```
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
```