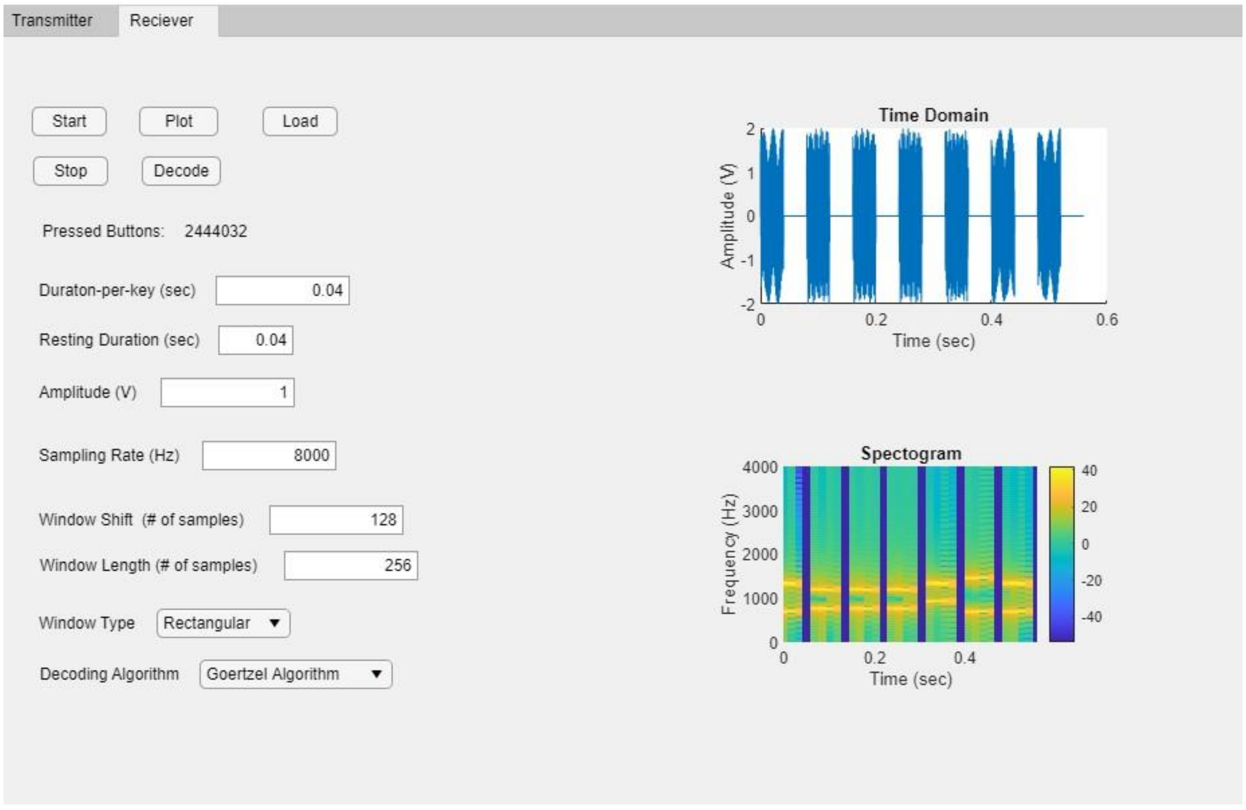
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Spectogram Based

Figure 7: $(T_d, T_r) = (40\, ms, 40\, ms)$*, Spectrogram-Based*



Pressed Buttons: 2444032

Duraton-per-key (sec) 0.04

Resting Duration (sec) 0.04

Amplitude (V) 1

Sampling Rate (Hz) 8000

Window Shift (# of samples) 128

Window Length (# of samples) 256

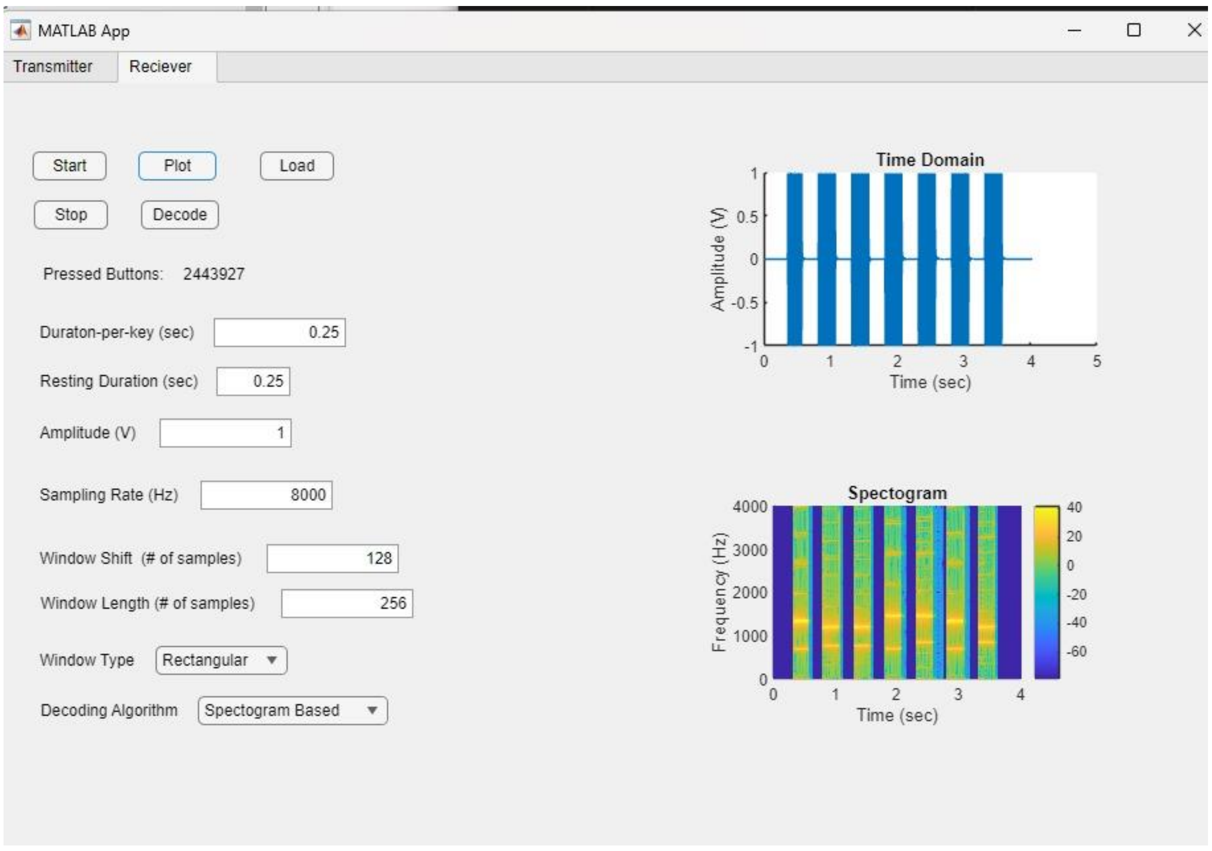
Window Type Rectangular

Decoding Algorithm Goertzel Algorithm

Figure 8: $(T_d, T_r) = (40\ ms, 40\ ms)$*, Goertzel*

As it can be seen from these results that both of the decoding algorithms are working error-free with the given signal and resting duration times.

For dialled input signal “2443927”, we tried our function by using microphone. Test results are shown in the Figure 9,10,11,12,13,14 where decoded signal and decoding parameters are shown.



Pressed Buttons: 2443927

Duraton-per-key (sec) 0.25

Resting Duration (sec) 0.25

Amplitude (V) 1

Sampling Rate (Hz) 8000

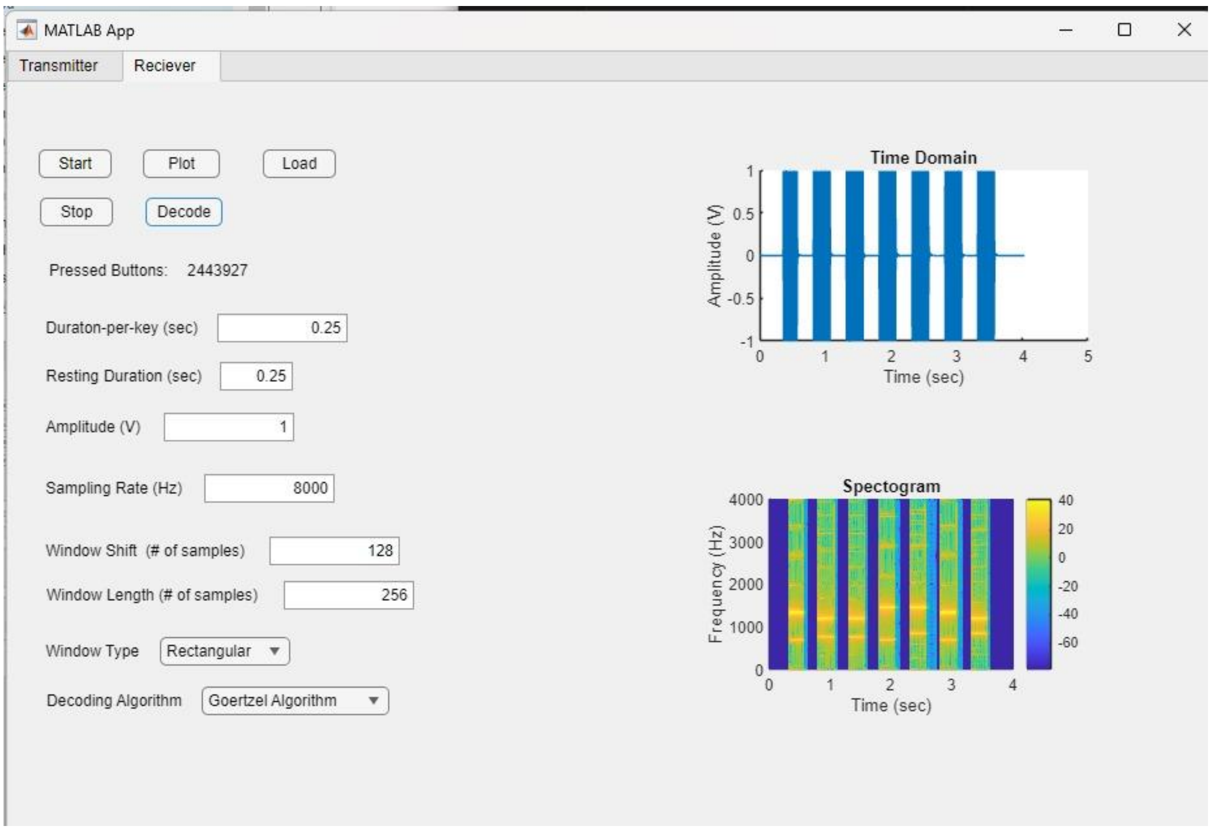
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Spectogram Based

Figure 9: $(T_d, T_r) = (250\ ms, 250\ ms)$, *Spectrogram-Based*



Pressed Buttons: 2443927

Duraton-per-key (sec) 0.25

Resting Duration (sec) 0.25

Amplitude (V) 1

Sampling Rate (Hz) 8000

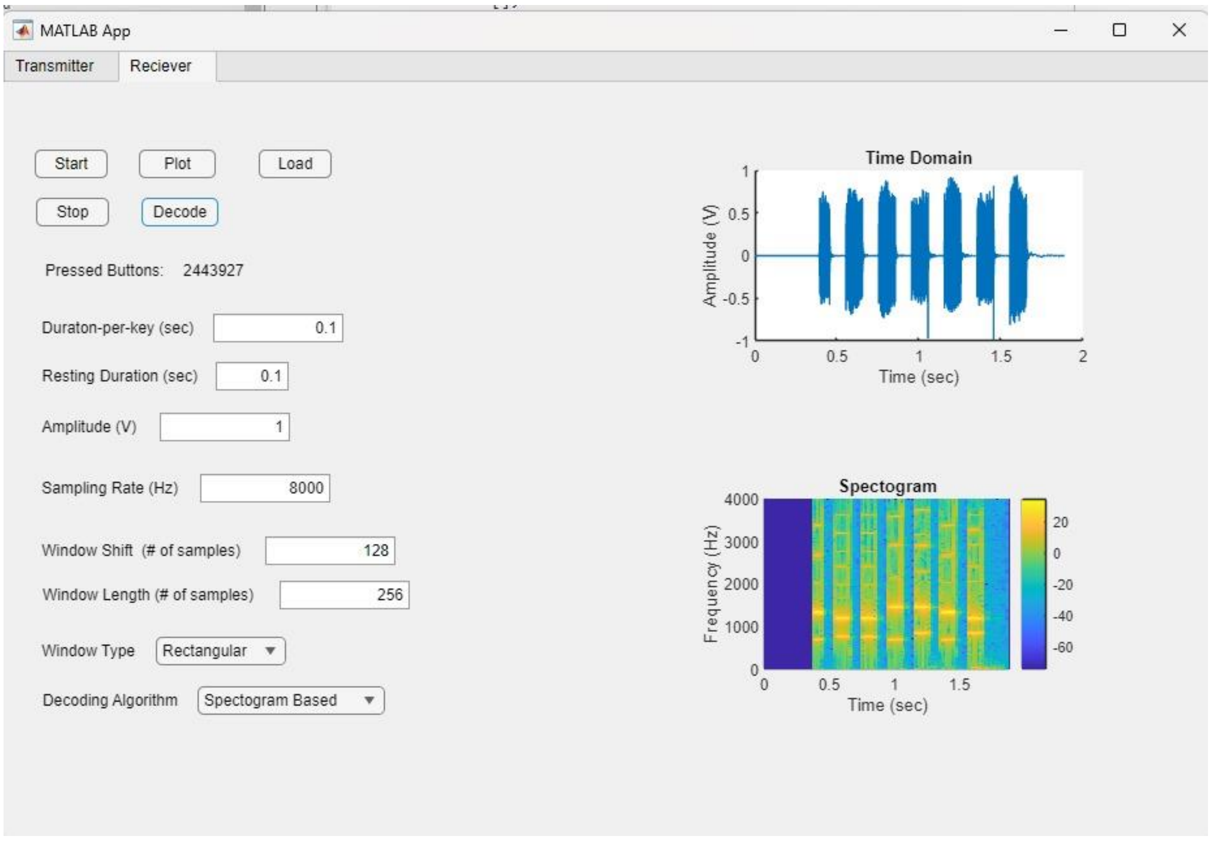
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Goertzel Algorithm

Figure 10: $(T_d, T_r) = (250\ ms, 250\ ms)$, *Goertzel*


MATLAB App

Transmitter | Reciever

Start | Plot | Load | Stop | Decode

Pressed Buttons: 2443927

Duraton-per-key (sec) 0.1

Resting Duration (sec) 0.1

Amplitude (V) 1

Sampling Rate (Hz) 8000

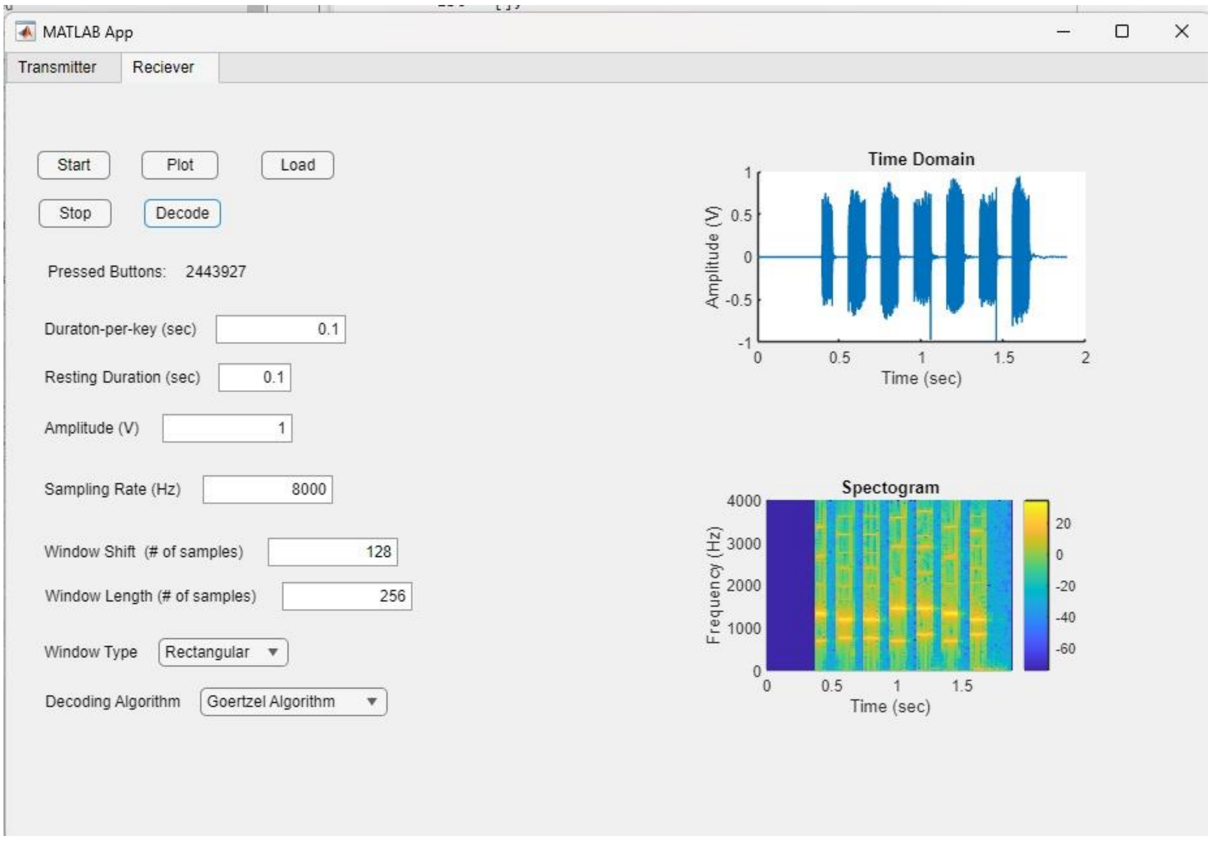
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type: Rectangular

Decoding Algorithm: Spectogram Based

Figure 11: $(T_d, T_r) = (100\ ms, 100\ ms)$, *Spectrogram-Based*


MATLAB App

Transmitter | Reciever

Start | Plot | Load | Stop | Decode

Pressed Buttons: 2443927

Duraton-per-key (sec) 0.1

Resting Duration (sec) 0.1

Amplitude (V) 1

Sampling Rate (Hz) 8000

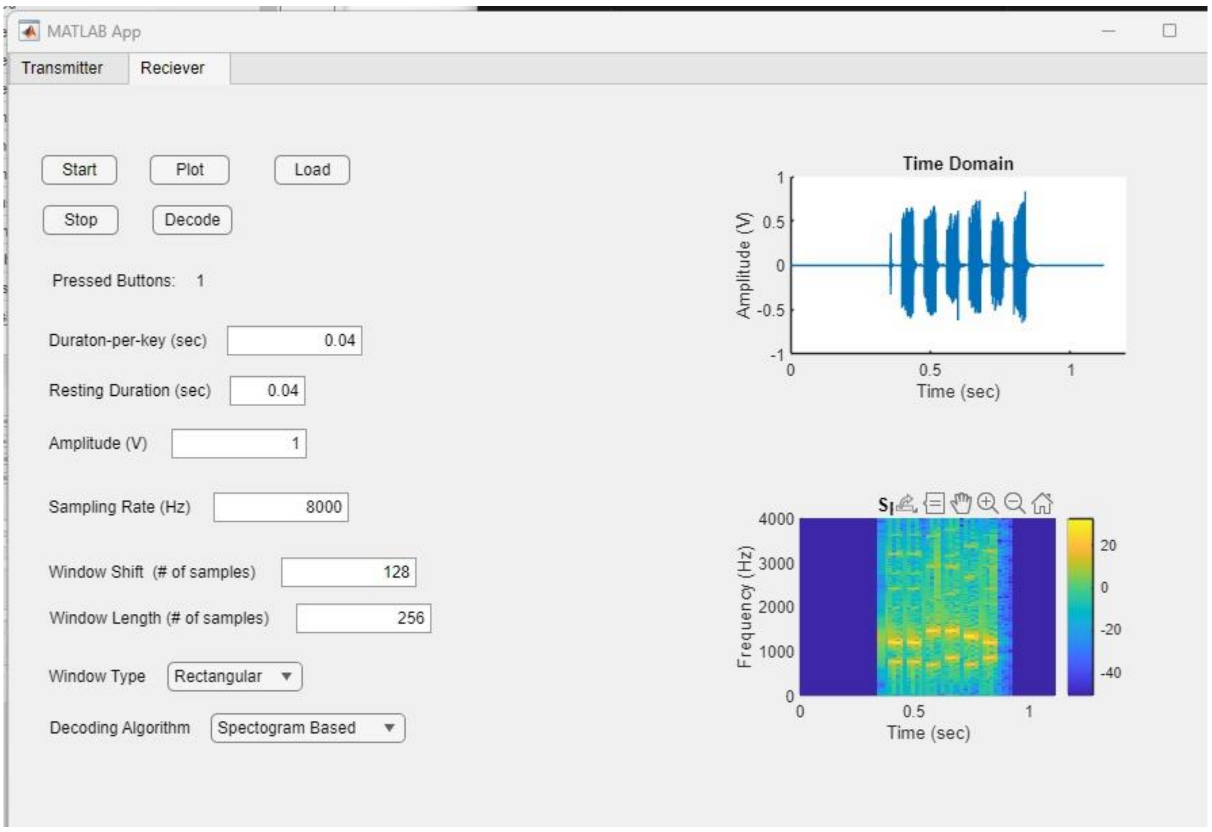
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type: Rectangular

Decoding Algorithm: Goertzel Algorithm

Figure 12: $(T_d, T_r) = (100\ ms, 100\ ms)$, *Goertzel*



MATLAB App

Transmitter | Reciever

Start | Plot | Load

Stop | Decode

Pressed Buttons: 1

Duraton-per-key (sec) 0.04

Resting Duration (sec) 0.04

Amplitude (V) 1

Sampling Rate (Hz) 8000

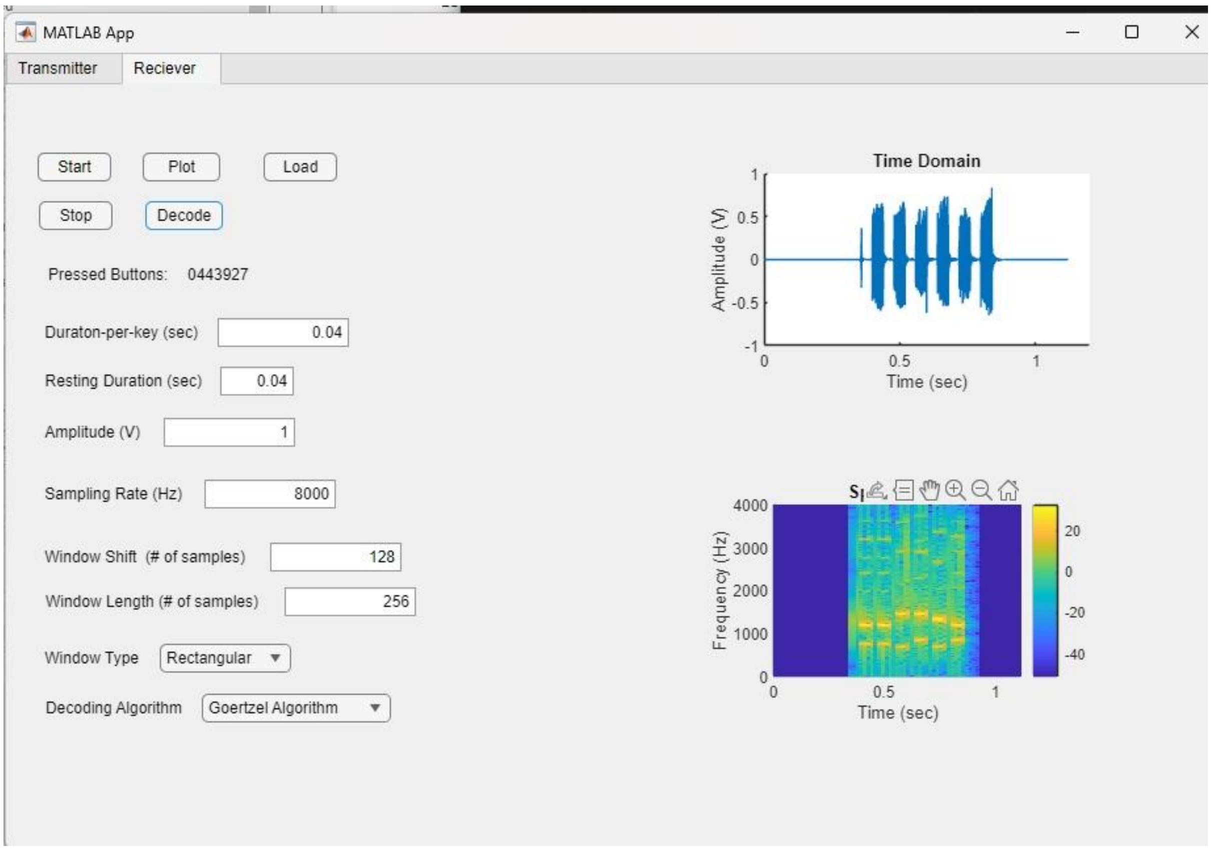
Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Spectogram Based

Figure 13: $(T_d, T_r) = (40\ ms, 40\ ms)$*, Spectrogram-Based*



MATLAB App

Transmitter | Reciever

Start | Plot | Load

Stop | Decode

Pressed Buttons: 0443927

Duraton-per-key (sec) 0.04

Resting Duration (sec) 0.04

Amplitude (V) 1

Sampling Rate (Hz) 8000

Window Shift (# of samples) 128

Window Length (# of samples) 256

Window Type Rectangular

Decoding Algorithm Goertzel Algorithm

Figure 14: $(T_d, T_r) = (40\ ms, 40\ ms)$*, Goertzel*

From these results, we can say that Spectrogram-based and Goertzel algorithm works properly for recorded input when per-key and resting duration is higher than 40 ms. However, when these durations become 40 ms, spectrogram-based algorithm misses the signal and fails to decode where Goertzel algorithm only misses first digit of the input and rest of the input is decoded properly. In conclusion, both algorithm works well for high resting and per-key durations but we start to see errors when these durations are shorter and in that case, Goertzel algorithm works better.

5. Conclusion

In conclusion, this project successfully implemented a dual-tone multi-frequency (DTMF) signalling system using MATLAB. The creation of a user-friendly graphical interface facilitated the transmission and reception of DTMF signals. Our programs effectively encoded and decoded these signals using two different approaches Spectrogram-Based and Goertzel algorithms. The project's experimental phase yielded valuable insights, particularly regarding algorithmic performance under various settings. While challenges were encountered such as when input is noisy the output is not in excepted format. But when we set input as a file, we saw that our algorithms work properly also if the microphone isolated well, we got good results.

6. Appendix

Matlab code:

```
classdef dsp2 < matlab.apps.AppBase


% Properties that correspond to app components
properties (Access = public)
UIFigure matlab.ui.Figure
TabGroup matlab.ui.container.TabGroup
TransmitterTab matlab.ui.container.Tab
AmplitudeVEditField matlab.ui.control.NumericEditField
AmplitudeVEditField_3Label matlab.ui.control.Label
SamplingRateHzEditField matlab.ui.control.NumericEditField
SamplingRateHzEditField_3Label matlab.ui.control.Label
WindowLengthofsamplesEditField matlab.ui.control.NumericEditField
WindowLengthofsamplesEditField_3Label matlab.ui.control.Label
DuratonperkeysecEditField matlab.ui.control.NumericEditField
DuratonperkeysecEditField_3Label matlab.ui.control.Label
WindowTypeDropDown matlab.ui.control.DropDown
WindowTypeDropDownLabel matlab.ui.control.Label
WindowShiftofsamplesEditField matlab.ui.control.NumericEditField
WindowShiftofsamplesEditFieldLabel matlab.ui.control.Label
PlotButton matlab.ui.control.Button
RestingDurationsecEditField matlab.ui.control.NumericEditField
RestingDurationsecEditFieldLabel matlab.ui.control.Label
PlayButton matlab.ui.control.Button
```

```
SaveButton matlab.ui.control.Button
ResetButton matlab.ui.control.Button
Label matlab.ui.control.Label
PressedButtonsLabel matlab.ui.control.Label
Button_12 matlab.ui.control.Button
Button_11 matlab.ui.control.Button
Button_10 matlab.ui.control.Button
Button_9 matlab.ui.control.Button
Button_8 matlab.ui.control.Button
Button_7 matlab.ui.control.Button
Button_6 matlab.ui.control.Button
Button_5 matlab.ui.control.Button
Button_4 matlab.ui.control.Button
Button_3 matlab.ui.control.Button
Button_2 matlab.ui.control.Button
Button matlab.ui.control.Button
UIAxes2 matlab.ui.control.UIAxes
UIAxes matlab.ui.control.UIAxes
RecieverTab matlab.ui.container.Tab
LoadButton matlab.ui.control.Button
WindowLengthofsamplesEditField_2 matlab.ui.control.NumericEditField
WindowLengthofsamplesEditField_2Label matlab.ui.control.Label
DuratonperkeysecEditField_2 matlab.ui.control.NumericEditField
DuratonperkeysecEditField_2Label matlab.ui.control.Label
DecodeButton matlab.ui.control.Button
PlotButton_2 matlab.ui.control.Button
DecodingAlgorithmDropDown matlab.ui.control.DropDown
DecodingAlgorithmDropDownLabel matlab.ui.control.Label
Label_2 matlab.ui.control.Label
WindowTypeDropDown_2 matlab.ui.control.DropDown
WindowTypeDropDown_2Label matlab.ui.control.Label
WindowShiftofsamplesEditField_2 matlab.ui.control.NumericEditField
WindowShiftofsamplesEditField_2Label matlab.ui.control.Label
SamplingRateHzEditField_2 matlab.ui.control.NumericEditField
SamplingRateHzEditField_2Label matlab.ui.control.Label
AmplitudeVEditField_2 matlab.ui.control.NumericEditField
AmplitudeVEditField_2Label matlab.ui.control.Label
RestingDurationsecEditField_2 matlab.ui.control.NumericEditField
RestingDurationsecEditField_2Label matlab.ui.control.Label
PressedButtonsLabel_2 matlab.ui.control.Label
StopButton matlab.ui.control.Button
StartButton matlab.ui.control.Button
UIAxes2_2 matlab.ui.control.UIAxes
UIAxes_2 matlab.ui.control.UIAxes
end


properties (Access = private)
lst = [];
Recorder;
loadedAudioFile;
end
methods (Access = private)
```

```
function [S,T,F] = spec(~, signal, winSize, winShift, winType,samplingRate,
signalDuration)
noverlap=winSize-winShift;
totalSamples = floor((signalDuration) * samplingRate + 1);
% Create the window based on the specified type
if strcmp(winType, "Rectangular")
windowVec = transpose(rectwin(winSize));
end
if strcmp(winType, "Hamming")
windowVec = transpose(hamming(winSize));
end
numWindows = floor((length(signal) - noverlap) / winShift);


freqVectorLength = ceil((winSize + 1) / 2);
T = linspace(0,(totalSamples-1)/samplingRate,numWindows);
F = ((0:freqVectorLength-1)/winSize)*samplingRate;
stftMatrix = zeros(numWindows, freqVectorLength);
segmentStart = 1;
for k = 1:numWindows
segment = signal(segmentStart:segmentStart + winSize - 1);
segmentFFT = fft(segment .* windowVec);
stftMatrix(k, :) = segmentFFT(1:freqVectorLength);
segmentStart = segmentStart + winShift;
end
S = transpose(stftMatrix);
end
function [t,signal] = generate_signal(app)
Td = app.DuratonperkeysecEditField.Value;
Tr = app.RestingDurationsecEditField.Value;
A = app.AmplitudeVEditField.Value;
n = length(app.lst);
samplingRate = app.SamplingRateHzEditField.Value;
length_of_signal = floor((Td + Tr) * n * samplingRate);
signal = zeros(1,length_of_signal);
t = linspace(0,(Td + Tr) * n , length_of_signal);
for i = 1:n
row = floor((app.lst(i)-1) / 3) ;
column = mod(app.lst(i),3);
switch row
case 0
fl = 697;
case 1
fl = 770;
case 2
fl = 852;
case 3
fl = 941;
end
switch column
case 1
fh = 1209;
case 2
```

```
fh = 1336;
case 0
fh = 1447;
end
for j = (Td + Tr) * (i-1) * samplingRate + 1 : ((Td + Tr) * (i-1) + Td) *
samplingRate
signal(int32(j)) = A * (sin(2 * pi * fl* (t(int32(j)) - (i-1) * (Td + Tr))) +
sin(2 * pi * fh* (t(int32(j)) - (i-1) * (Td + Tr))));
end
end
end
function decoded_digits = decode_dtmf_goertzel(app,signal, fs, durationPerKey,
restingDuration, amplitude, windowShift, windowLength, windowType)
% Define DTMF frequency bins
lowFreqBins = [697, 770, 852, 941];
highFreqBins = [1209, 1336, 1477];
keys = ['1', '2', '3'; '4', '5', '6'; '7', '8', '9'; '*', '0', '#'];
% Calculate the number of samples per key and resting duration
samplesPerKey = round(durationPerKey * fs);
samplesResting = round(restingDuration * fs);
% Initialize the output variable
decoded_digits = '';
% Initialize variables for Goertzel algorithm
goertzelBins = [lowFreqBins, highFreqBins];
numBins = length(goertzelBins);
binIndices = round(goertzelBins / fs * windowLength);
% Window function choice
switch windowType
case 'Rectangular'
windowFunc = rectwin(windowLength);
% Add other cases for different window types
case 'Hamming'
windowFunc = hamming(windowLength);
% case 'Hanning'
% windowFunc = hann(windowLength);
% ...
otherwise
windowFunc = rectwin(windowLength); % Default case
end
% Step through the signal in increments of windowShift
startIndex = 1;
while startIndex <= length(signal) - windowLength
% Apply the window to the segment
segment = signal(startIndex:startIndex + windowLength - 1) .* windowFunc;
% Run Goertzel algorithm on the bins of interest
magnitudes = zeros(1, numBins);
for binIndex = 1:numBins
magnitudes(binIndex) = goertzel(app,segment, binIndices(binIndex));
end
% Find the two strongest frequencies in the low and high bins
[~, lowBin] = max(magnitudes(1:length(lowFreqBins)));
[~, highBin] = max(magnitudes(length(lowFreqBins)+1:end));
% Decode the digit
```

```
if magnitudes(lowBin) > amplitude && magnitudes(highBin+length(lowFreqBins)) >
amplitude
row = find(lowFreqBins == goertzelBins(lowBin));
col = find(highFreqBins == goertzelBins(highBin+length(lowFreqBins)));
decoded_digit = keys(row, col);
decoded_digits = [decoded_digits, decoded_digit];
% Skip over the resting period to the next key
startIndex = startIndex + samplesPerKey + samplesResting - windowShift;
else
startIndex = startIndex + windowShift;
end
end
end
function magnitude = goertzel(app,segment, binIndex)
% Initialize Goertzel variables
prev1 = 0;
prev2 = 0;
for i = 1:length(segment)
curr = segment(i) + 2 * cos(2 * pi * binIndex / length(segment)) * prev1 -
prev2;
prev2 = prev1;
prev1 = curr;
end
magnitude = prev1^2 + prev2^2 - prev1 * prev2 * 2 * cos(2 * pi * binIndex /
length(segment));
end


end


% Callbacks that handle component events
methods (Access = private)


% Button pushed function: Button
function ButtonPushed(app, event)
app.lst(end+1) = 1;
app.Label.Text = app.Label.Text + "1 ";
end


% Button pushed function: Button_2
function Button_2Pushed(app, event)
app.lst(end+1) = 2;
app.Label.Text = app.Label.Text + "2 ";
end


% Button pushed function: Button_3
function Button_3Pushed(app, event)
app.lst(end+1) = 3;
app.Label.Text = app.Label.Text + "3 ";
end
```

```matlab
% Button pushed function: Button_4
function Button_4Pushed(app, event)
app.lst(end+1) = 4;
app.Label.Text = app.Label.Text + "4 ";
end


% Button pushed function: Button_5
function Button_5Pushed(app, event)
app.lst(end+1) = 5;
app.Label.Text = app.Label.Text + "5 ";
end


% Button pushed function: Button_6
function Button_6Pushed(app, event)
app.lst(end+1) = 6;
app.Label.Text = app.Label.Text + "6 ";
end


% Button pushed function: Button_7
function Button_7Pushed(app, event)
app.lst(end+1) = 7;
app.Label.Text = app.Label.Text + "7 ";
end


% Button pushed function: Button_8
function Button_8Pushed(app, event)
app.lst(end+1) = 8;
app.Label.Text = app.Label.Text + "8 ";
end


% Button pushed function: Button_9
function Button_9Pushed(app, event)
app.lst(end+1) = 9;
app.Label.Text = app.Label.Text + "9 ";
end


% Button pushed function: Button_10
function Button_10Pushed(app, event)
app.lst(end+1) = 10;
app.Label.Text = app.Label.Text + "* ";
end


% Button pushed function: Button_11
function Button_11Pushed(app, event)
app.lst(end+1) = 11;
app.Label.Text = app.Label.Text + "0 ";
end


% Button pushed function: Button_12
function Button_12Pushed(app, event)
app.lst(end+1) = 12;
app.Label.Text = app.Label.Text + "# ";
end


% Button pushed function: ResetButton
function ResetButtonPushed(app, event)
app.lst = [];
app.Label.Text = "";
end


% Button pushed function: PlotButton
function PlotButtonPushed(app, event)
clear app.UIAxes;
clear app.UIaxes2;
if isempty(app.lst) == false
[t,signal] = generate_signal(app);
plot(app.UIAxes,t,signal);
xlim(app.UIAxes,[min(t),max(t)]);
windowSize = app.WindowLengthofsamplesEditField.Value;
windowShift = app.WindowShiftofsamplesEditField.Value;
windowType = app.WindowTypeDropDown.Value;
samplingRate = app.SamplingRateHzEditField.Value;
[S,T,F]=spec(app,signal,windowSize,windowShift,windowType,samplingRate,max(t))
;
axes(app.UIAxes2);
imagesc(app.UIAxes2,T, F, (20*log10(abs(S))))
axis(app.UIAxes2,'xy')
xlim(app.UIAxes2,[min(T),max(T)]);
ylim(app.UIAxes2,[min(F),max(F)]);
colorbar(app.UIAxes2);
end


end


% Button pushed function: PlayButton
function PlayButtonPushed(app, event)
if isempty(app.lst) == false
samplingRate = app.SamplingRateHzEditField.Value;
[t,signal] = generate_signal(app);
sound(signal,samplingRate);
end
end
```

```matlab
% Button pushed function: SaveButton
function SaveButtonPushed(app, event)
if isempty(app.lst) == false
samplingRate = app.SamplingRateHzEditField.Value;
[t,signal] = generate_signal(app);
signal = signal / max(abs(signal));
audiowrite('dialing.wav', signal, samplingRate);
end
end


% Button pushed function: StartButton
function StartButtonPushed(app, event)
samplingRate = app.SamplingRateHzEditField_2.Value;
app.Recorder = audiorecorder(samplingRate,8,1);
record(app.Recorder);
end


% Button pushed function: StopButton
function StopButtonPushed(app, event)
stop(app.Recorder);
app.loadedAudioFile = [];
end


% Button pushed function: PlotButton_2
function PlotButton_2Pushed(app, event)
clear app.UIAxes2_2;
clear app.UIAxes_2;
if(isempty(app.loadedAudioFile))
recordedData = getaudiodata(app.Recorder);
else
[recordedData, ~] = audioread(app.loadedAudioFile);
recordedData = app.AmplitudeVEditField_2.Value * 2 * recordedData;
end
samplingRate = app.SamplingRateHzEditField_2.Value;
n = length(recordedData)-1;
max_time = n / samplingRate;
t = 0:(1/samplingRate):max_time;
plot(app.UIAxes_2,t,recordedData);
windowSize = app.WindowLengthofsamplesEditField_2.Value;
windowShift = app.WindowShiftofsamplesEditField_2.Value;
windowType = app.WindowTypeDropDown_2.Value;
[S,T,F]=spec(app,recordedData,windowSize,windowShift,windowType,samplingRate,m
ax_time);
axes(app.UIAxes2_2);
imagesc(app.UIAxes2_2,T, F, (20*log10(abs(S))))
axis(app.UIAxes2_2,'xy')
xlim(app.UIAxes2_2,[min(T),max(T)]);
ylim(app.UIAxes2_2,[min(F),max(F)]);
colorbar(app.UIAxes2_2);
```

```
end


% Button pushed function: DecodeButton
function DecodeButtonPushed(app, event)
fl_lst = [697 770 852 941];
fh_lst = [1209 1336 1477];
app.Label_2.Text = " ";
switch app.DecodingAlgorithmDropDown.Value
case "Spectogram Based"
if(isempty(app.loadedAudioFile))
recordedData = getaudiodata(app.Recorder);
else
[recordedData, ~] = audioread(app.loadedAudioFile);
recordedData = app.AmplitudeVEditField_2.Value * 2 * recordedData;
end
samplingRate = app.SamplingRateHzEditField_2.Value;
n = length(recordedData)-1;
max_time = n / samplingRate;
windowSize = app.WindowLengthofsamplesEditField_2.Value;
windowShift = app.WindowShiftofsamplesEditField_2.Value;
windowType = app.WindowTypeDropDown_2.Value;
[S,~,F]=spec(app,recordedData,windowSize,windowShift,windowType,samplingRate,m
ax_time);
S = abs(S);
sum_ = sum(S,1);
[~,col] = find(sum_< 10);
col = unique(col);
if col(1) ~= 1
zerostartlst = [col(1)-1];
zeroendlst = [1];
else
zerostartlst = [];
zeroendlst = [];
end
for i = 2:length(col)
if col(i) - col(i-1) > 1
zerostartlst(end+1) = col(i) - 1;
zeroendlst(end+1) = col(i-1) + 1;
end
end
n = length(zeroendlst);
F_lower_1000 = F(F<1000);
F_higher_1000 = F(F>1000);
for k = 1:n
S_ = S(:, zeroendlst(k):zerostartlst(k));
S_with_F_lower_1000 = S_(F<1000,:);
S_with_F_higher_1000 = S_(F>1000,:);
[~,i] = max(S_with_F_lower_1000,[],1);
fl = mean(F_lower_1000(i));
[~,i] = max(S_with_F_higher_1000,[],1);
fh = mean(F_higher_1000(i));
[~,i] = min(abs(fl_lst - fl),[],2);
```

```
[~,j] = min(abs(fh_lst - fh),[],2);
number = (i-1)*3 + j;
if number < 10
app.Label_2.Text = strcat(app.Label_2.Text, num2str(number));
elseif number == 10
app.Label_2.Text = app.Label_2.Text + "*";
elseif number == 11
app.Label_2.Text = app.Label_2.Text + "0";
elseif number == 12
app.Label_2.Text = app.Label_2.Text + "#";
end
end
case "Goertzel Algorithm"
if(isempty(app.loadedAudioFile))
recordedData = getaudiodata(app.Recorder);
else
[recordedData, ~] = audioread(app.loadedAudioFile);
recordedData = app.AmplitudeVEditField_2.Value * 2 * recordedData;
end
samplingRate = app.SamplingRateHzEditField_2.Value;
windowSize = app.WindowLengthofsamplesEditField_2.Value;
windowShift = app.WindowShiftofsamplesEditField_2.Value;
windowType = app.WindowTypeDropDown_2.Value;
durationPerKey = app.DuratonperkeysecEditField_2.Value;
restingDuration = app.RestingDurationsecEditField_2.Value;
amplitude = app.AmplitudeVEditField_2.Value;
decoded_digits = decode_dtmf_goertzel(app,recordedData, samplingRate,
durationPerKey, restingDuration, amplitude, windowShift, windowSize,
windowType);
app.Label_2.Text = decoded_digits;
end
end


% Button pushed function: LoadButton
function LoadButtonPushed(app, event)
[filename, pathname] = uigetfile({'*.wav';'*.mp3';}, 'Select an Audio File');
if ischar(filename) % An audio file was selected
app.loadedAudioFile = fullfile(pathname, filename);
assignin("base","signal",app.loadedAudioFile);


end
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 = [100 100 927 606];
app.UIFigure.Name = 'MATLAB App';


% Create TabGroup
app.TabGroup = uitabgroup(app.UIFigure);
app.TabGroup.Position = [1 1 928 606];


% Create TransmitterTab
app.TransmitterTab = uitab(app.TabGroup);
app.TransmitterTab.Title = 'Transmitter';


% Create UIAxes
app.UIAxes = uiaxes(app.TransmitterTab);
title(app.UIAxes, 'Time Domain')
xlabel(app.UIAxes, 'Time (sec)')
ylabel(app.UIAxes, 'Amplitude (V)')
zlabel(app.UIAxes, 'Z')
app.UIAxes.Position = [520 345 300 185];


% Create UIAxes2
app.UIAxes2 = uiaxes(app.TransmitterTab);
title(app.UIAxes2, 'Spectogram')
xlabel(app.UIAxes2, 'Time (sec)')
ylabel(app.UIAxes2, 'Frequency (Hz)')
zlabel(app.UIAxes2, 'Z')
app.UIAxes2.Position = [520 84 300 185];


% Create Button
app.Button = uibutton(app.TransmitterTab, 'push');
app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed, true);
app.Button.Position = [35 508 31 22];
app.Button.Text = '1';


% Create Button_2
app.Button_2 = uibutton(app.TransmitterTab, 'push');
app.Button_2.ButtonPushedFcn = createCallbackFcn(app, @Button_2Pushed, true);
app.Button_2.Position = [75 508 31 22];
app.Button_2.Text = '2';


% Create Button_3
app.Button_3 = uibutton(app.TransmitterTab, 'push');
app.Button_3.ButtonPushedFcn = createCallbackFcn(app, @Button_3Pushed, true);
app.Button_3.Position = [120 508 31 22];
app.Button_3.Text = '3';
```

```
% Create Button_4
app.Button_4 = uibutton(app.TransmitterTab, 'push');
app.Button_4.ButtonPushedFcn = createCallbackFcn(app, @Button_4Pushed, true);
app.Button_4.Position = [35 471 31 22];
app.Button_4.Text = '4';


% Create Button_5
app.Button_5 = uibutton(app.TransmitterTab, 'push');
app.Button_5.ButtonPushedFcn = createCallbackFcn(app, @Button_5Pushed, true);
app.Button_5.Position = [75 471 31 22];
app.Button_5.Text = '5';


% Create Button_6
app.Button_6 = uibutton(app.TransmitterTab, 'push');
app.Button_6.ButtonPushedFcn = createCallbackFcn(app, @Button_6Pushed, true);
app.Button_6.Position = [119 471 31 22];
app.Button_6.Text = '6';


% Create Button_7
app.Button_7 = uibutton(app.TransmitterTab, 'push');
app.Button_7.ButtonPushedFcn = createCallbackFcn(app, @Button_7Pushed, true);
app.Button_7.Position = [35 438 31 22];
app.Button_7.Text = '7';


% Create Button_8
app.Button_8 = uibutton(app.TransmitterTab, 'push');
app.Button_8.ButtonPushedFcn = createCallbackFcn(app, @Button_8Pushed, true);
app.Button_8.Position = [79 438 31 22];
app.Button_8.Text = '8';


% Create Button_9
app.Button_9 = uibutton(app.TransmitterTab, 'push');
app.Button_9.ButtonPushedFcn = createCallbackFcn(app, @Button_9Pushed, true);
app.Button_9.Position = [120 438 31 22];
app.Button_9.Text = '9';


% Create Button_10
app.Button_10 = uibutton(app.TransmitterTab, 'push');
app.Button_10.ButtonPushedFcn = createCallbackFcn(app, @Button_10Pushed,
true);
app.Button_10.Position = [35 403 31 22];
app.Button_10.Text = '*';


% Create Button_11
app.Button_11 = uibutton(app.TransmitterTab, 'push');
app.Button_11.ButtonPushedFcn = createCallbackFcn(app, @Button_11Pushed,
true);
```

```
app.Button_11.Position = [75 403 31 22];
app.Button_11.Text = '0';


% Create Button_12
app.Button_12 = uibutton(app.TransmitterTab, 'push');
app.Button_12.ButtonPushedFcn = createCallbackFcn(app, @Button_12Pushed,
true);
app.Button_12.Position = [120 403 31 22];
app.Button_12.Text = '#';


% Create PressedButtonsLabel
app.PressedButtonsLabel = uilabel(app.TransmitterTab);
app.PressedButtonsLabel.Position = [45 345 100 22];
app.PressedButtonsLabel.Text = 'Pressed Buttons: ';


% Create Label
app.Label = uilabel(app.TransmitterTab);
app.Label.Position = [142 346 272 22];
app.Label.Text = '';


% Create ResetButton
app.ResetButton = uibutton(app.TransmitterTab, 'push');
app.ResetButton.ButtonPushedFcn = createCallbackFcn(app, @ResetButtonPushed,
true);
app.ResetButton.Position = [179 508 58 22];
app.ResetButton.Text = 'Reset';


% Create SaveButton
app.SaveButton = uibutton(app.TransmitterTab, 'push');
app.SaveButton.ButtonPushedFcn = createCallbackFcn(app, @SaveButtonPushed,
true);
app.SaveButton.Position = [179 471 58 22];
app.SaveButton.Text = 'Save';


% Create PlayButton
app.PlayButton = uibutton(app.TransmitterTab, 'push');
app.PlayButton.ButtonPushedFcn = createCallbackFcn(app, @PlayButtonPushed,
true);
app.PlayButton.Position = [179 438 58 22];
app.PlayButton.Text = 'Play';


% Create RestingDurationsecEditFieldLabel
app.RestingDurationsecEditFieldLabel = uilabel(app.TransmitterTab);
app.RestingDurationsecEditFieldLabel.HorizontalAlignment = 'right';
app.RestingDurationsecEditFieldLabel.Position = [39 285 125 22];
app.RestingDurationsecEditFieldLabel.Text = 'Resting Duration (sec)';
```

```matlab
% Create RestingDurationsecEditField
app.RestingDurationsecEditField = uieditfield(app.TransmitterTab, 'numeric');
app.RestingDurationsecEditField.Position = [178 285 56 22];
app.RestingDurationsecEditField.Value = 0.1;


% Create PlotButton
app.PlotButton = uibutton(app.TransmitterTab, 'push');
app.PlotButton.ButtonPushedFcn = createCallbackFcn(app, @PlotButtonPushed,
true);
app.PlotButton.Position = [179 403 58 22];
app.PlotButton.Text = 'Plot';


% Create WindowShiftofsamplesEditFieldLabel
app.WindowShiftofsamplesEditFieldLabel = uilabel(app.TransmitterTab);
app.WindowShiftofsamplesEditFieldLabel.HorizontalAlignment = 'right';
app.WindowShiftofsamplesEditFieldLabel.Position = [39 84 158 22];
app.WindowShiftofsamplesEditFieldLabel.Text = 'Window Shift (# of samples)';


% Create WindowShiftofsamplesEditField
app.WindowShiftofsamplesEditField = uieditfield(app.TransmitterTab,
'numeric');
app.WindowShiftofsamplesEditField.Position = [213 84 100 22];
app.WindowShiftofsamplesEditField.Value = 128;


% Create WindowTypeDropDownLabel
app.WindowTypeDropDownLabel = uilabel(app.TransmitterTab);
app.WindowTypeDropDownLabel.HorizontalAlignment = 'right';
app.WindowTypeDropDownLabel.Position = [39 47 77 22];
app.WindowTypeDropDownLabel.Text = 'Window Type';


% Create WindowTypeDropDown
app.WindowTypeDropDown = uidropdown(app.TransmitterTab);
app.WindowTypeDropDown.Items = {'Rectangular', 'Hamming'};
app.WindowTypeDropDown.Position = [131 47 100 22];
app.WindowTypeDropDown.Value = 'Rectangular';


% Create DuratonperkeysecEditField_3Label
app.DuratonperkeysecEditField_3Label = uilabel(app.TransmitterTab);
app.DuratonperkeysecEditField_3Label.HorizontalAlignment = 'right';
app.DuratonperkeysecEditField_3Label.Position = [39 315 122 22];
app.DuratonperkeysecEditField_3Label.Text = 'Duraton-per-key (sec)';


% Create DuratonperkeysecEditField
app.DuratonperkeysecEditField = uieditfield(app.TransmitterTab, 'numeric');
app.DuratonperkeysecEditField.Position = [173 315 100 22];
app.DuratonperkeysecEditField.Value = 0.1;
```

```
% Create WindowLengthofsamplesEditField_3Label
app.WindowLengthofsamplesEditField_3Label = uilabel(app.TransmitterTab);
app.WindowLengthofsamplesEditField_3Label.HorizontalAlignment = 'right';
app.WindowLengthofsamplesEditField_3Label.Position = [39 117 168 22];
app.WindowLengthofsamplesEditField_3Label.Text = 'Window Length (# of
samples)';


% Create WindowLengthofsamplesEditField
app.WindowLengthofsamplesEditField = uieditfield(app.TransmitterTab,
'numeric');
app.WindowLengthofsamplesEditField.Position = [224 117 100 22];
app.WindowLengthofsamplesEditField.Value = 256;


% Create SamplingRateHzEditField_3Label
app.SamplingRateHzEditField_3Label = uilabel(app.TransmitterTab);
app.SamplingRateHzEditField_3Label.HorizontalAlignment = 'right';
app.SamplingRateHzEditField_3Label.Position = [39 150 110 22];
app.SamplingRateHzEditField_3Label.Text = 'Sampling Rate (Hz)';


% Create SamplingRateHzEditField
app.SamplingRateHzEditField = uieditfield(app.TransmitterTab, 'numeric');
app.SamplingRateHzEditField.Position = [164 150 100 22];
app.SamplingRateHzEditField.Value = 8000;


% Create AmplitudeVEditField_3Label
app.AmplitudeVEditField_3Label = uilabel(app.TransmitterTab);
app.AmplitudeVEditField_3Label.HorizontalAlignment = 'right';
app.AmplitudeVEditField_3Label.Position = [39 244 78 22];
app.AmplitudeVEditField_3Label.Text = 'Amplitude (V)';


% Create AmplitudeVEditField
app.AmplitudeVEditField = uieditfield(app.TransmitterTab, 'numeric');
app.AmplitudeVEditField.Position = [132 244 100 22];
app.AmplitudeVEditField.Value = 1;


% Create RecieverTab
app.RecieverTab = uitab(app.TabGroup);
app.RecieverTab.Title = 'Reciever';


% Create UIAxes_2
app.UIAxes_2 = uiaxes(app.RecieverTab);
title(app.UIAxes_2, 'Time Domain')
xlabel(app.UIAxes_2, 'Time (sec)')
ylabel(app.UIAxes_2, 'Amplitude (V)')
zlabel(app.UIAxes_2, 'Z')
app.UIAxes_2.Position = [535 348 300 185];
```

```
% Create UIAxes2_2
app.UIAxes2_2 = uiaxes(app.RecieverTab);
title(app.UIAxes2_2, 'Spectogram')
xlabel(app.UIAxes2_2, 'Time (sec)')
ylabel(app.UIAxes2_2, 'Frequency (Hz)')
zlabel(app.UIAxes2_2, 'Z')
app.UIAxes2_2.Position = [536 96 300 185];


% Create StartButton
app.StartButton = uibutton(app.RecieverTab, 'push');
app.StartButton.ButtonPushedFcn = createCallbackFcn(app, @StartButtonPushed,
true);
app.StartButton.Position = [25 508 56 22];
app.StartButton.Text = 'Start';


% Create StopButton
app.StopButton = uibutton(app.RecieverTab, 'push');
app.StopButton.ButtonPushedFcn = createCallbackFcn(app, @StopButtonPushed,
true);
app.StopButton.Position = [26 471 56 22];
app.StopButton.Text = 'Stop';


% Create PressedButtonsLabel_2
app.PressedButtonsLabel_2 = uilabel(app.RecieverTab);
app.PressedButtonsLabel_2.Position = [33 426 100 22];
app.PressedButtonsLabel_2.Text = 'Pressed Buttons: ';


% Create RestingDurationsecEditField_2Label
app.RestingDurationsecEditField_2Label = uilabel(app.RecieverTab);
app.RestingDurationsecEditField_2Label.HorizontalAlignment = 'right';
app.RestingDurationsecEditField_2Label.Position = [25 345 125 22];
app.RestingDurationsecEditField_2Label.Text = 'Resting Duration (sec)';


% Create RestingDurationsecEditField_2
app.RestingDurationsecEditField_2 = uieditfield(app.RecieverTab, 'numeric');
app.RestingDurationsecEditField_2.Position = [164 345 56 22];
app.RestingDurationsecEditField_2.Value = 0.1;


% Create AmplitudeVEditField_2Label
app.AmplitudeVEditField_2Label = uilabel(app.RecieverTab);
app.AmplitudeVEditField_2Label.HorizontalAlignment = 'right';
app.AmplitudeVEditField_2Label.Position = [25 306 78 22];
app.AmplitudeVEditField_2Label.Text = 'Amplitude (V)';


% Create AmplitudeVEditField_2
app.AmplitudeVEditField_2 = uieditfield(app.RecieverTab, 'numeric');
app.AmplitudeVEditField_2.Position = [121 306 100 22];
app.AmplitudeVEditField_2.Value = 1;


% Create SamplingRateHzEditField_2Label
app.SamplingRateHzEditField_2Label = uilabel(app.RecieverTab);
app.SamplingRateHzEditField_2Label.HorizontalAlignment = 'right';
app.SamplingRateHzEditField_2Label.Position = [25 259 110 22];
app.SamplingRateHzEditField_2Label.Text = 'Sampling Rate (Hz)';


% Create SamplingRateHzEditField_2
app.SamplingRateHzEditField_2 = uieditfield(app.RecieverTab, 'numeric');
app.SamplingRateHzEditField_2.Position = [152 259 100 22];
app.SamplingRateHzEditField_2.Value = 8000;


% Create WindowShiftofsamplesEditField_2Label
app.WindowShiftofsamplesEditField_2Label = uilabel(app.RecieverTab);
app.WindowShiftofsamplesEditField_2Label.HorizontalAlignment = 'right';
app.WindowShiftofsamplesEditField_2Label.Position = [25 211 158 22];
app.WindowShiftofsamplesEditField_2Label.Text = 'Window Shift (# of samples)';


% Create WindowShiftofsamplesEditField_2
app.WindowShiftofsamplesEditField_2 = uieditfield(app.RecieverTab, 'numeric');
app.WindowShiftofsamplesEditField_2.Position = [202 211 100 22];
app.WindowShiftofsamplesEditField_2.Value = 128;


% Create WindowTypeDropDown_2Label
app.WindowTypeDropDown_2Label = uilabel(app.RecieverTab);
app.WindowTypeDropDown_2Label.HorizontalAlignment = 'right';
app.WindowTypeDropDown_2Label.Position = [25 134 77 22];
app.WindowTypeDropDown_2Label.Text = 'Window Type';


% Create WindowTypeDropDown_2
app.WindowTypeDropDown_2 = uidropdown(app.RecieverTab);
            app.WindowTypeDropDown_2.Items = {'Rectangular', 'Hamming'};
            app.WindowTypeDropDown_2.Position = [118 134 100 22];
            app.WindowTypeDropDown_2.Value = 'Rectangular';


            % Create Label_2
            app.Label_2 = uilabel(app.RecieverTab);
            app.Label_2.Position = [139 426 272 22];
            app.Label_2.Text = '';


            % Create DecodingAlgorithmDropDownLabel
            app.DecodingAlgorithmDropDownLabel = uilabel(app.RecieverTab);
            app.DecodingAlgorithmDropDownLabel.HorizontalAlignment = 'right';
            app.DecodingAlgorithmDropDownLabel.Position = [25 96 110 22];
            app.DecodingAlgorithmDropDownLabel.Text = 'Decoding Algorithm';
            % Create DecodingAlgorithmDropDown
            app.DecodingAlgorithmDropDown = uidropdown(app.RecieverTab);
            app.DecodingAlgorithmDropDown.Items = {'Spectogram Based',
'Goertzel Algorithm'};
            app.DecodingAlgorithmDropDown.Position = [150 96 144 22];
            app.DecodingAlgorithmDropDown.Value = 'Spectogram Based';

            % Create PlotButton_2
            app.PlotButton_2 = uibutton(app.RecieverTab, 'push');
            app.PlotButton_2.ButtonPushedFcn = createCallbackFcn(app,
@PlotButton_2Pushed, true);
            app.PlotButton_2.Position = [105 508 59 22];
            app.PlotButton_2.Text = 'Plot';

            % Create DecodeButton
            app.DecodeButton = uibutton(app.RecieverTab, 'push');
            app.DecodeButton.ButtonPushedFcn = createCallbackFcn(app,
@DecodeButtonPushed, true);
            app.DecodeButton.Position = [107 471 59 22];
            app.DecodeButton.Text = 'Decode';

            % Create DuratonperkeysecEditField_2Label
            app.DuratonperkeysecEditField_2Label = uilabel(app.RecieverTab);
            app.DuratonperkeysecEditField_2Label.HorizontalAlignment =
'right';
            app.DuratonperkeysecEditField_2Label.Position = [25 382 122 22];
            app.DuratonperkeysecEditField_2Label.Text = 'Duraton-per-key
(sec)';

            % Create DuratonperkeysecEditField_2
            app.DuratonperkeysecEditField_2 = uieditfield(app.RecieverTab,
'numeric');
            app.DuratonperkeysecEditField_2.Position = [162 382 100 22];
            app.DuratonperkeysecEditField_2.Value = 0.1;
% Create WindowLengthofsamplesEditField_2Label

        app.WindowLengthofsamplesEditField_2Label = uilabel(app.RecieverTab);

        app.WindowLengthofsamplesEditField_2Label.HorizontalAlignment = 'right';

        app.WindowLengthofsamplesEditField_2Label.Position = [25 177 168 22];

        app.WindowLengthofsamplesEditField_2Label.Text = 'Window Length (# of samples)';


        % Create WindowLengthofsamplesEditField_2

        app.WindowLengthofsamplesEditField_2 = uieditfield(app.RecieverTab, 'numeric');
```

```
            app.WindowLengthofsamplesEditField_2.Position = [213 177 100 22];
            app.WindowLengthofsamplesEditField_2.Value = 256;

            % Create LoadButton
            app.LoadButton = uibutton(app.RecieverTab, 'push');
            app.LoadButton.ButtonPushedFcn = createCallbackFcn(app, @LoadButtonPushed, true);
            app.LoadButton.Position = [197 508 56 22];
            app.LoadButton.Text = 'Load';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)
```