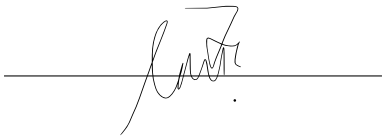
ECE280 - Lab 4: Touch-Tone Phone System

Sylvester Arizie (sna31)

October 30, 2024

I have adhered to the Duke Community Standard in completing this assignment.



Contents

1 Objectives

In this project we set out to: Explore the standard DTMF phone dialing system Create a software-based DTMF tone generator (encoder) Build a real-time software-based DTMF receiver (decoder)

We aim to accomplish these goals by developing two key subsystems: a DTMF encoder and a DTMF decoder. For the encoder, we will write MATLAB code to generate tones for any phone number. For the decoder, we will design a system using a Simulink block diagram to translate the tones back into numbers, which will be displayed on a scope.

2 Background

The design of a DTMF system consists of two main components: the encoder and the decoder.

Encoder Design In the encoder, pressing any key on the telephone keypad generates two specific sinusoidal tones, one from a row frequency and one from a column frequency. For example, pressing the '6' key produces a signal that combines 770 Hz (row) and 1477 Hz (column). This dual-tone signal effectively transmits the corresponding number. The frequencies, as shown in the table given in the pre-lab, were carefully selected to ensure that no two frequencies are harmonically related, meaning none are integer multiples of each other or equal to the sum or difference of others. This design choice simplifies the decoding process and enhances reliability against potential line distortions or noise.

Decoder Design The decoder's role is to identify the DTMF tones received and translate them back into their corresponding numbers. It analyzes the incoming signal, detects the two active frequencies, and maps them back to the appropriate keypad entry based on their positions in the frequency table. The dual-tone approach also prevents accidental detection of voice sounds as button presses, ensuring accuracy in the identification process.

Overall, the DTMF system's robust design minimizes the effects of non-linear distortions and enhances the clarity of signal transmission, facilitating effective communication.

3 Results and Discussion

1. Include your dtmfdial.m file.

```
fs =8000;
function PhoneNum = dtmfdial(KeyNames, fs)
% DTMFDIAL Create a signal vector of tones which will dial a
% DTMF telephone system.
%
% usage: PhoneNum = dtmfdial(KeyNames, fs)
% KeyNames = a vector of characters containing valid key names
% fs = sampling frequency
% PhoneNum = signal vector that is the concatenation of DTMF tones
fs =8000;
% Defining the DTMF key matrix and frequency matrices
dtmf.Keys = ['1', '2', '3';
             '4', '5', '6';
             '7', '8', '9';
             '*', '0', '#'];

dtmf.colTones = ones(4,1) * [1209, 1336, 1477];
dtmf.rowTones = [697; 770; 852; 941] * ones(1,3);


% Duration of each tone and pause
toneDuration = 0.5;
pauseDuration = 0.2;
```

```
% time vectors for tone and pause
tNote = 0:(1/fs):(toneDuration-1/fs);
tPause = 0:(1/fs):(pauseDuration-1/fs);

% Create pause signal
pauseSignal = zeros(size(tPause));

% Initialize output vector
PhoneNum = [];

% Process each key in the input
for key = KeyNames
    % Find the position of the key in the matrix
    [R, C] = find(dtmf.Keys == key);

    if ~isempty(R) && ~isempty(C)
        % Get the corresponding frequencies
        f1 = dtmf.rowTones(R, C);
        f2 = dtmf.colTones(R, C);

        % Generate the dual tone signal
        toneSignal = sin(2*pi*f1*tNote) + sin(2*pi*f2*tNote);

        % Add tone and pause to the output
        PhoneNum = [PhoneNum, toneSignal, pauseSignal];
    end
end
end

function plot_dtmf_tones_1_5_9()
    % Setting up parameters
    fs = 8000;  % Sampling frequency
    keys = ['1', '5', '9'];  % Keys to analyze
    freqs = [697, 770, 852, 941, 1209, 1336, 1477];  % DTMF frequencies

    % Create figure for all plots
    figure('Position', [100, 100, 1200, 800]);

    % Loop through each key and plot its DTMF tone
    for i = 1:length(keys)
        key = keys(i);
        signal = dtmfdial(key, fs);  % Generate DTMF signal
        t = (0:0.5*fs-1)/fs;

        % Plotting the time-domain signal
        subplot(3, 1, i);
        plot(t, signal(1:length(t)), 'DisplayName', ['Key ' key]);
        title(['DTMF Signal in Time Domain for Key ' key]);
        xlabel('Time (s)');
        ylabel('Amplitude');
        legend('show');
        grid on;
    end
```

```
end

%creating plots for keys 1, 5, and 9
%plot_dtmf_tones_1_5_9();

%analysis with correlations
function analyze_dtmf_tones()
    % Setup parameters
    fs = 8000;  % Sampling frequency
    keys = ['1', '5', '9'];
    freqs = [697, 770, 852, 941, 1209, 1336, 1477];

    for i = 1:length(keys)
        % Create a new figure for each key
        figure('Position', [100, 100, 800, 600]);

        % Generating the DTMF signal for the current key
        signal = dtmfdial(keys(i), fs);
        t = (0:0.5*fs-1)/fs;

        % Normalizing the signal to the range -1 to 1
        signal = signal / max(abs(signal));  %

        % Plotting original signal against each frequency in subplots
        for j = 1:length(freqs)
            % Creating reference signal for the current frequency
            ref = sin(2*pi*freqs(j)*t);
            ref = ref / max(abs(ref));  % Normalize reference signal

            % Creating a subplot for the current frequency
            subplot(length(freqs), 1, j);  % Change to vertical layout
            plot(t, signal(1:length(t)), 'k', 'LineWidth', 0.1,
            'DisplayName', ['Key ' keys(i)]);  % Black for original
            hold on;
            plot(t, ref, 'Color', [0.778 0.547 0.502],
            'LineWidth', 0.01, 'DisplayName', sprintf('%d Hz', freqs(j)));
            % Light blue for reference
            hold off;
            title(sprintf('Key %s - Original vs %d Hz', keys(i), freqs(j)));
            xlabel('Time (s)');
            ylabel('Amplitude');
            ylim([-1 1]);  % Set y-axis limits
            grid on;
            legend('show');
        end
    end
end

%plotting the analysis
analyze_dtmf_tones();

%using correlation to find unknown_key
function analyze_unknown_key()
    % Load the unknown key
```

```
    load('UnknownKey.mat');

    % Normalize the UnknownKey to be between -1 and 1
    UnknownKey = UnknownKey / max(abs(UnknownKey));

    fs = 8000;
    freqs = [697, 770, 852, 941, 1209, 1336, 1477];

    figure('Position', [100, 100, 800, 600]);

    % Store maximum correlation for each frequency
    max_corrs = zeros(1, length(freqs));

    t_ref = 0:1/fs:0.01; % Time reference for sinusoids
    for i = 1:length(freqs)
        ref = sin(2*pi*freqs(i)*t_ref);
        corr_result = xcorr(UnknownKey, ref);
        corr_result = corr_result / max(abs(corr_result));
        % Normalize correlation result
        max_corrs(i) = max(abs(corr_result));

        lag = (-length(corr_result)/2:length(corr_result)/2-1)/fs;

        % Only include non-negative lags
        positive_lags = lag(lag >= 0);
        positive_corrs = corr_result(lag >= 0);

        % Create a subplot for each frequency
        subplot(length(freqs), 1, i);
        plot(positive_lags * 1000, positive_corrs,
        'DisplayName', sprintf('%d Hz', freqs(i)), 'LineWidth', 0.01);
        hold on;

        % Plot the normalized unknown key for reference (time axis adjusted)
        t_unknown = (0:length(UnknownKey)-1) / fs;
        plot(t_unknown * 1000, UnknownKey, 'k', 'LineWidth', 1,
        'DisplayName', 'Unknown Key');
        hold off;

        title(sprintf('Correlation with %d Hz', freqs(i)));
        xlabel('Lag (ms)');
        ylabel('Normalized Correlation');
        grid on;
        legend('show', 'Location', 'best');
    end
end

%analyze_unknown_key();
```

2. Include a printout of your completed Simple_Corr.mdl.

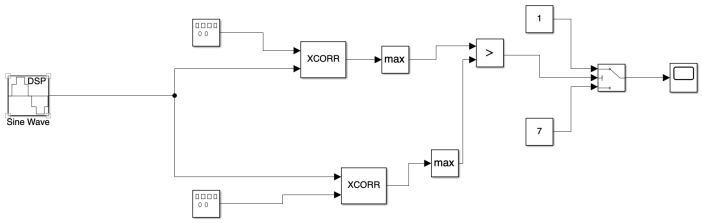
   Explain how this model works.

   The Simple_Corr model is designed to identify input tones of either 697 Hz or 1477 Hz using a basic correlator. It starts by generating a sine wave signal at the desired frequency through a DSP Sine Wave

block.This sine wave represents the tone that needs to be analyzed. The output from this block is then fed into a Relational Operator, which compares the incoming signal against reference frequencies. If the incoming tone is lower than 697 Hz, the Relational Operator outputs '1'; if it is higher, indicating 1477 Hz, it outputs '0'. The output from the Relational Operator is directed to a Switch block. This block decides which constant value to pass to the Scope based on the comparison. If the Relational Operator outputs '1', the Switch transmits a constant value of '1' to indicate the presence of the 697 Hz tone. Conversely, if the output is '0', the Switch sends a constant value of '7' to indicate the 1477 Hz tone. The final output is displayed on a Scope, allowing for real-time visualization of the detected tone. Overall, the Simple_Corr model effectively illustrates fundamental concepts in signal processing, demonstrating how to implement a basic tone detection system that can be further expanded for more complex tasks.

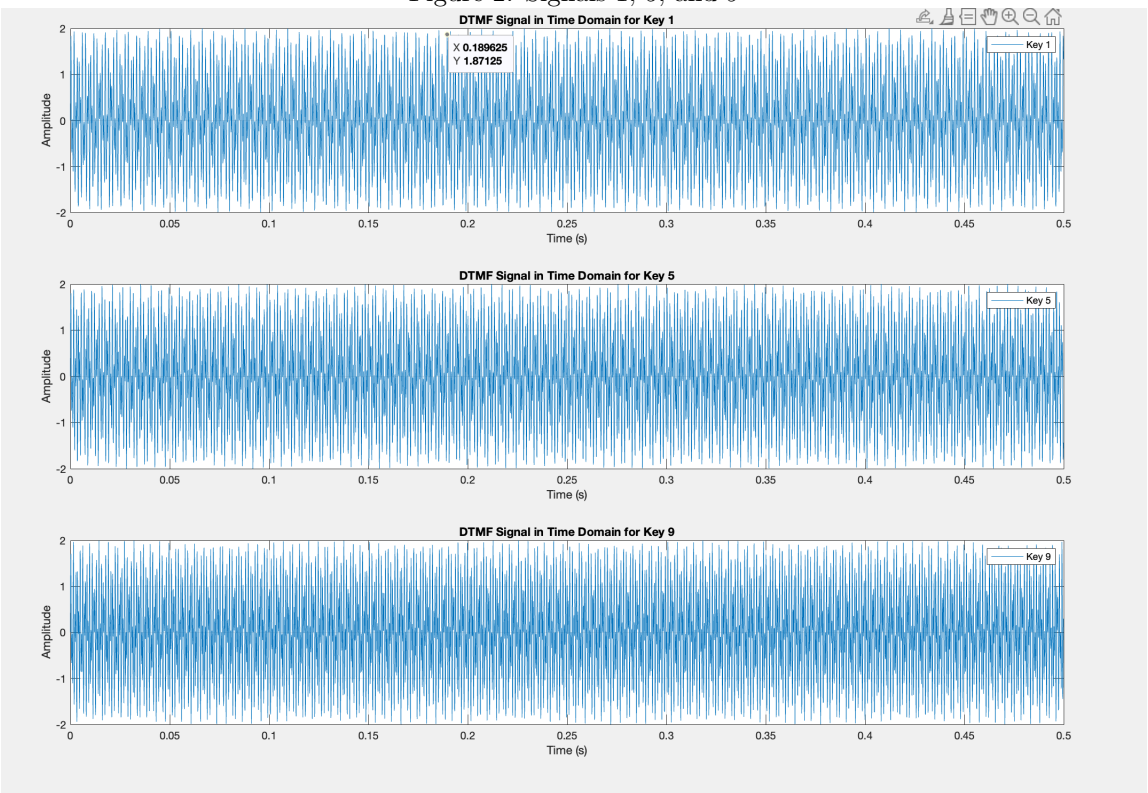
Figure 1: Printout of

SimpleCorr.mdl



3. Include plots of the signals corresponding to the keys 1, 5, and 9.

Figure 2: Signals 1, 5, and 9



4. Include plots of the correlation of the keys 1, 5, and 9 to the seven individual DTMF tones.

Figure 3: Plot of correlation of Key 1 with all 7 frequencies

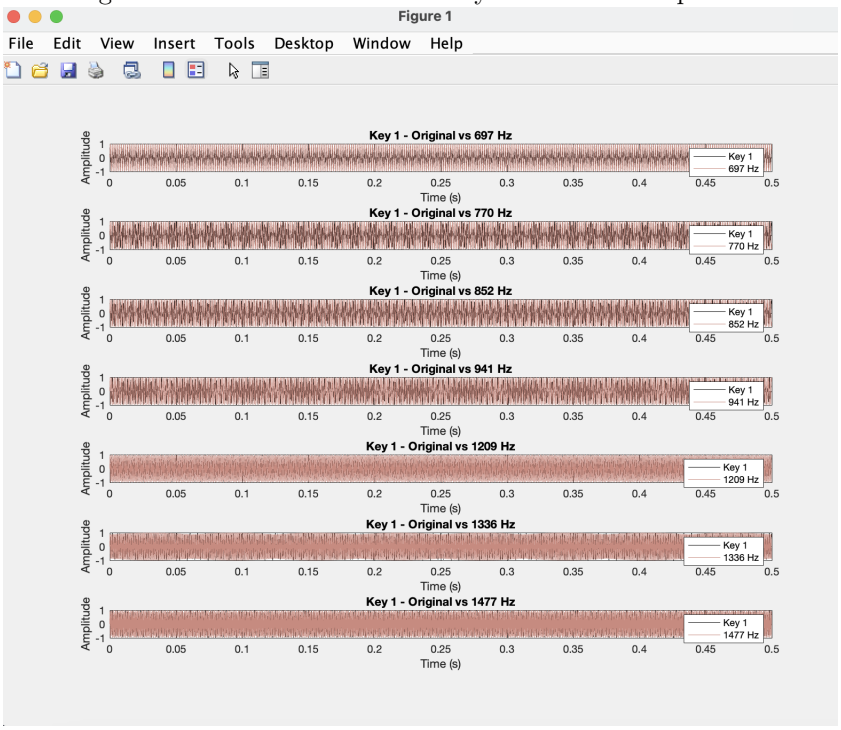


Figure 4: Plot of correlation of Key 5 with all 7 frequencies

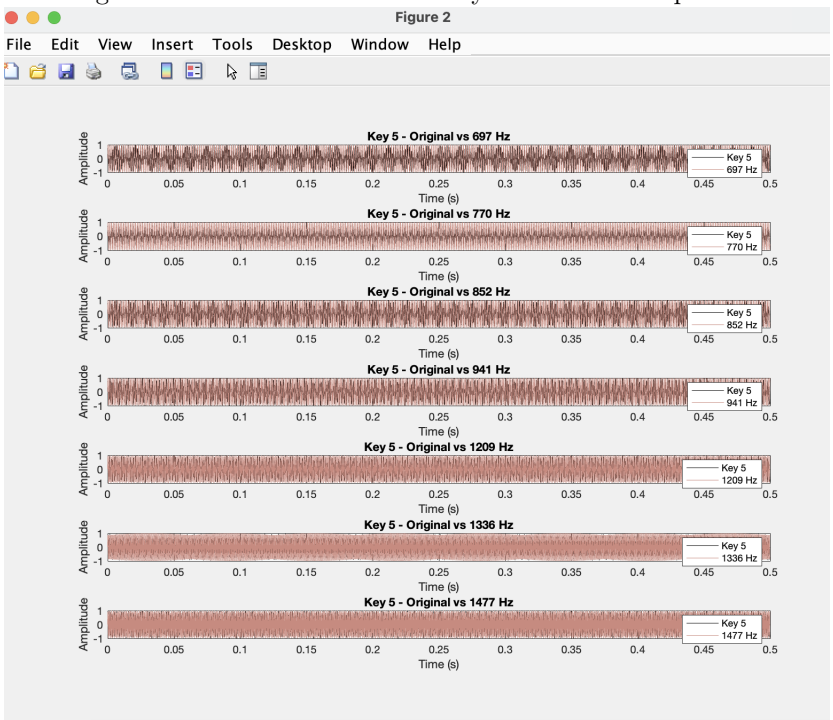
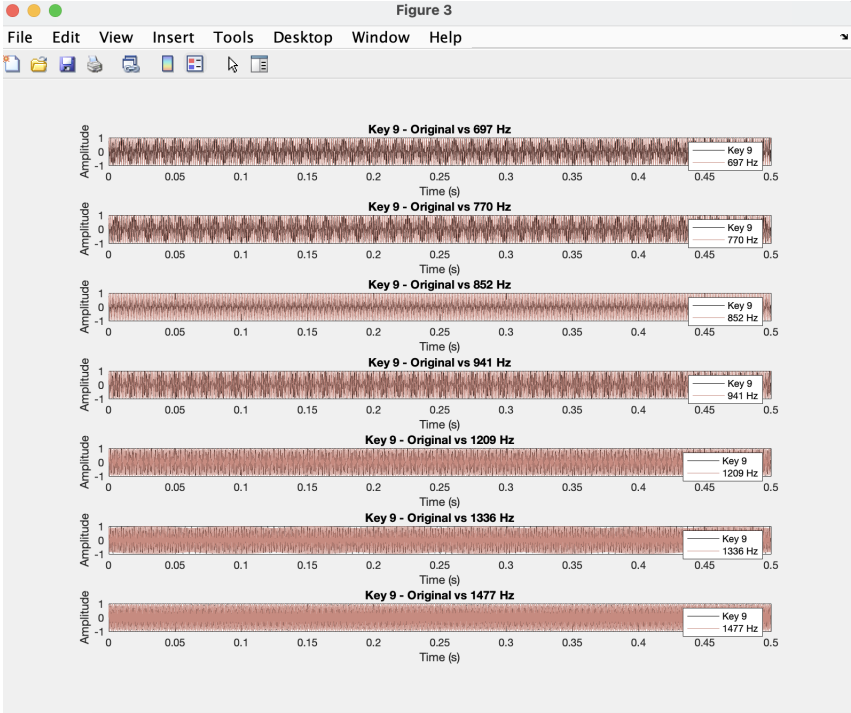


Figure 5: Plot of correlation of Key 9 with all 7 frequencies



5. Discuss what you observed about the correlation plots you generated for the three keys (1, 5, 9).

   From the correlation plots of the signals, there is a heightened overlap between the signals of the key and the frequencies from which the keys were generated whereas huge distortions are seen when the signal is correlated with other frequencies. For instance, for key 9, we see a greater overlap in the correlation when key 9 is correlated with '852Hz' and '1477Hz'. .

6. How did you go about determining the value of the UnknownKey? The Unknown Key was identified by overlapping the unknown signal with all 7 frequencies, and from the correlation results, I identified the frequencies with the maximum overlap to backtrack them to the detected key using the pre-lab table as a guide. There were three frequencies 697, 770 and 1477 HZ , and this corresponded to the keys 3 and 6. However , the greater overlap corresponded to key 3 and this noise could have been introduced by the closeness of 697Hz and 770Hz. Additionally, instead of hard coding, we could have used the DTMF decoder to identify what key was detected or formed in our signal as a check.

Figure 6: Correlation of Unknown Signal with all seven frequencies

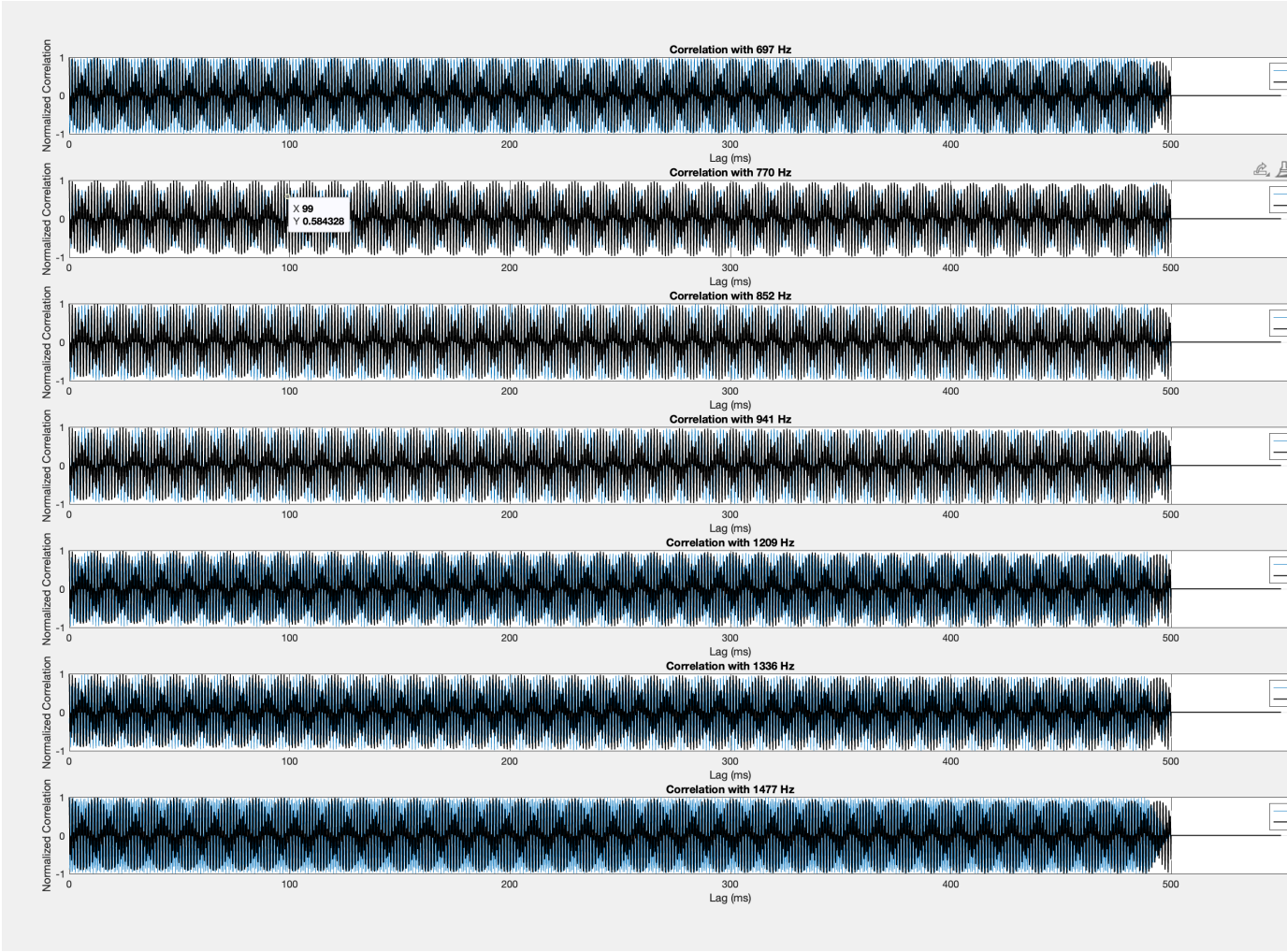
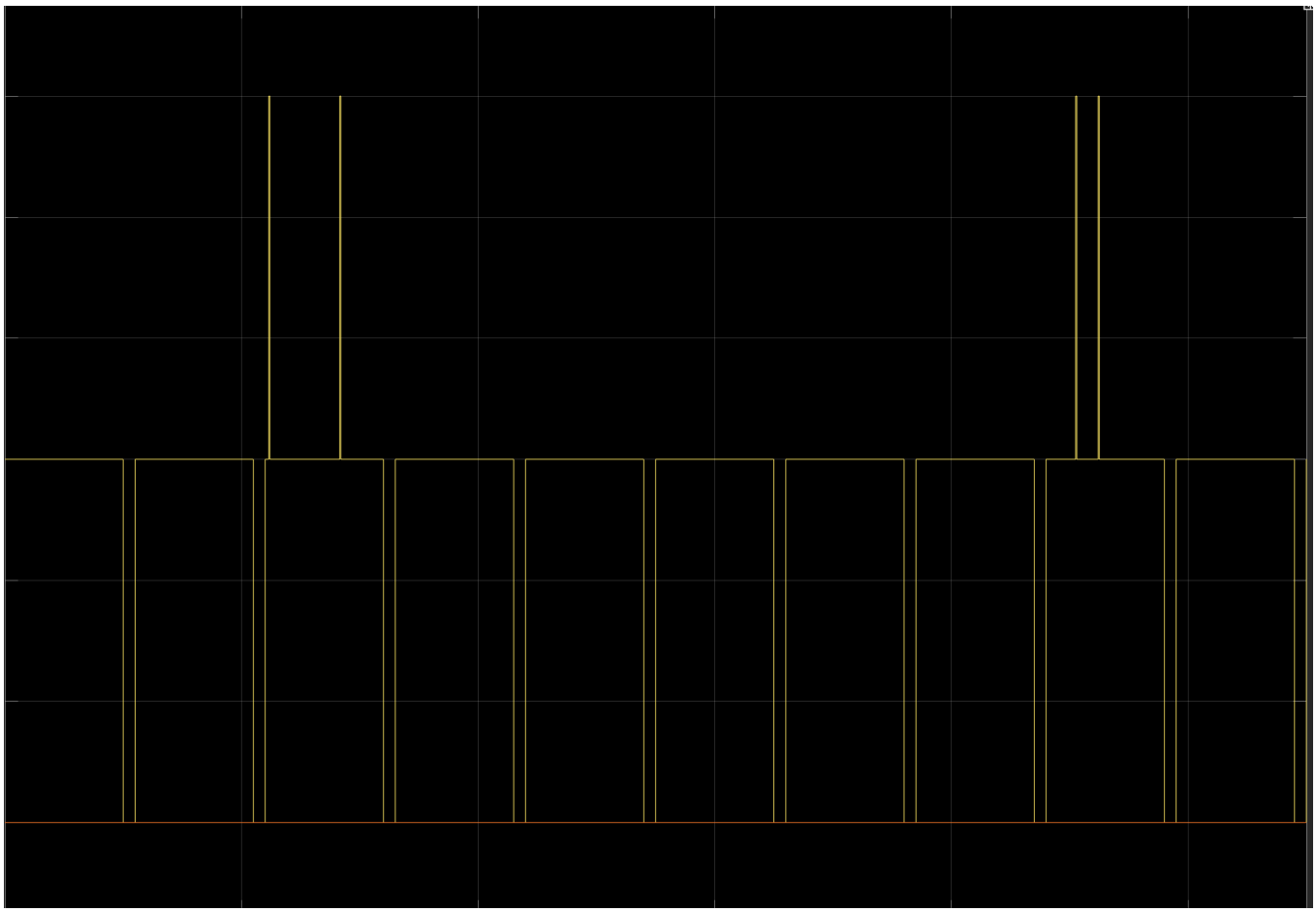


Figure 7: Plot of Decoded signal



7. Explain, step-by-step, how the DTMF decoding algorithm presented in

   $$DTMF.mdl$$

   works.

   The input signal which should be a signal that has already been converted through the dtmfdial function is loaded from the workspaces using the from workspace block. This data is correlated with a pure signal generated from one of the 7 frequencies to find a match. The correlation block takes two inputs in the time or frequency domain. The ouput of the correlation goes through a minmax block that has been set to max to find the area or data points of maximum overlap. For multiple inputs, operators are applied across the inputs. The same kinds of frequency correlations as to row frequency or column frequencies are then muxed together by combine scalar or vector signals of the same data type and complexity into a virtual vector without concatenating the signals. the output of the mux goes through a quick sort in descending order using Value and index of sorted elements in vector or matrix. The values are terminated and the indexes demuxed. The matching indices are sent to a 2 way direct lookup to identify the matching frequencies in order to generate an output signal that is sent to the scpe. Failing indices are then terminated.

8. When you used the Scope to decode the phone number, you should have noticed erroneous values that did not correspond to the numbers you dialed. Comment on why those values occur and how you could improve the system so that they are eliminated.

   Those were created from noise in the system and can be reduced by consistent normalization of the vectors to ensure consistency in amplitudes and thus comparisons. Additionally, the use of floats and doubles implies that approximations are made during the calculations

9. Answer all questions asked in the instructions.

4 Conclusions

In conclusion, through the lab we successfully explored the standard Dual-Tone Multi-Frequency (DTMF) phone dialing system by developing a comprehensive DTMF encoder and decoder. The encoder, implemented through MATLAB, efficiently generated tones corresponding to any phone number, showcasing the fundamental principles of DTMF signaling. Meanwhile, the decoder, constructed using a Simulink block diagram, effectively translates these tones back into their numeric representations, allowing for real-time visualization on a scope.

Through this endeavor, we gained practical experience in signal processing and also reinforced our understanding of how DTMF technology underpins modern telecommunication systems.

5 Extension