```
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);

        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;

    %% Signal Preprocessing ends.
    figure;
    plot(signal)

    N = length(signal);
    segment_lenght=(tone_duration+tone_pause)*fs;
    digit_number= round((N)/segment_lenght);
    row=zeros(1,digit_number);
    col=zeros(1,digit_number);
    power_high=zeros(1,4);
    power_low=zeros(1,4);

    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;


    else
        startIdx = round((i - 1)*segment_lenght) + 1;
        endIdx = round(startIdx+tone_duration*fs);

    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;
        power_low(c)=power;
     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;

        power_high(d)=power1;

        end
        [~,row(i)]=max(power_low);

        [~,col(i)]=max(power_high);
        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
```