SIGNAL PROCESSING PROJECT

Vedant Tejas 2023112018

Deepak Pandey 2023102053

Manas Inamdar 2023102052

QUESTION 1

We are using MATLAB to process the audio recordings of bird calls and classify them into predefined categories (Bird1, Bird2, Bird3).

Then we are extracting features from both reference and task audio files and applying some processing parameters and a simple threshold-based classifier for species identification.

PARAMETERS USED

Energy : $\sum(Audio\ Data)^2$

Significance: It represents the power and the audio intensity of the signal

Bird calls often vary in loudness based on their type, which is the primary characteristic. It serves as a basic yet effective feature for classification.

PARAMETERS USED

Zero-Crossing Rate (ZCR):

Significance: ZCR is the rate at which the signal crosses the zero amplitude line. It is calculated as:

$\frac{\sum |diff(audio\ data>0)|}{N}$ where N is the length of the signal.

ZCR measures the noisiness or tonal nature of the sound. Bird species with sharp, chirping calls tend to have higher ZCR values, while species with smooth, tonal calls exhibit lower ZCR.

PARAMETERS USED

Spectral Rolloff (85% energy point)

Significance Spectral rolloff is the frequency below which a specified percentage (in this case, 85%) of the total spectral energy is concentrated. Spectal Rolloff=|cumsum(X)≥0.85*sum(X)|

Spectral rolloff provides a frequency point that divides the spectrum into low and high energy regions, offering a sense of where most of the energy is concentrated.

WHY SPECTRAL ROLLOFF

While energy measures overall loudness and ZCR captures tonal/noisy characteristics, neither directly quantifies how energy is distributed across frequencies.

Spectral rolloff provides a frequency point that divides the spectrum into low and high energy regions, offering a sense of where most of the energy is concentrated.

Bird species with calls predominantly in lower frequency ranges will have lower spectral rolloff values, while species with higher-pitched calls will exhibit higher rolloff frequencies.

This is particularly useful in distinguishing between birds with overlapping features in energy or ZCR but differing vocal pitch profiles.

WHY SPECTRAL ROLLOFF

Mel Frequency Cepstral Coefficients (MFCCs)

MFCCs are compact representations of the short-term power spectrum of a sound, modeled on human auditory perception. They are calculated by mapping the spectrum to the Mel scale, taking the log of the powers, and then applying a discrete cosine transform (DCT).

MFCCs are widely used in speech and audio recognition tasks because they effectively capture the timbral qualities of a sound. In the context of bird calls, MFCCs provide a rich feature set that can represent the unique tonal and spectral characteristics of each species. Their inclusion enhances the robustness of the classification process.

WHY SPECTRAL FLUX

Spectral flux measures the rate of change in the power spectrum of the audio signal over time. It is calculated as:

$SpectralFlux = \sum(diff(X)2)$,

where X is the magnitude of the FFT.

Bird calls often exhibit dynamic changes in their spectral content. Spectral flux quantifies how rapidly the frequency components of a call change, which can help differentiate species with rapid, complex calls from those with more uniform vocalizations.

EXPLAINING THE CODE

```
% Define paths and list audio files (reference and task)
referenceFolder = "C:\Users\vedan\OneDrive\Desktop\Signals\Signals\Project_BirdRecognition\Reference";
taskFolder = "C:\Users\vedan\OneDrive\Desktop\Signals\Signals\Project_BirdRecognition\Task";

referenceFiles = dir(fullfile(referenceFolder, '*.wav'));
taskFiles = dir(fullfile(taskFolder, '*.wav'));

% Initialize arrays to store feature vectors for classification
featuresReference = [];
labelsReference = {'Bird1', 'Bird2', 'Bird3'};  % Assuming these labels are correct for reference files
```

The script retrieves all .wav files in these folders using dir. This ensures dynamic processing of any number of audio files without hardcoding file names.

EXPLAINING THE CODE

```
disp('Processing all Reference audio files...');
for i = 1:length(referenceFiles)
    % Read the audio file
    [audioData, fs] = audioread(fullfile(referenceFolder, referenceFiles(i).name));
    N = length(audioData);

    % Compute Energy
    energy = sum(audioData.^2);

    % Compute Zero-Crossing Rate (ZCR)
    zcr = sum(abs(diff(audioData > 0))) / N;

    % Compute Spectral Rolloff (85% energy point)
    X = abs(fft(audioData));
    f = (0:N-1) * (fs / N);  % Frequency axis
    cumulativeEnergy = cumsum(X);
    spectralRolloff = f(find(cumulativeEnergy >= 0.85 * sum(X), 1));

    % Compute Spectral Flux
    XDiff = diff(X);
    spectralFlux = sum(XDiff.^2);

    % Compute MFCCs (using the built-in function in MATLAB)
    mfccs = mfcc(audioData, fs);

    % Create feature vector for each file
    featuresReference = [featuresReference; energy, zcr, spectralRolloff, spectralFlux, mean(mfccs, 2)'];
end
disp('All Reference files processed successfully.');
```

Processing of the files and getting the parameters

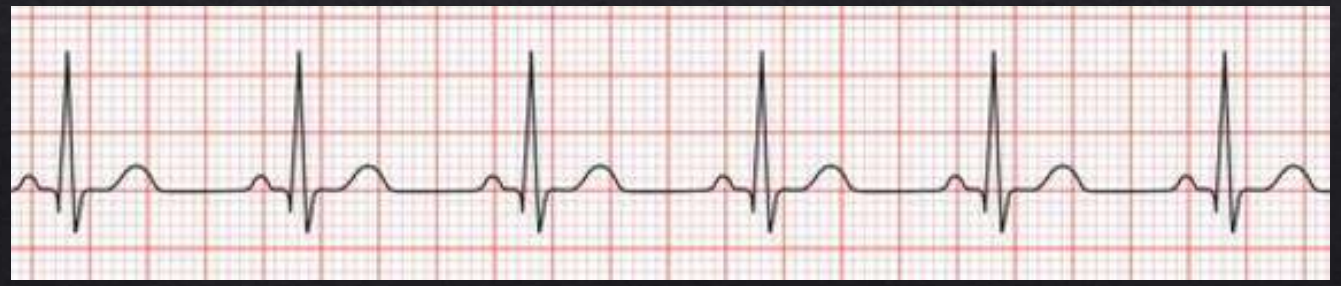
These features are stored in feature reference as a feature vector for each file

EXPLAINING THE CODE

```
% Visualize feature space for Reference files (for debugging and tuning thresholds)
figure;
scatter(featuresReference(:, 1), featuresReference(:, 2), 'filled'); % Energy vs ZCR
xlabel('Energy');
ylabel('Zero Crossing Rate');
title('Feature Space of Reference Files');

```

Plot energy vs. ZCR to visualize how the reference files are distributed in feature space
The scatter plot helps identify clusters or patterns in the features. This aids in understanding how well the chosen features separate bird species and in debugging classification thresholds.

Part 2:
Heart rate estimation



INTRODUCTION

Electrocardiogram (ECG) signals are heart signals (electrical activity of the heart) routinely used to monitor heart health. Heart rate (HR) is an important parameter used for health assessment. The ECG is a quasi-periodic signal, and the HR typically varies over time based on the activity involved. Often, these signals get corrupted during the measurement process.

ECG INTERPRETATION

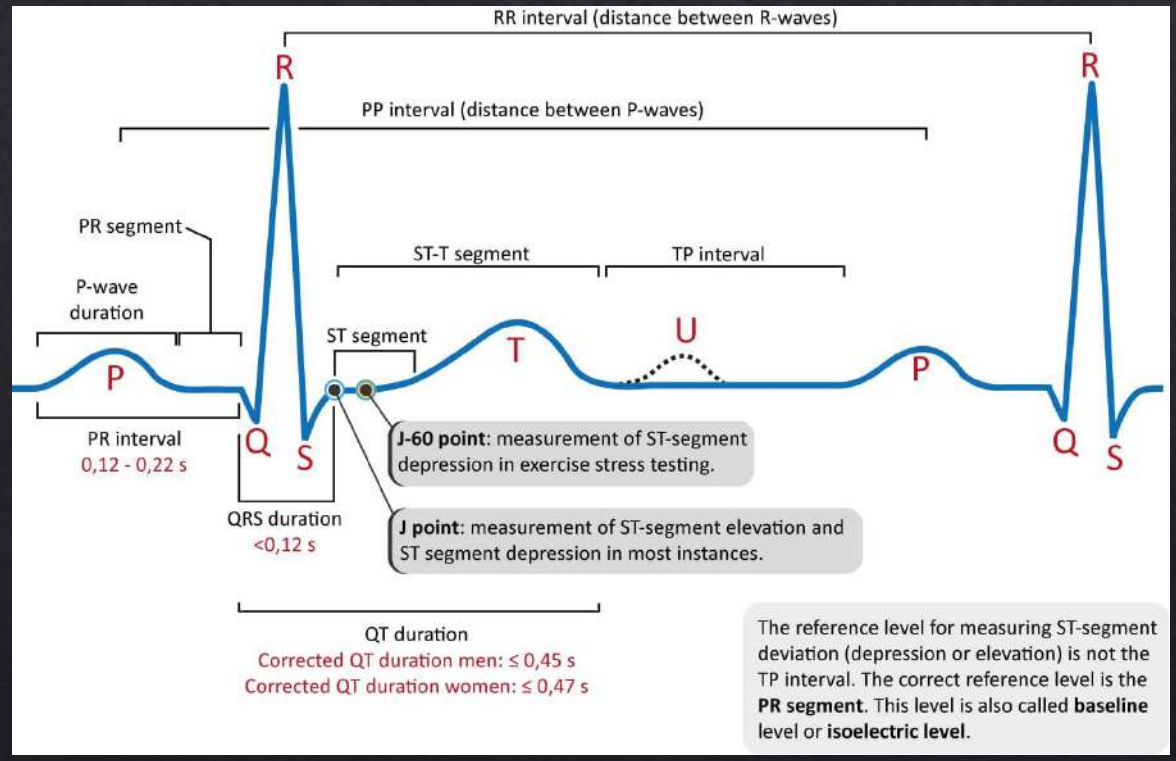
STEP 1 :
Use an ECG device to record the signal. Ensure the sampling rate is high enough to capture the signal's features.

STEP 2 :
Apply filters to remove unwanted frequencies(noise).

STEP 3 :
Detect R-Peaks.

STEP 4 :
Measure the time difference between consecutive R-peaks.

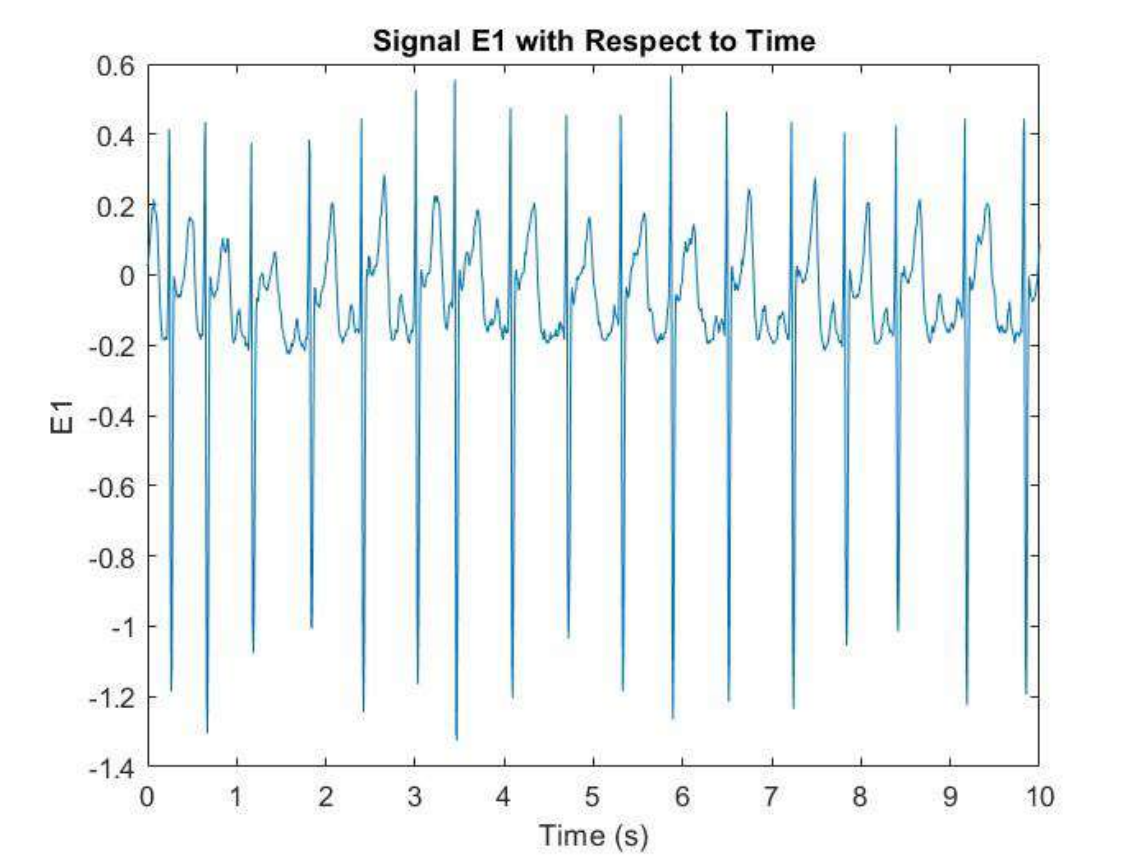
STEP 5 :
Compute the heart rate.

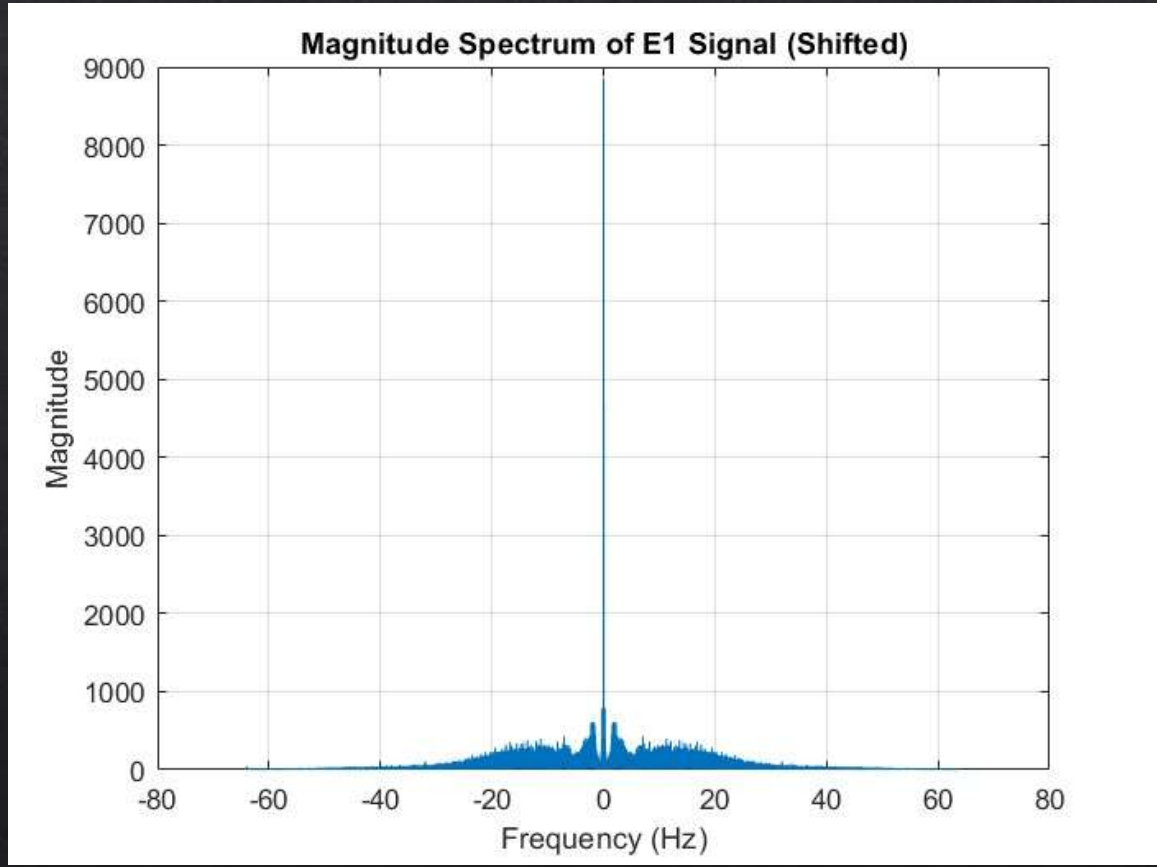


TASK-1

We are given signal E1 which is a noise free ECG signal. The sampling rate is 128 samples/second. Our objective for Task 1 is to find and plot the HR as function of time, and perform the estimation per minute.

ORIGINAL SIGNAL





SETTING PARAMETERS

Sampling Rate : 128 Hz

Threshold Value :
The threshold value has been set to 0.17 on observing the given ECG graph.

Minimum Distance between peaks :
The minimum distance between the R-R peaks has been set to 0.3 * (sampling rate) on doing the following calculations -

- **Maximum Heart Rate** ($HR_{\max}$) = 200 bpm.
- Convert this to the period (R-R interval in seconds):

$$\text{R-R Interval (s)} = \frac{60}{HR_{\max}}$$

Substituting $HR_{\max} = 200\,\text{bpm}$:

$$\text{R-R Interval (s)} = \frac{60}{200} = 0.3\,\text{seconds}.$$

If the ECG signal is sampled at a given sampling rate (f_s), the time resolution of the samples is:

$$\Delta t = \frac{1}{f_s}\ \text{seconds per sample.}$$

Thus, the **number of samples** corresponding to the minimum R-R interval is:

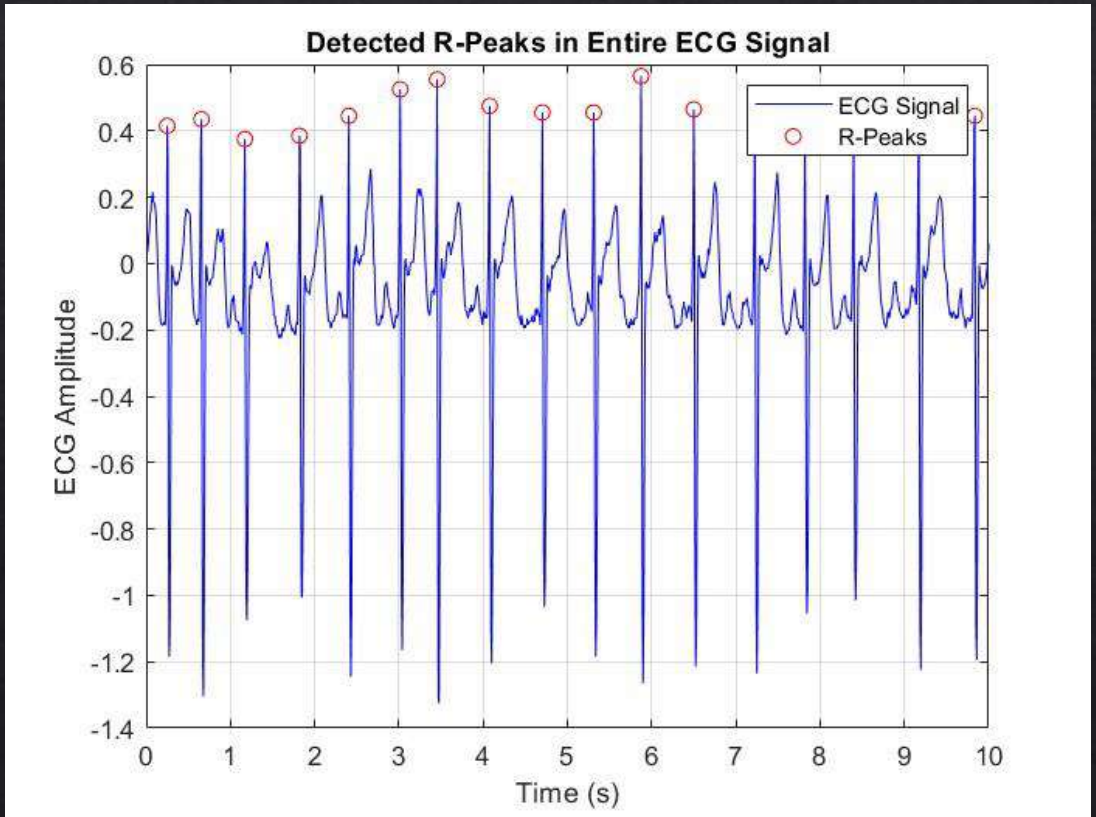
$$\text{Min Distance (samples)} = \text{R-R Interval (s)} \times f_s$$

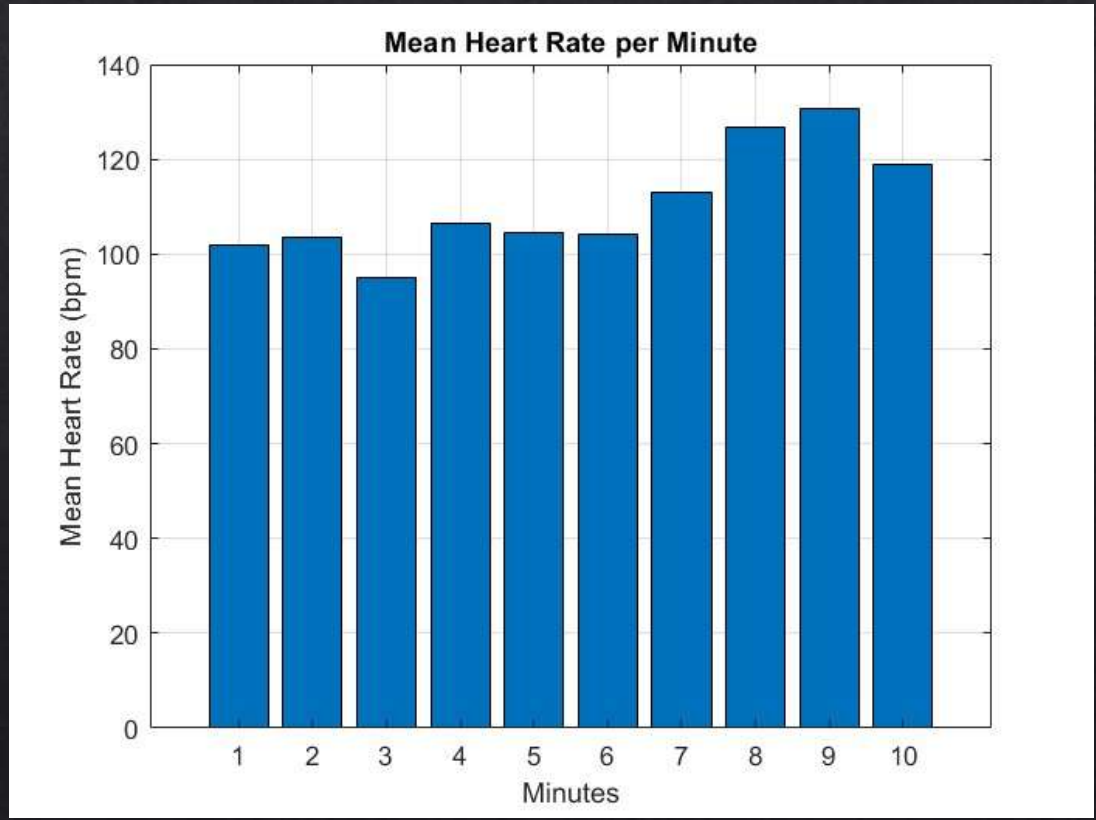
Substituting $\text{R-R Interval (s)} = 0.3$ and the sampling rate f_s:

$$\text{Min Distance (samples)} = 0.3 \times f_s.$$

RESULTS

The average Heart Rate detected is 108.72bpm

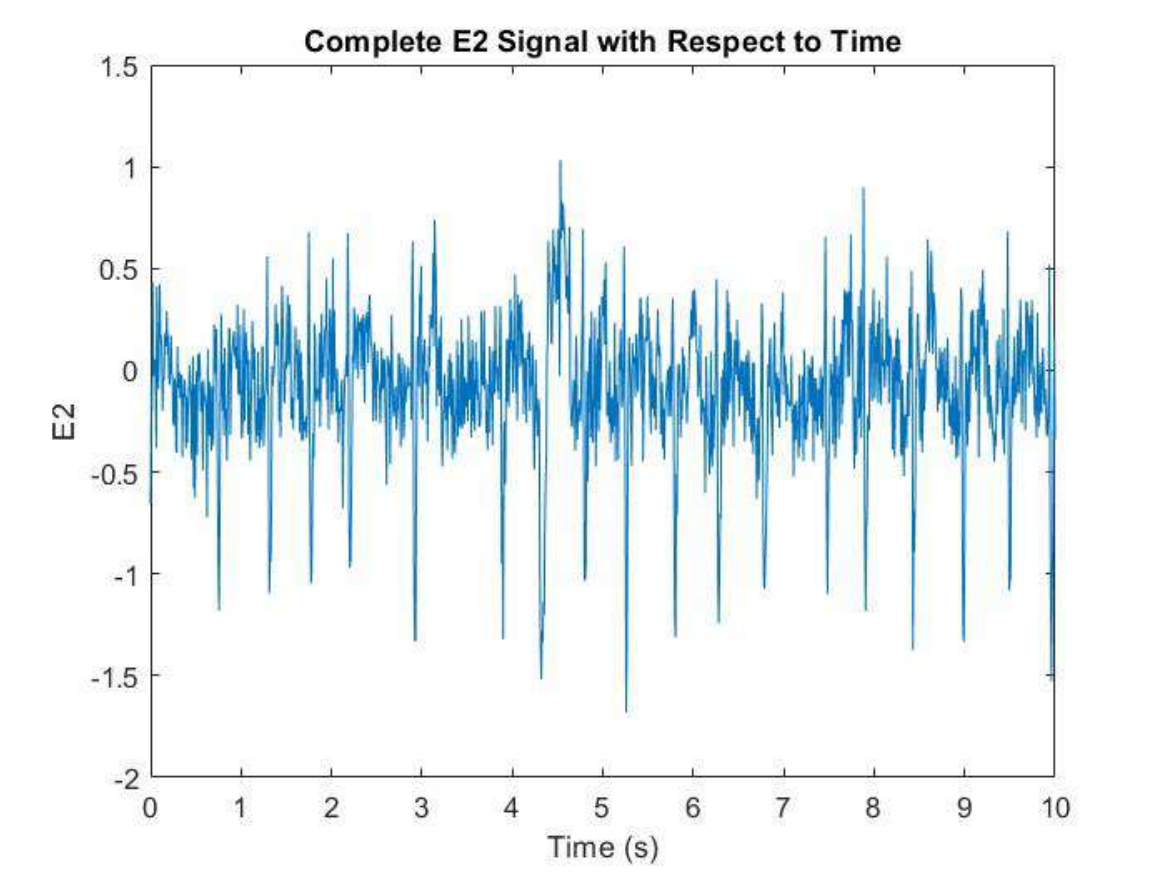


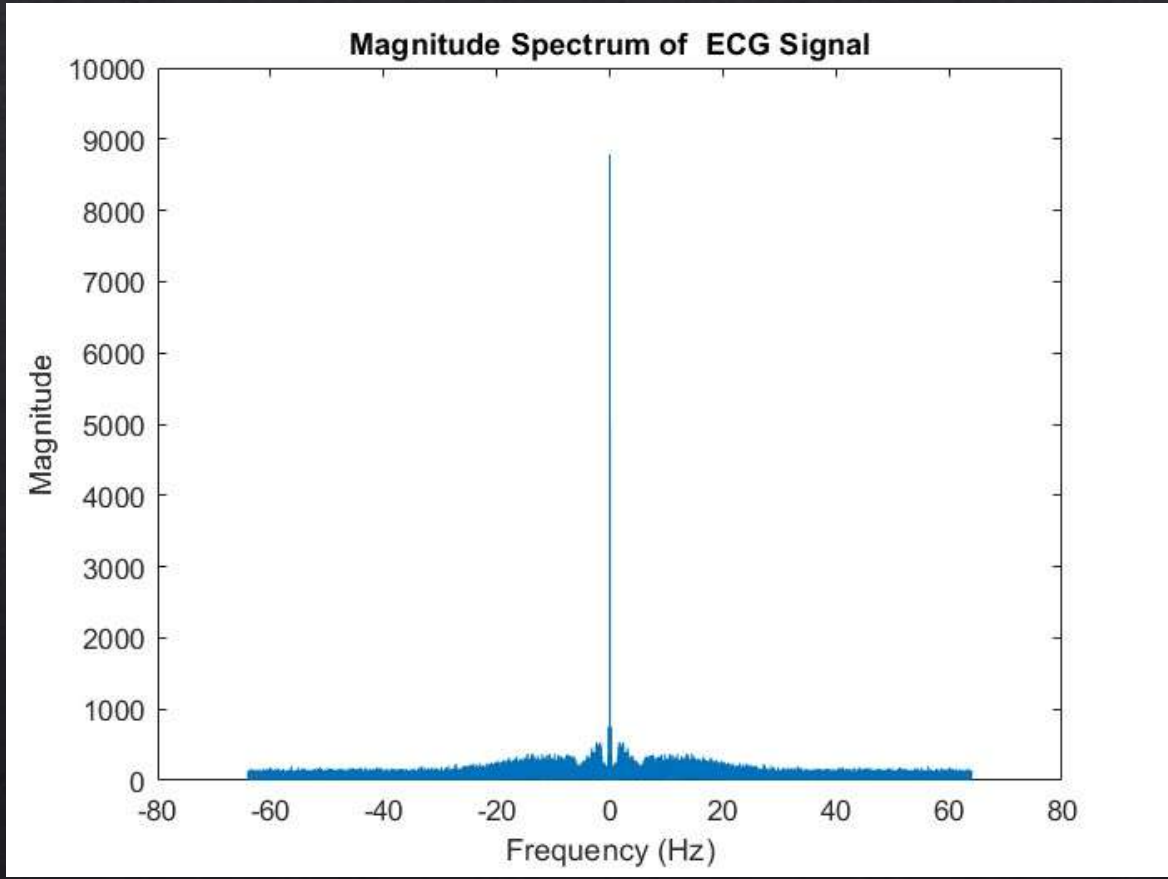


TASK-2

We are given signals E2 and E3 which are noisy ECG signals. The sampling rate is 128 samples/second. Our objective for Task 2 is to perform appropriate noise removal and find and plot the HR as function of time, and perform the estimation per minute.

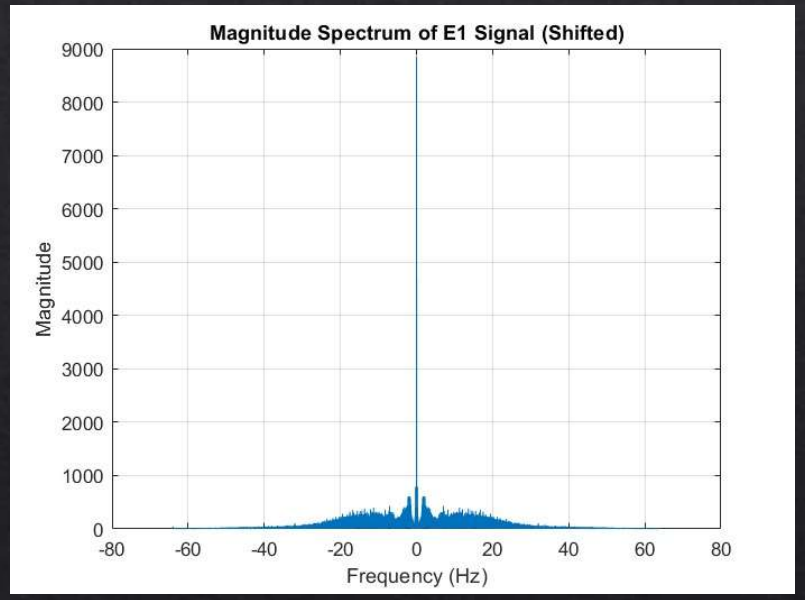
ORIGINAL SIGNAL E2

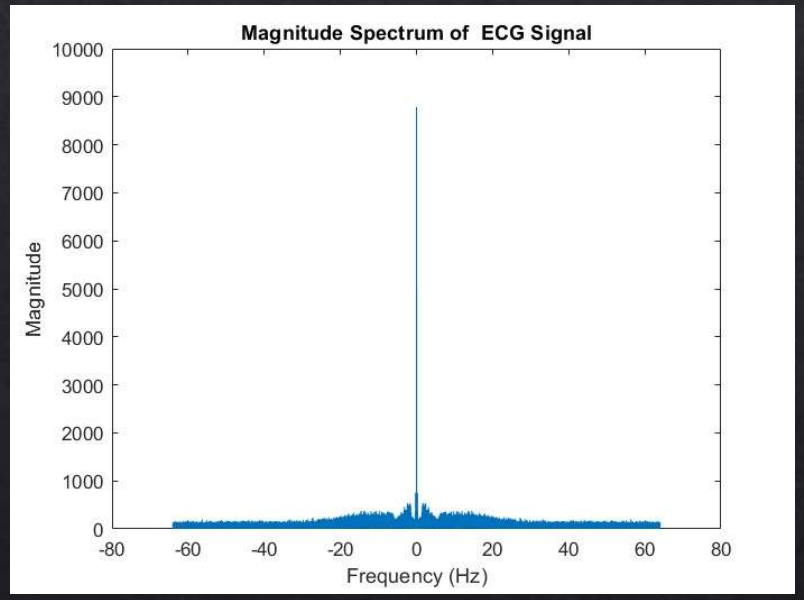




SIGNAL E2

As we can see the plots, we can observe that E2 has a lot of noise.
To eliminate noise, we have to apply filters. If we compare magnitude spectrum of noiseless E1 and noisy E2 :





We can clearly observe that we need to apply a low pass filter to eliminate frequencies above 45Hz.

SIGNAL E2

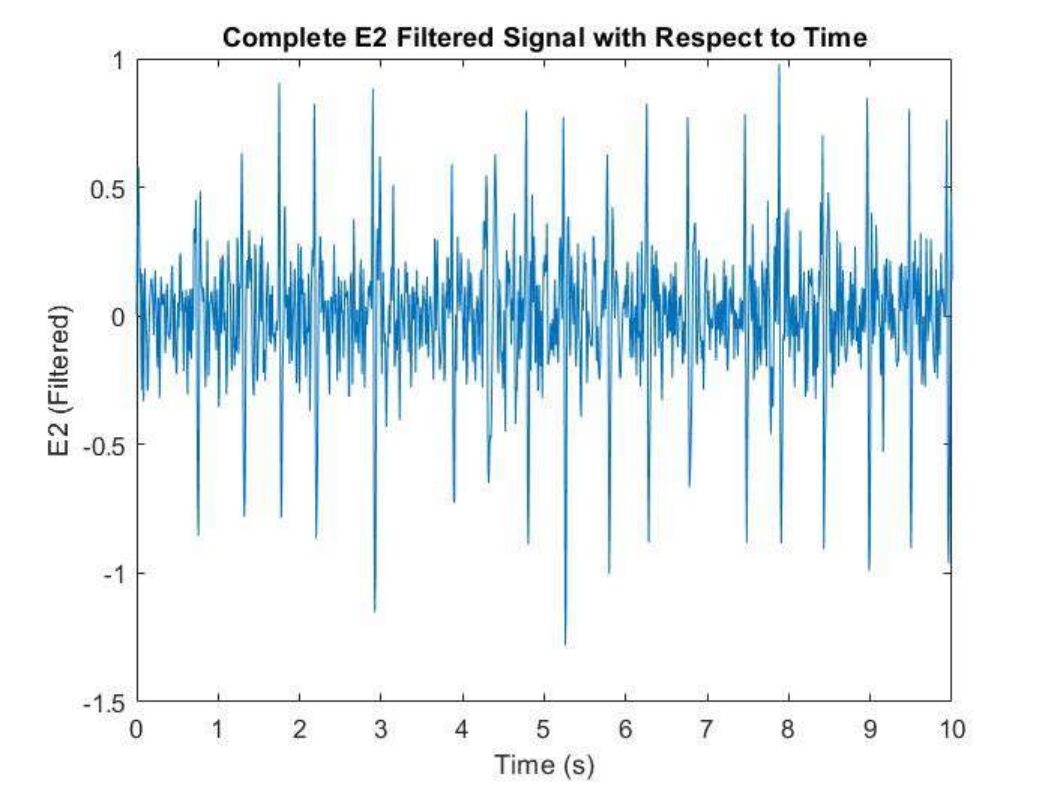
There is a problem, if we are considering frequencies 0Hz - 45Hz, we are still getting noisy time domain signal, and we are having difficulty in setting a threshold because non R peaks might be detected.

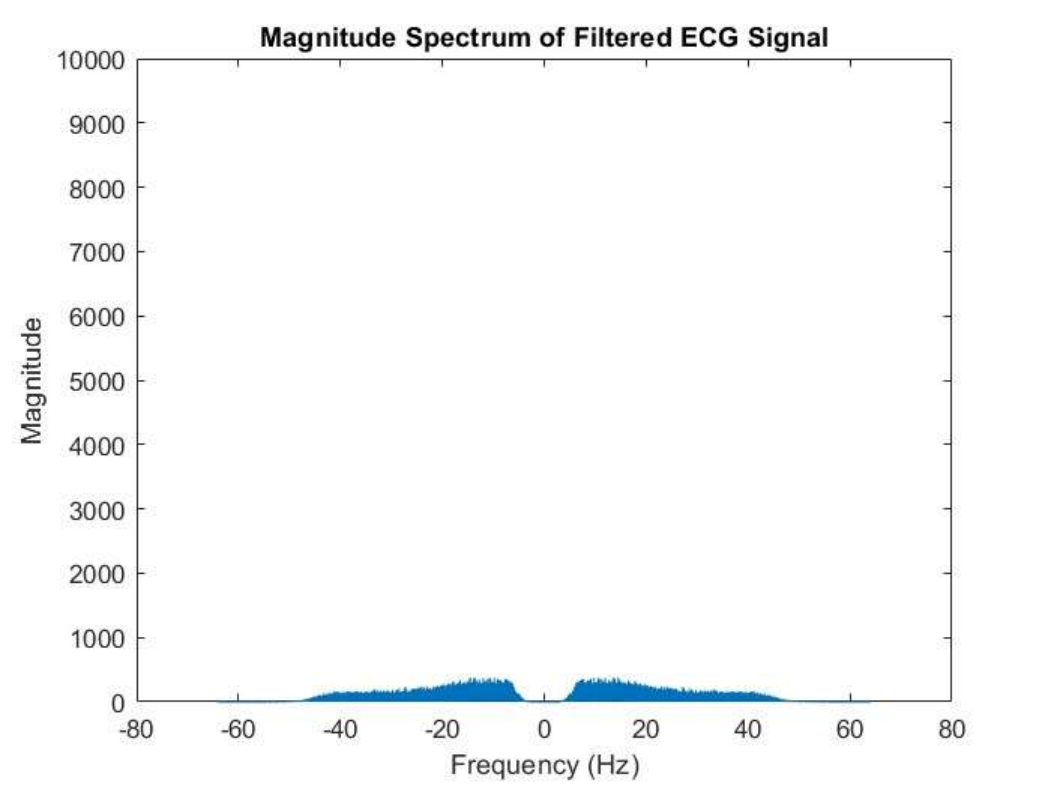
So, we have considered frequencies 5Hz - 45Hz using band pass filter. The time domain signal is still noisy but we are able to set a clear threshold.

This change reduces baseline wander and other low-frequency noise, making it easier to distinguish between true R peaks and noise.

Baseline wander is a low-frequency oscillation in signals, caused by breathing, electrode motion, or other artifacts, distorting analysis. Eliminating this can help us significantly differentiate between R peaks and noise.

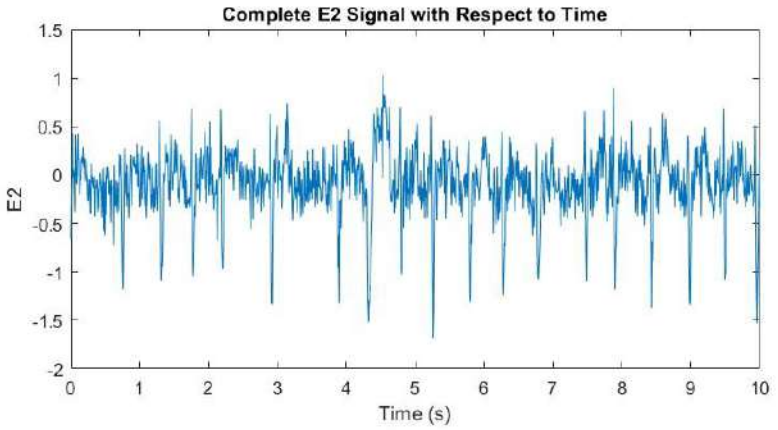
FILTERED SIGNAL

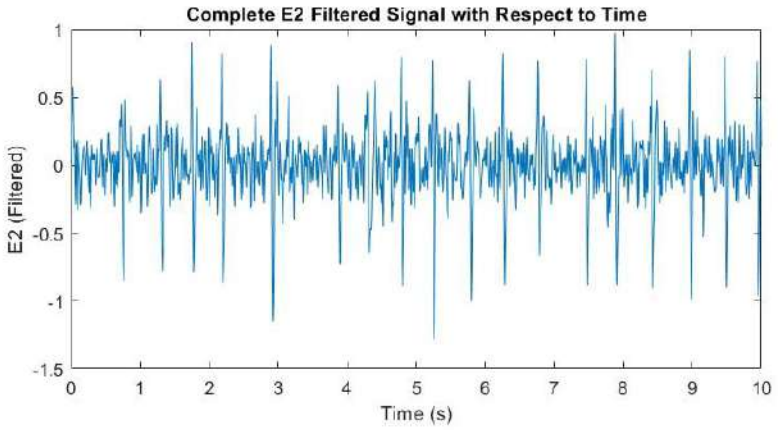


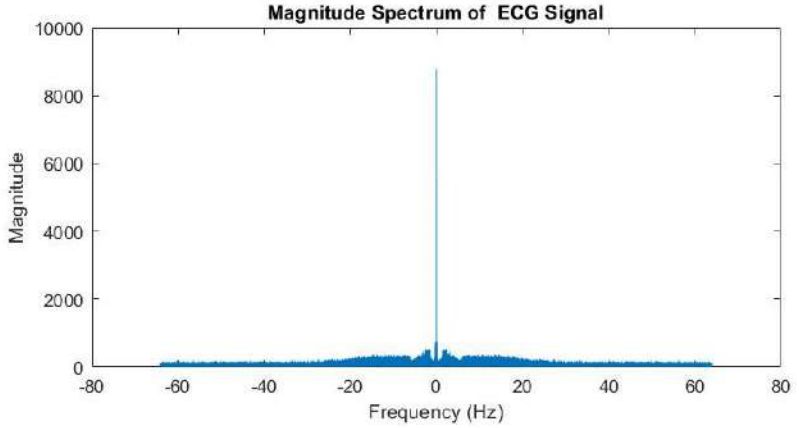


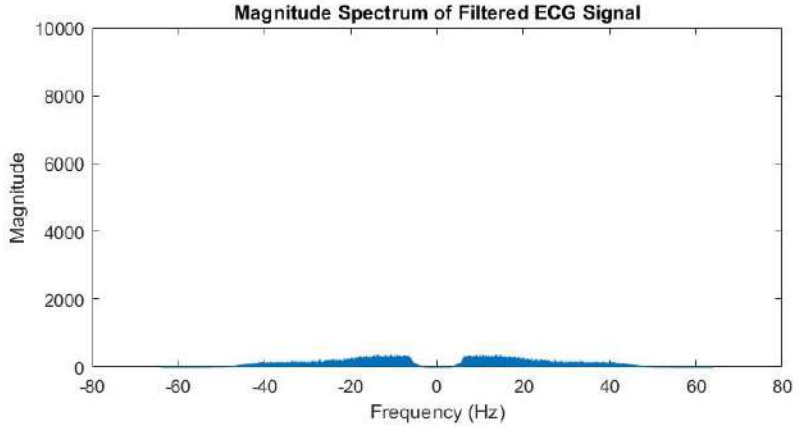
Threshold Value :
The threshold value has been set to 0.38 on observing the given ECG graph.

COMPARING BEFORE AND AFTER FILTERING PLOTS



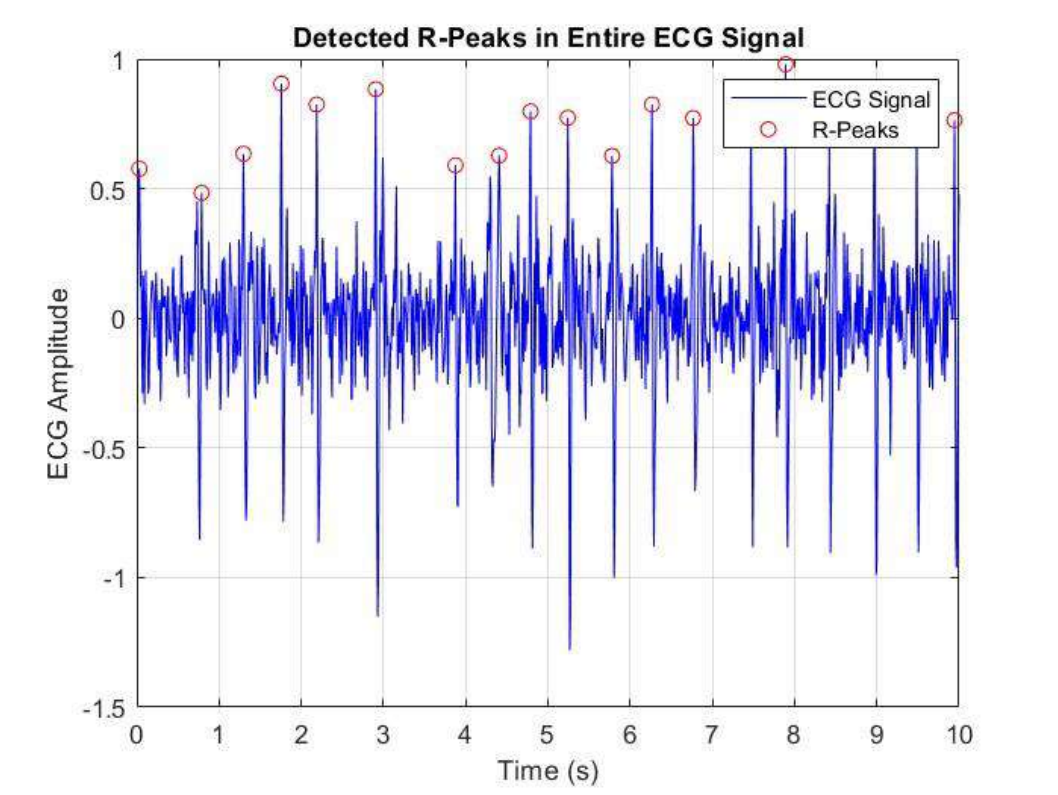


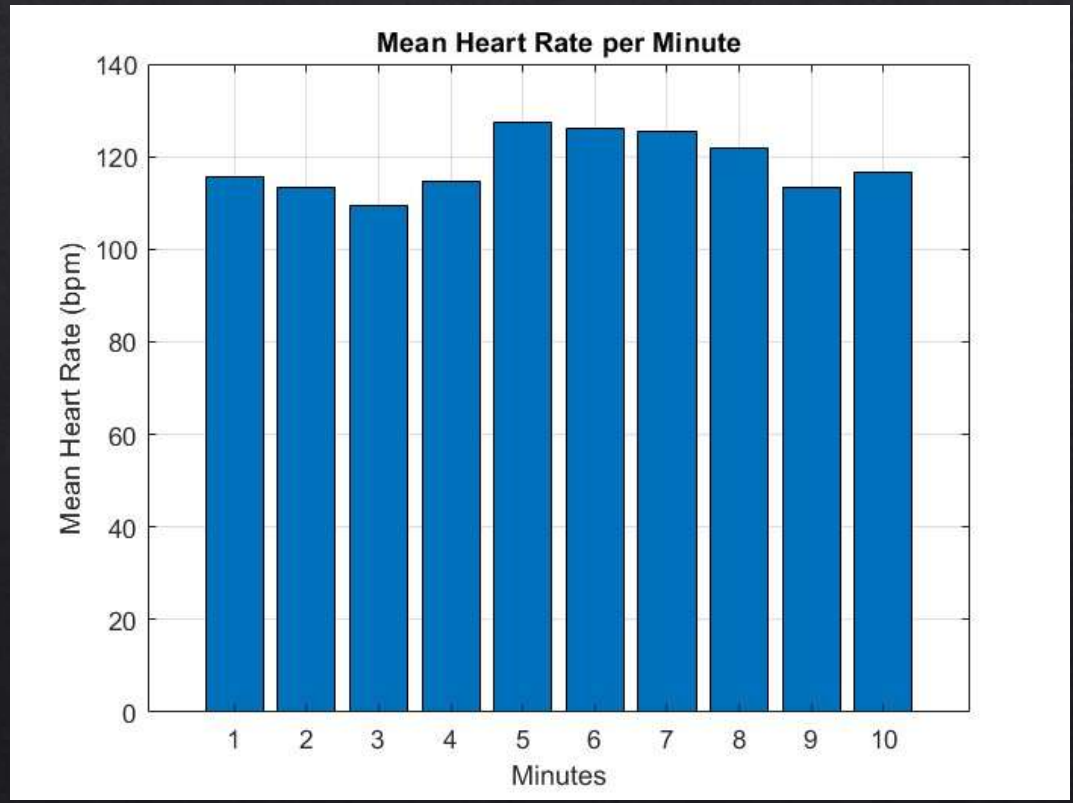




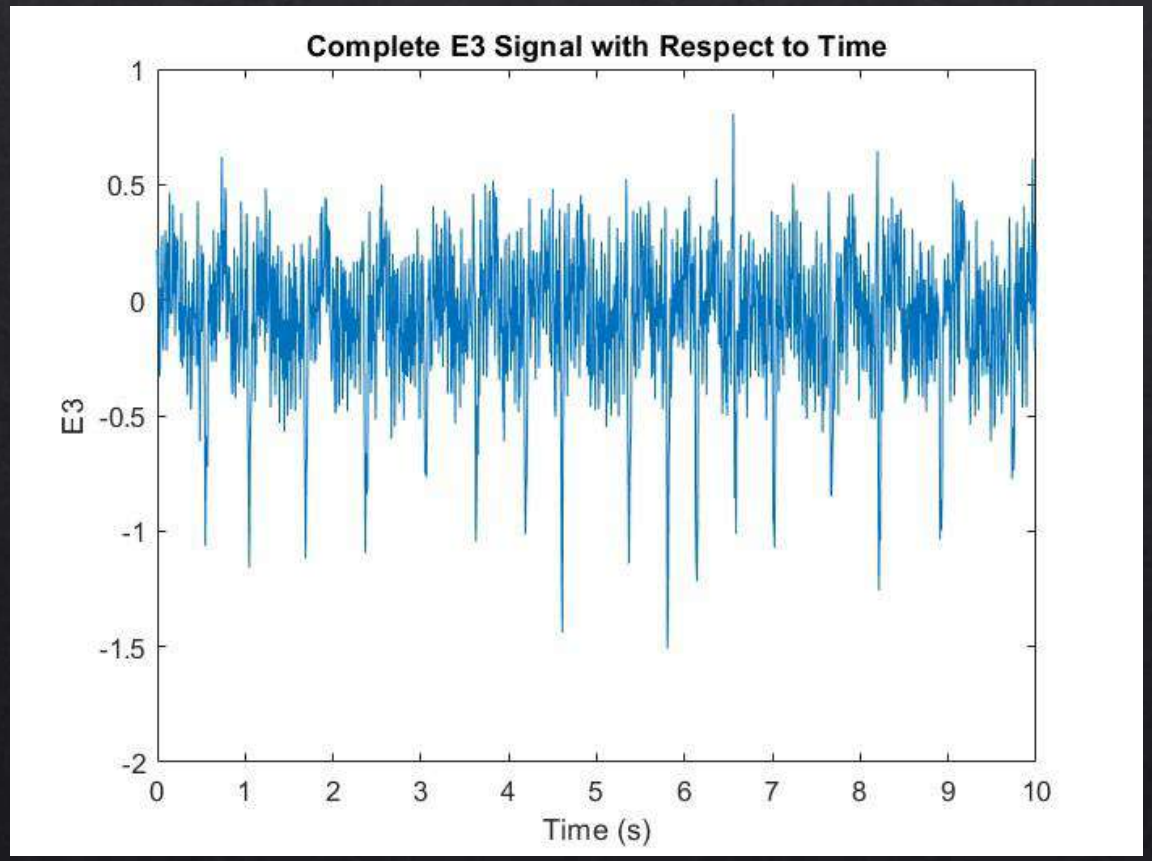
RESULTS

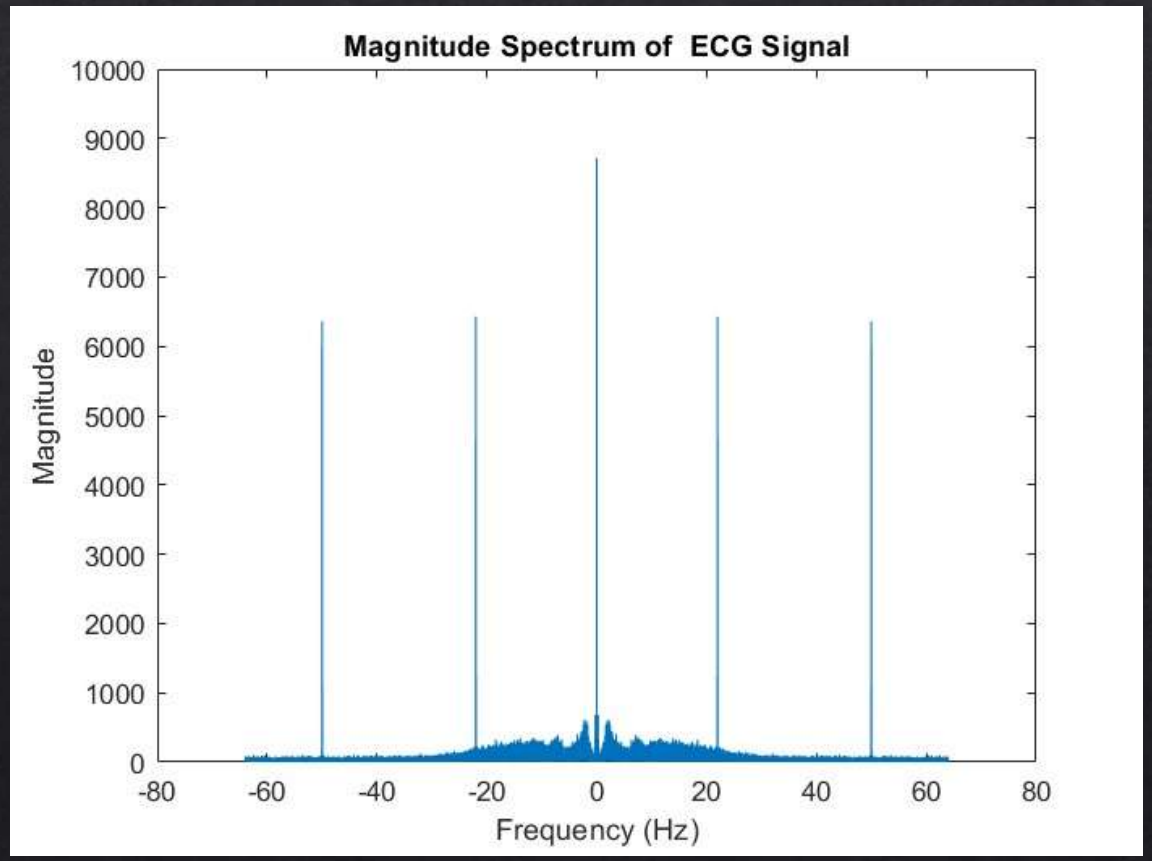
The average Heart Rate detected is 114.10bpm





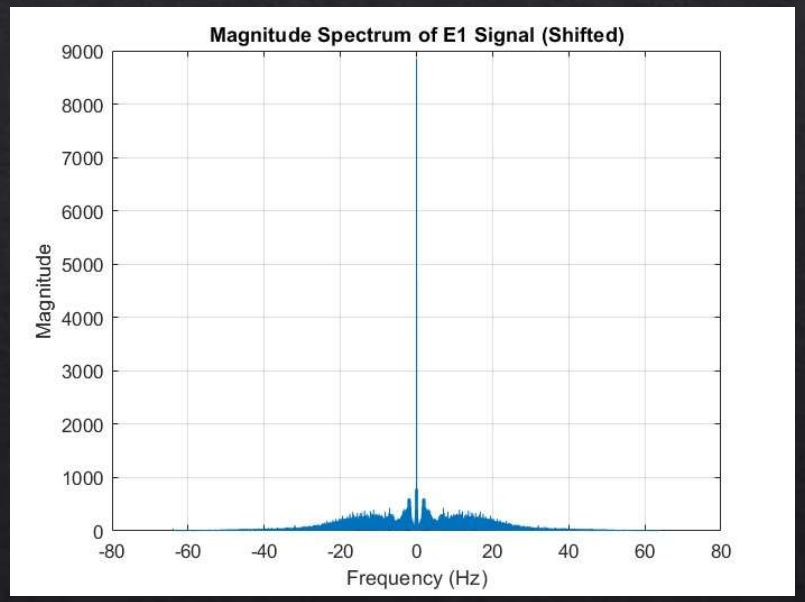
ORIGINAL SIGNAL E3

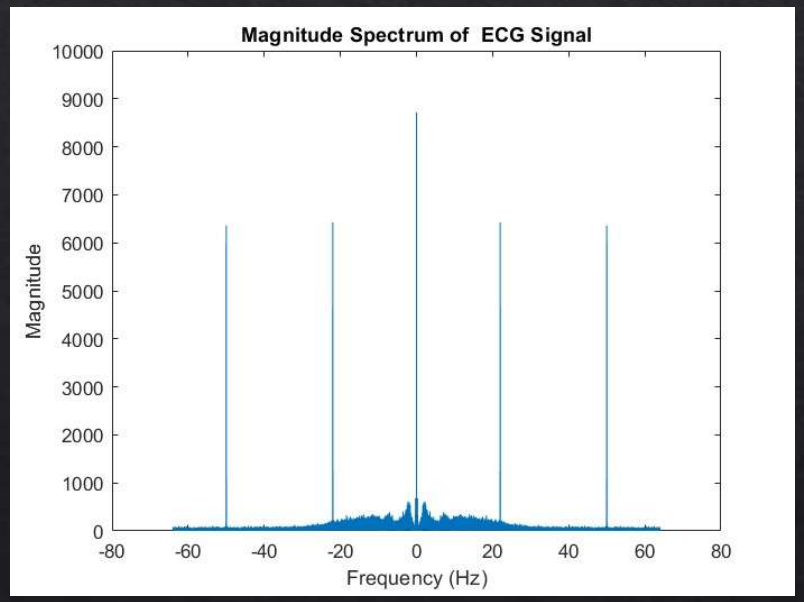




SIGNAL E3

As we can see the plots, we can observe that E3 has a lot of noise.
To eliminate noise, we have to apply filters. If we compare magnitude spectrum of noiseless E1 and noisy E3 :



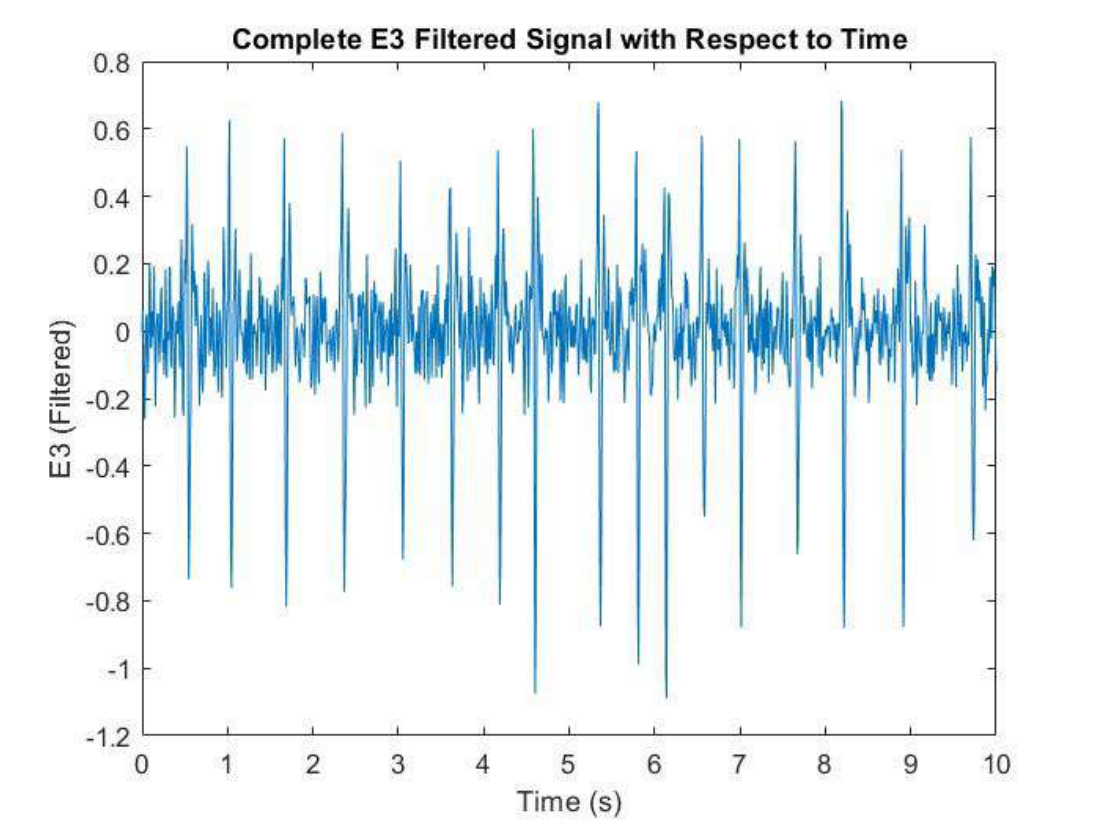


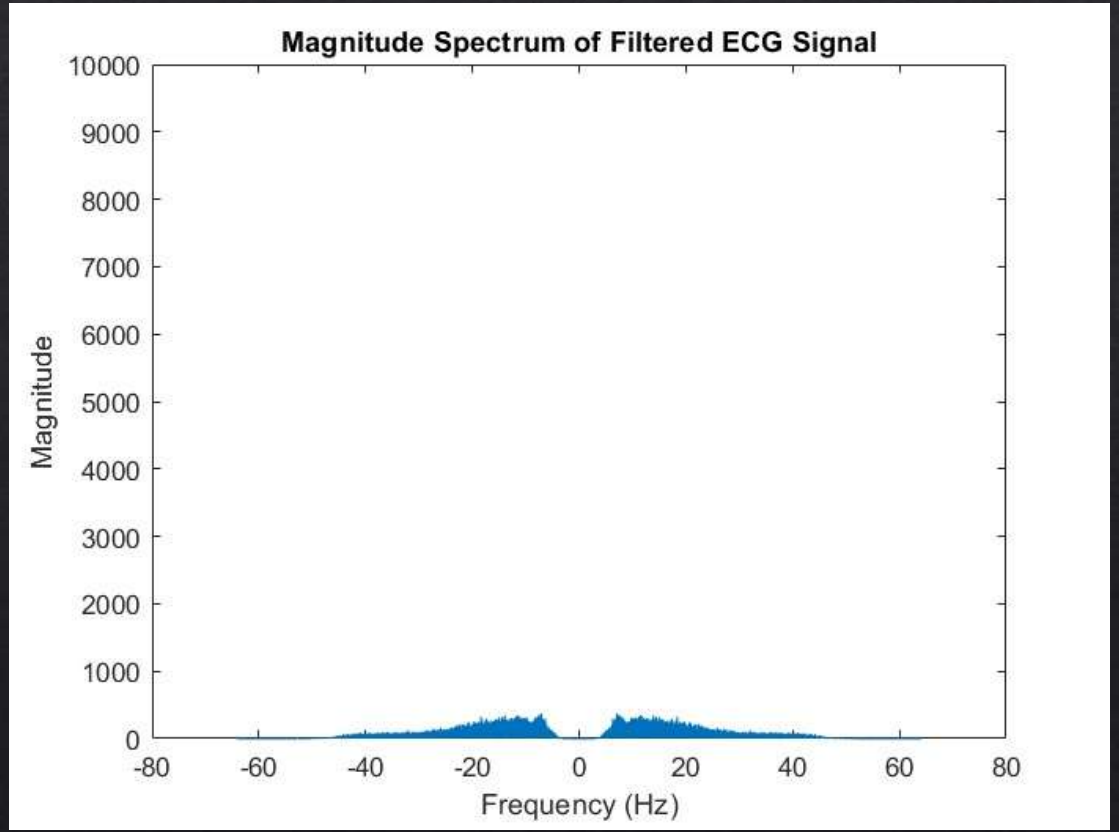
SIGNAL E3

First, we need to apply a notch filter which does not pass the abnormal noisy peaks we are getting at 22Hz and 50Hz.

Then, we will again apply a band pass filter which passes 5Hz – 45Hz.

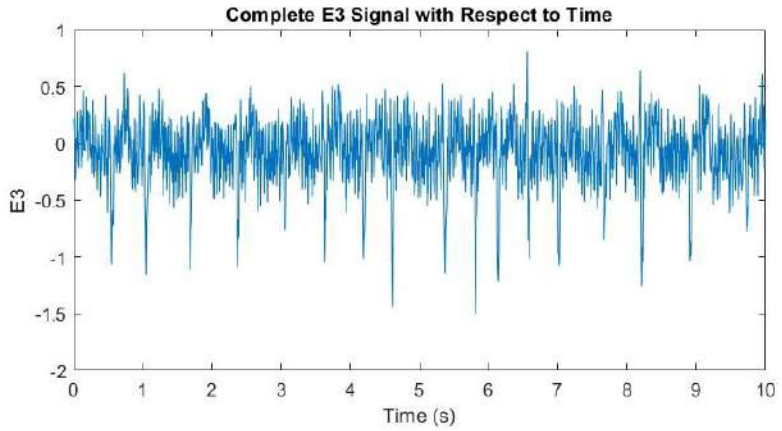
FILTERED SIGNAL

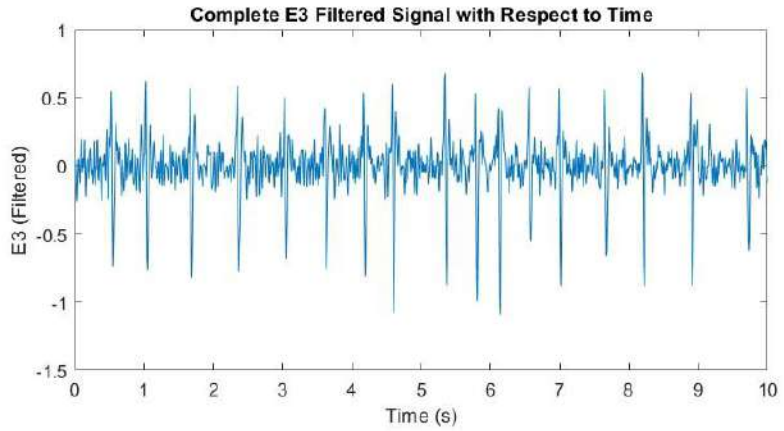


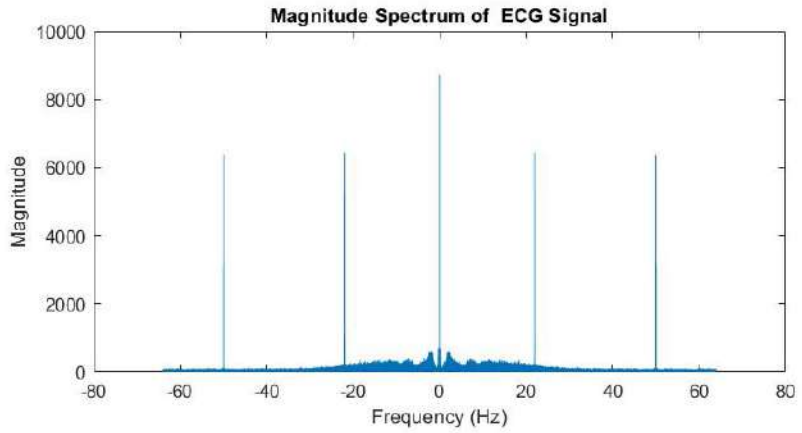


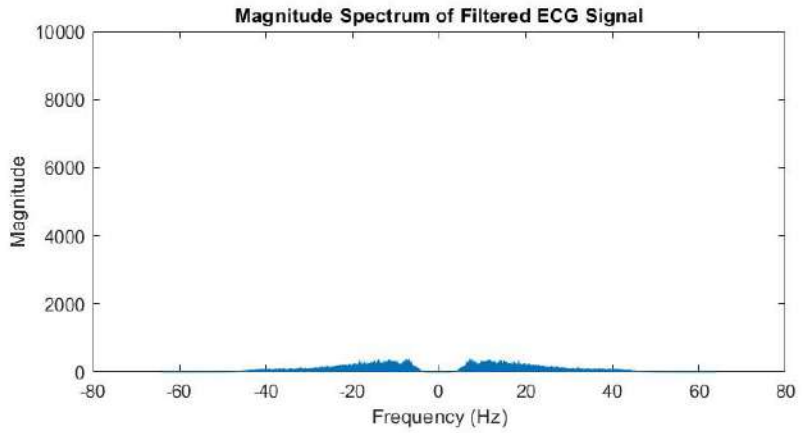
Threshold Value :
The threshold value has been set to 0.38 on observing the given ECG graph.

COMPARING BEFORE AND AFTER FILTERING PLOTS



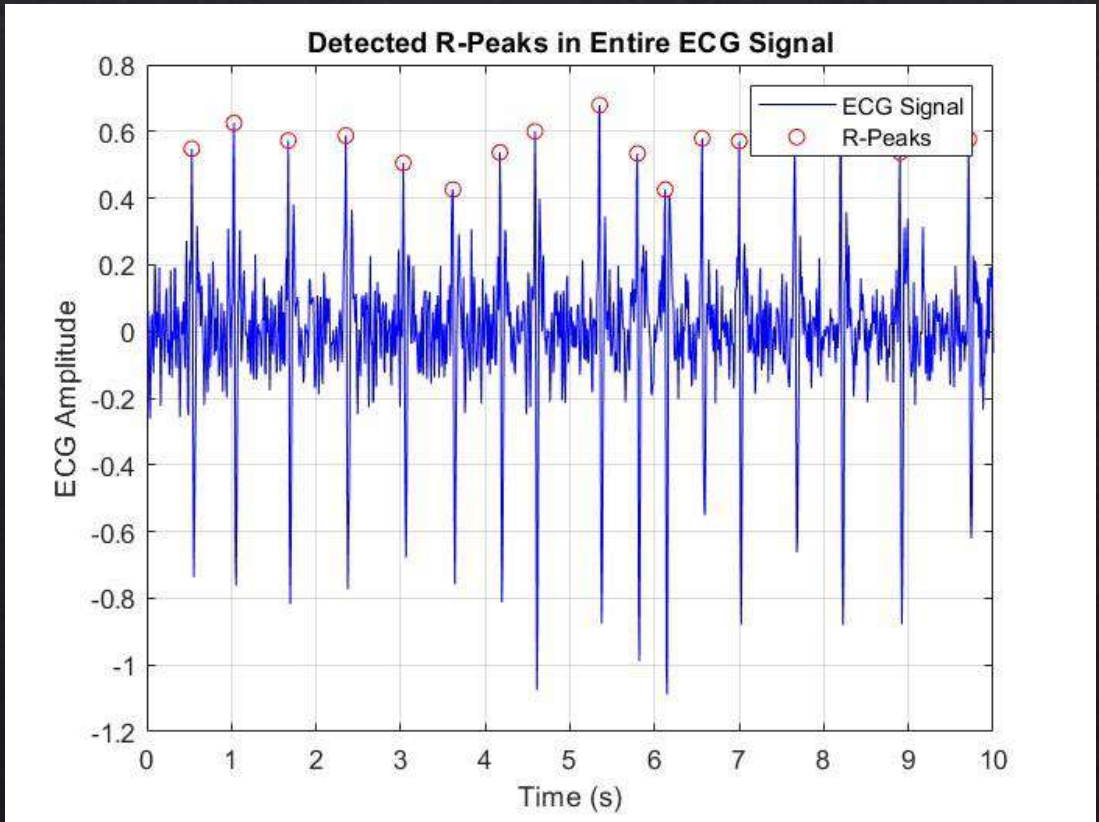


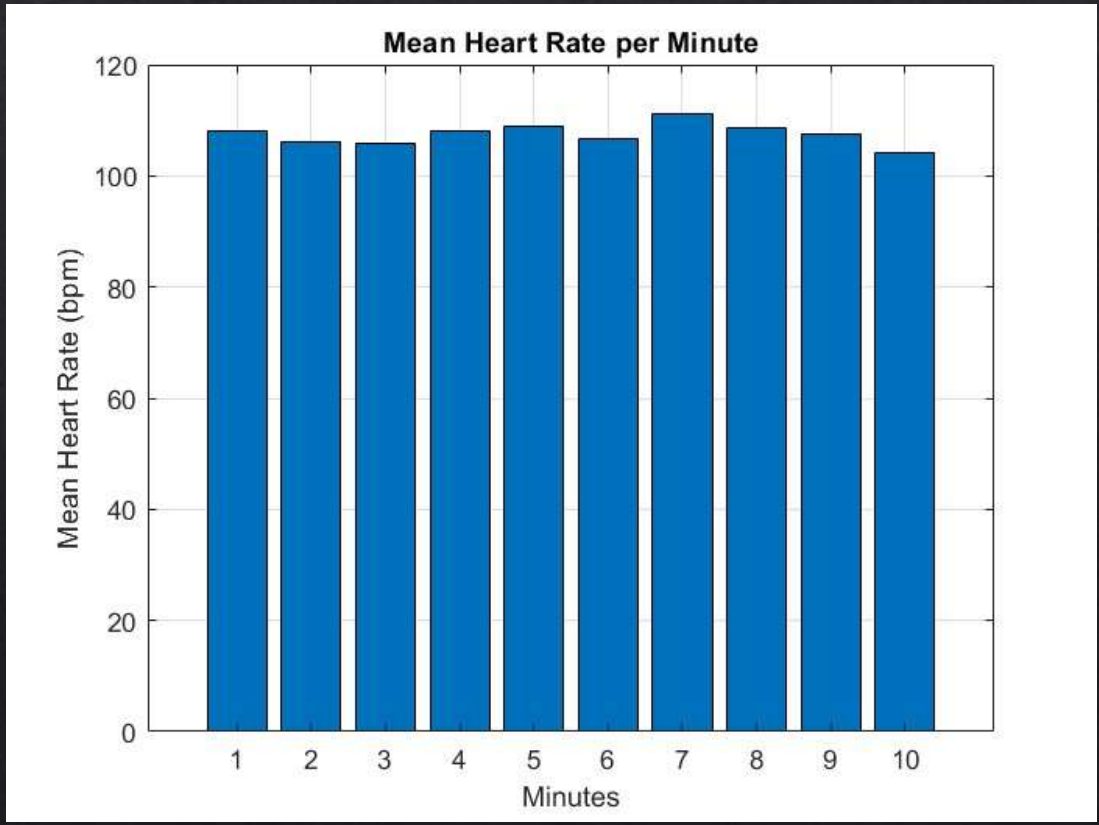




RESULTS

The average Heart Rate detected is 110.73bpm





QUESTIONS :

The Butterworth filter is preferred in signal processing due to its maximally flat passband response, ensuring no ripples and accurate signal preservation. Its smooth frequency transition minimizes distortions, making it ideal for applications requiring clean amplitude representation, such as biomedical signals, audio, and control systems, where signal integrity is critical.

- `filtfilt`: This is a MATLAB function that applies a digital filter to a signal in both the forward and reverse directions. This process,

the spectrum. The `fftshift` function rearranges the output of the FFT so that the zero frequency component is in the middle of the array,

The line of code you provided uses the `findpeaks` function in MATLAB to detect peaks in the `E1` signal. Here's a detailed explanation of each part of the line:

```
[pks, locs] = findpeaks(E1, 'MinPeakHeight', threshold, 'MinPeakDistance', min_distance);
```

- `findpeaks`: This is a MATLAB function used to identify local maxima (peaks) in a signal. It returns the values and locations of the peaks that meet certain criteria.
- `E1`: This is the input signal in which the peaks are to be detected. In this case, `E1` is the signal being analyzed.
- `'MinPeakHeight', threshold`: This specifies the minimum height that a peak must have to be considered a valid peak. The `threshold` variable defines this minimum height. Peaks with values below this threshold will be ignored.
- `'MinPeakDistance', min_distance`: This specifies the minimum distance between consecutive peaks. The `min_distance` variable defines this minimum separation in terms of the number of samples. Peaks that are closer together than this distance will be considered part of the same peak, and only the highest peak in that range will be detected.
- `[pks, locs]`: These are the output variables that store the results of the peak detection:
  - `pks`: This variable contains the values of the detected peaks.
  - `locs`: This variable contains the indices (locations) of the detected peaks in the `E1` signal.

In summary, this line of code uses the `findpeaks` function to detect peaks in the `E1` signal that are above a specified height (`threshold`) and separated by at least a specified distance (`min_distance`). The values of the detected peaks are stored in the `pks` variable, and their corresponding locations (indices) in the `E1` signal are stored in the `locs` variable.

We could have also done noise removal in E1 because signal given is noisy(therefore we can set a very evident threshold).

PART 3

LOUDNESS SEGMENTATION

Introduction -

Speech is a natural mode of communication to convey the feelings and intentions. These intentions will be passed by changing loudness of the words spoken in one's speech. Often, the words in speech have strong perceptual boundaries however similar strong indications can't be observed in the speech signal.

Signals -

You are given different speech samples and text files. The text files contain information of spoken words, its start and end time and markings indicate louder (1) or not (0) for each speech sample.

e.g. -

1	i	0.452592	0.646911	0
2	cant	0.646911	1.003572	1
3	believe	1.003572	1.340556	0
4	we	1.340556	1.426647	0
5	are	1.426647	1.542255	0
6	actually	1.542255	2.088316	1
7	going	2.088316	2.540908	0
8	to	2.540908	2.789341	0
9	paros	2.789341	3.360000	1

Task 1 -

For given speech samples, find the louder words using its respective start and end times.

Approach

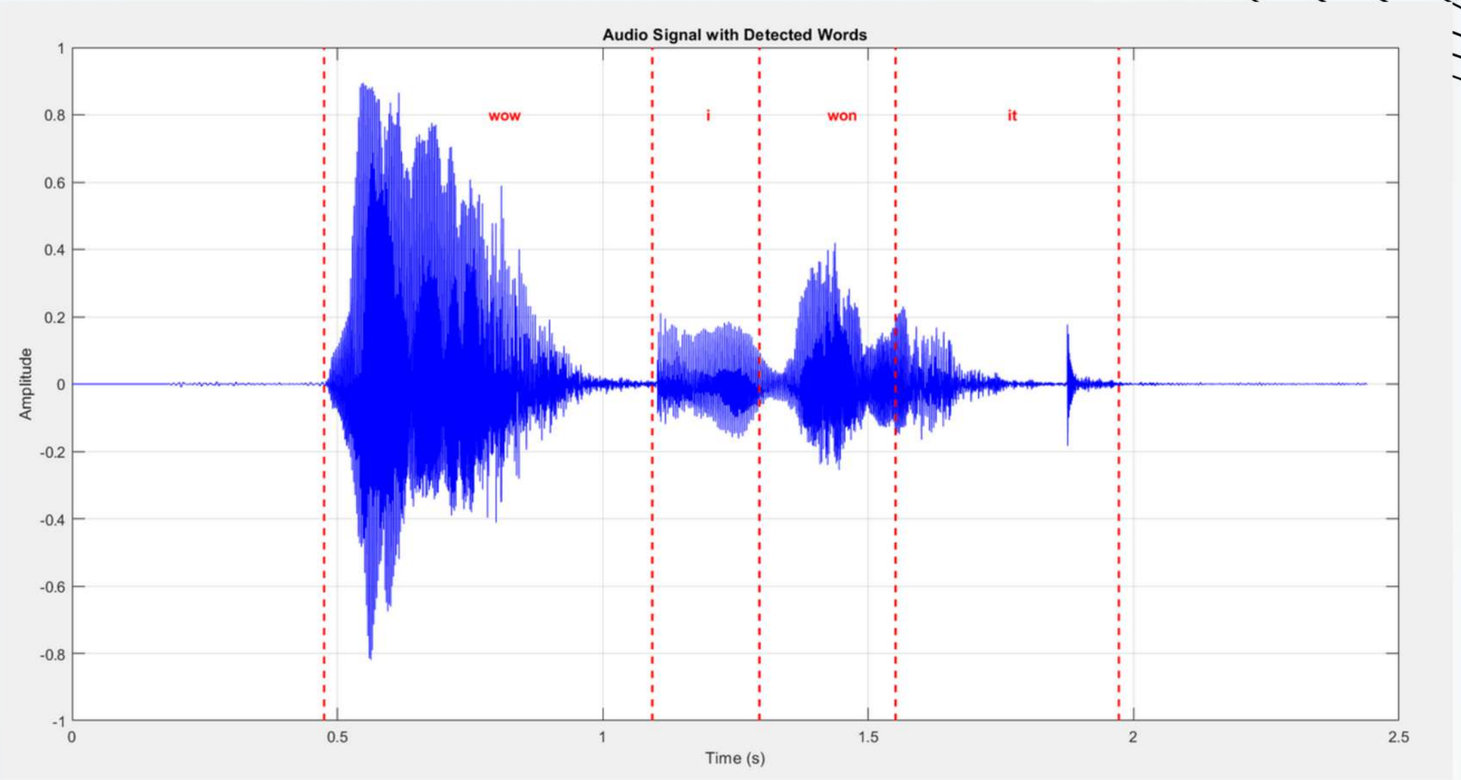
To compare loudness, we need to evaluate the RMS amplitude of all word segments, as RMS is proportional to energy, which determines loudness. We will then set a threshold to identify whether a segment is sufficiently loud.

Here, we have assumed the Threshold = 0.8 x maxRMS

Let’s now test our approach in the given speech samples and see what’s the efficiency of our approach

Sample 1-

1	wow	0.475139	1.093177	1
2	i	1.093177	1.295022	0
3	won	1.295022	1.551347	0
4	it	1.551347	1.972006	0

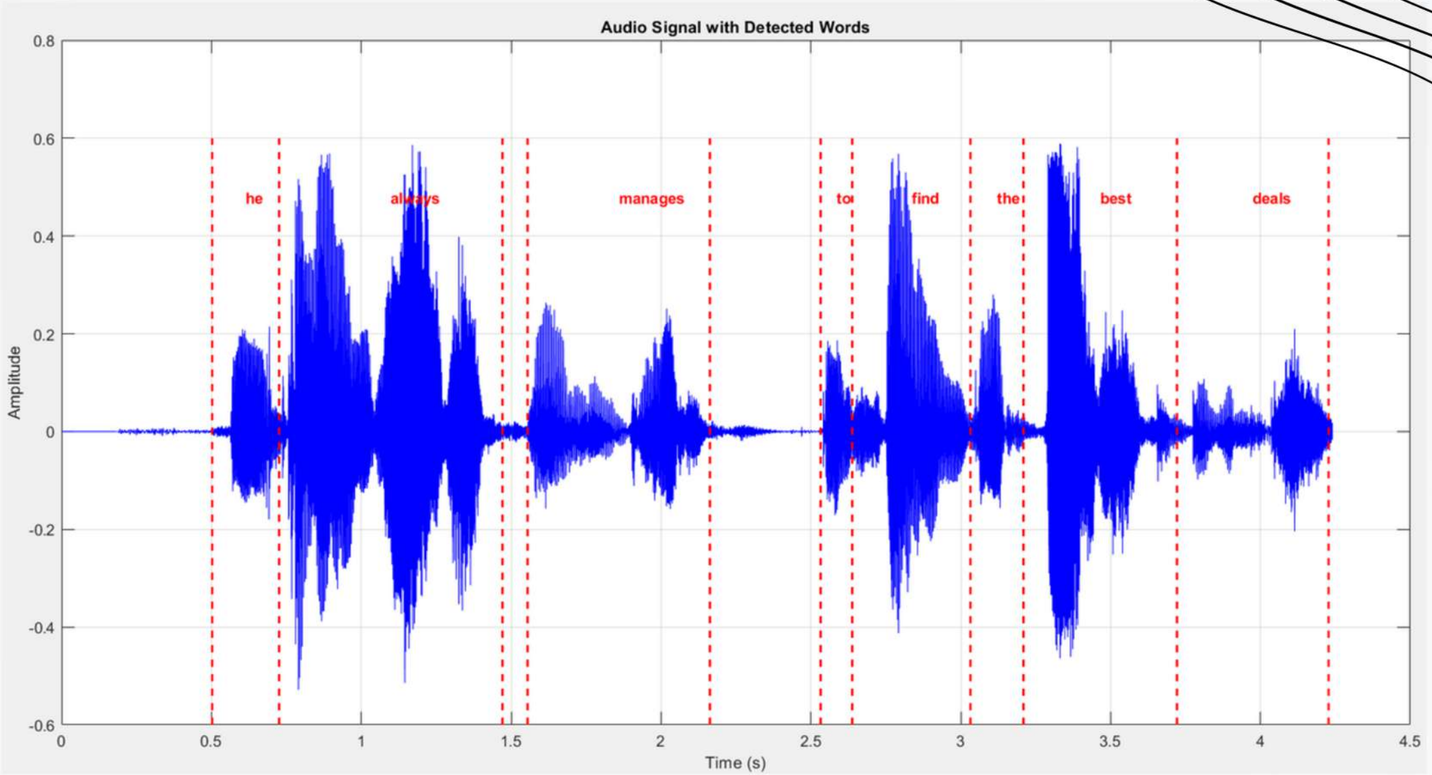



```
RMS values of each segments -
wow - 0.1955
i - 0.0642
won - 0.0940
it - 0.0396
```

```
Words louder than the threshold -
wow - 0.1955

Words which are actually louder -
wow - 0.1955
```

Sample 2-

1	he	0.502840	0.726325	0
2	always	0.726325	1.471274	1
3	manages	1.555081	2.163455	0
4	to	2.532826	2.638360	0
5	find	2.638360	3.032562	0
6	the	3.032562	3.209488	0
7	best	3.209488	3.721640	0
8	deals	3.721640	4.227584	0

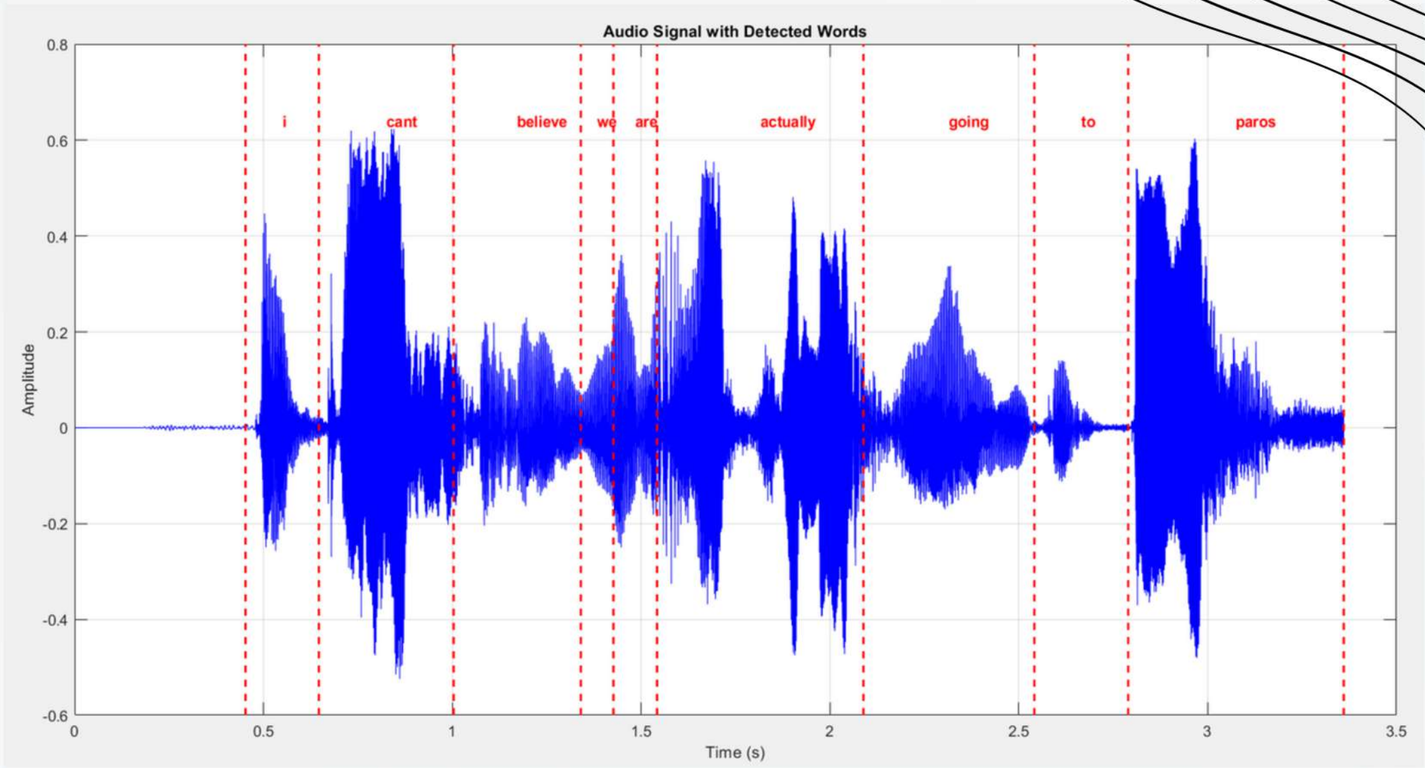



```
RMS values of each segments -
he - 0.0680
always - 0.1358
manages - 0.0514
to - 0.0659
find - 0.0982
the - 0.0677
best - 0.1190
deals - 0.0302
```

```
Words louder than the threshold -
always - 0.1358

Words which are actually louder -
always - 0.1358
```

Sample 3-

1	i	0.452592	0.646911	0
2	cant	0.646911	1.003572	1
3	believe	1.003572	1.340556	0
4	we	1.340556	1.426647	0
5	are	1.426647	1.542255	0
6	actually	1.542255	2.088316	1
7	going	2.088316	2.540908	0
8	to	2.540908	2.789341	0
9	paros	2.789341	3.360000	1

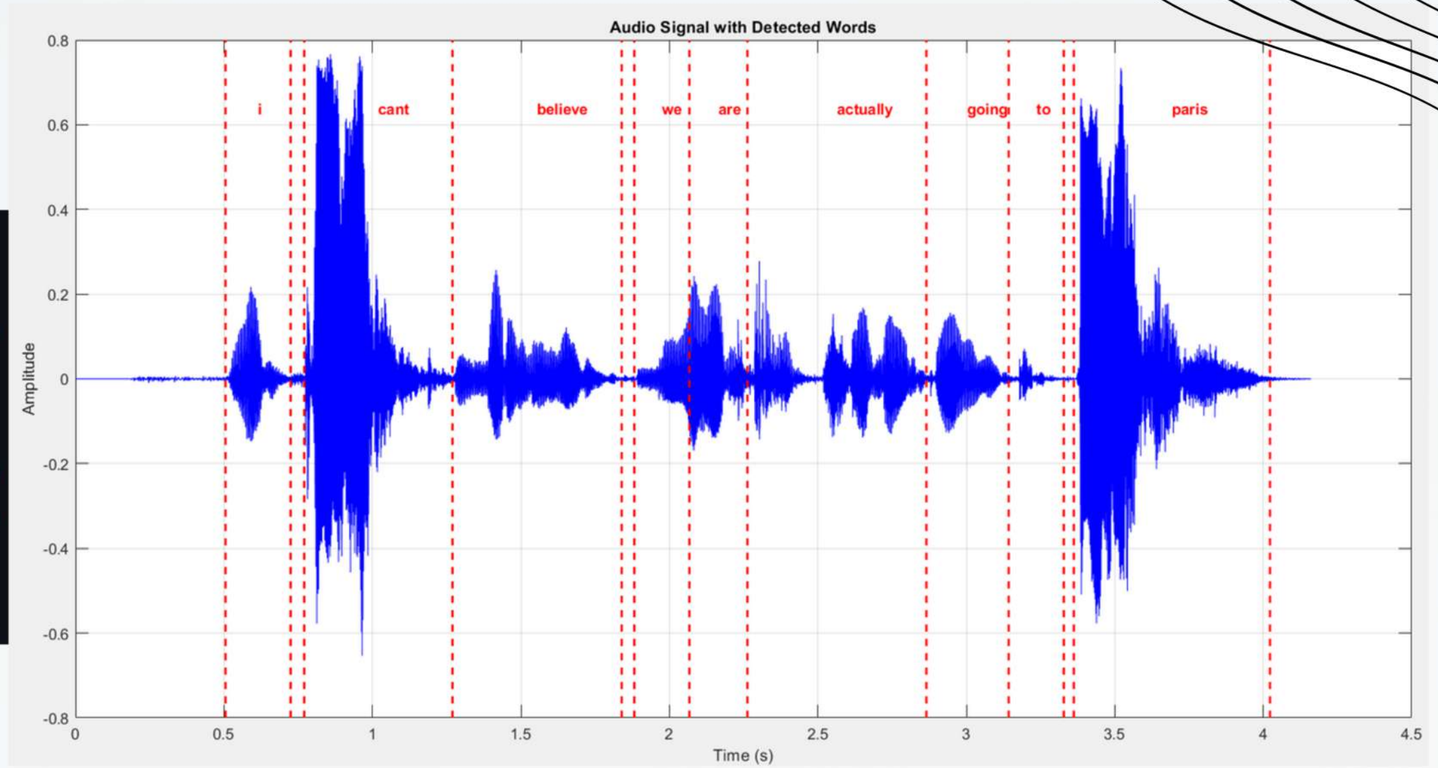



```
RMS values of each segments -
i - 0.0801
cant - 0.1786
believe - 0.0799
we - 0.0687
are - 0.1049
actually - 0.1493
going - 0.0717
to - 0.0294
paros - 0.1298
```

```
Words louder than the threshold -
cant - 0.1786
actually - 0.1493
paros - 0.1298

Words which are actually louder -
cant - 0.1786
actually - 0.1493
paros - 0.1298
```

Sample 4-

1	i	0.505534	0.724802	0
2	cant	0.770483	1.269927	1
3	believe	1.269927	1.839414	0
4	we	1.882050	2.067818	0
5	are	2.067818	2.262723	0
6	actually	2.262723	2.865710	0
7	going	2.865710	3.142840	0
8	to	3.142840	3.328609	0
9	paris	3.362108	4.022958	1

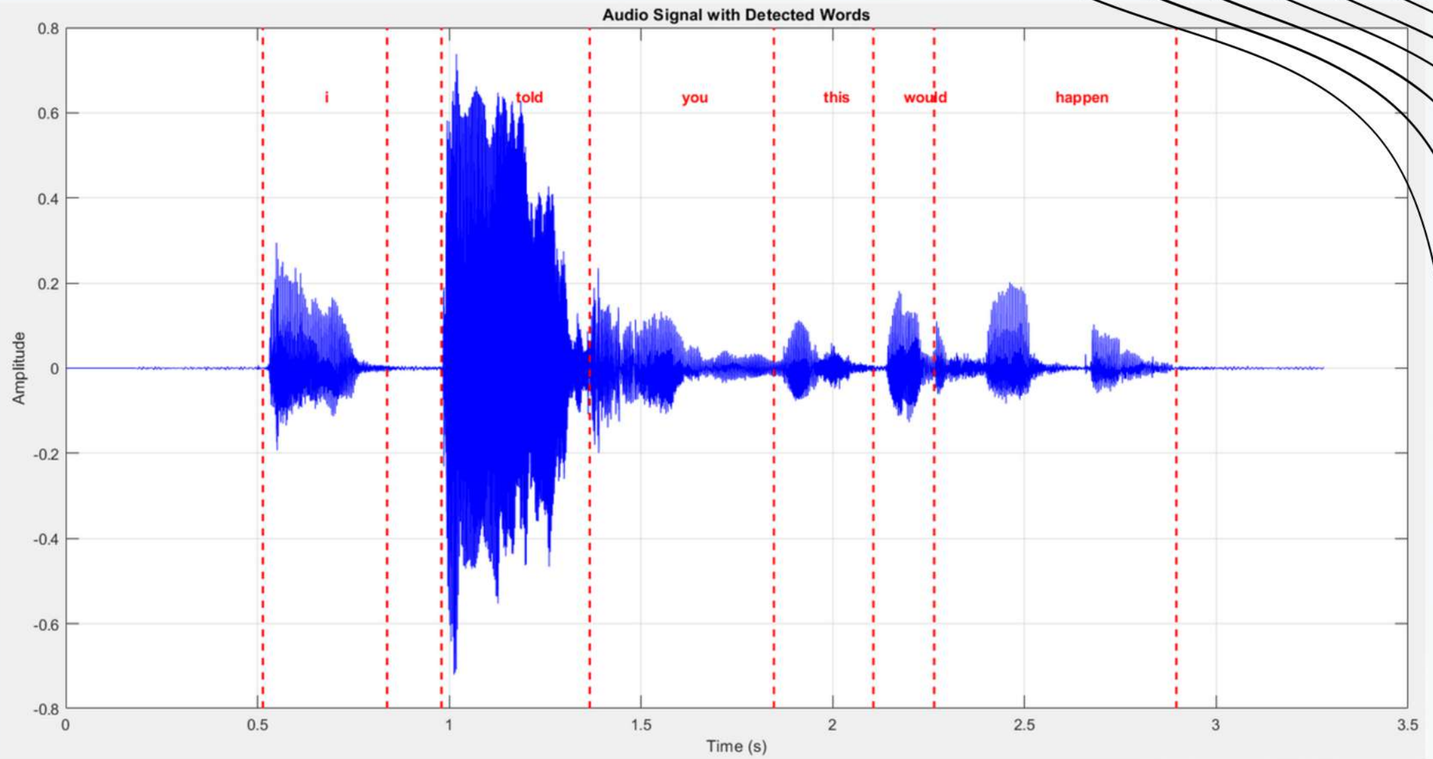



```
RMS values of each segments -
i - 0.0496
cant - 0.1615
believe - 0.0444
we - 0.0377
are - 0.0716
actually - 0.0387
going - 0.0437
to - 0.0124
paris - 0.1419
```

```
Words louder than the threshold -
cant - 0.1615
paris - 0.1419

Words which are actually louder -
cant - 0.1615
paris - 0.1419
```

Sample 5-

1	i	0.513851	0.838009	0
2	told	0.979678	1.366266	1
3	you	1.366266	1.846501	0
4	this	1.846501	2.105827	0
5	would	2.105827	2.264305	0
6	happen	2.264305	2.895813	0



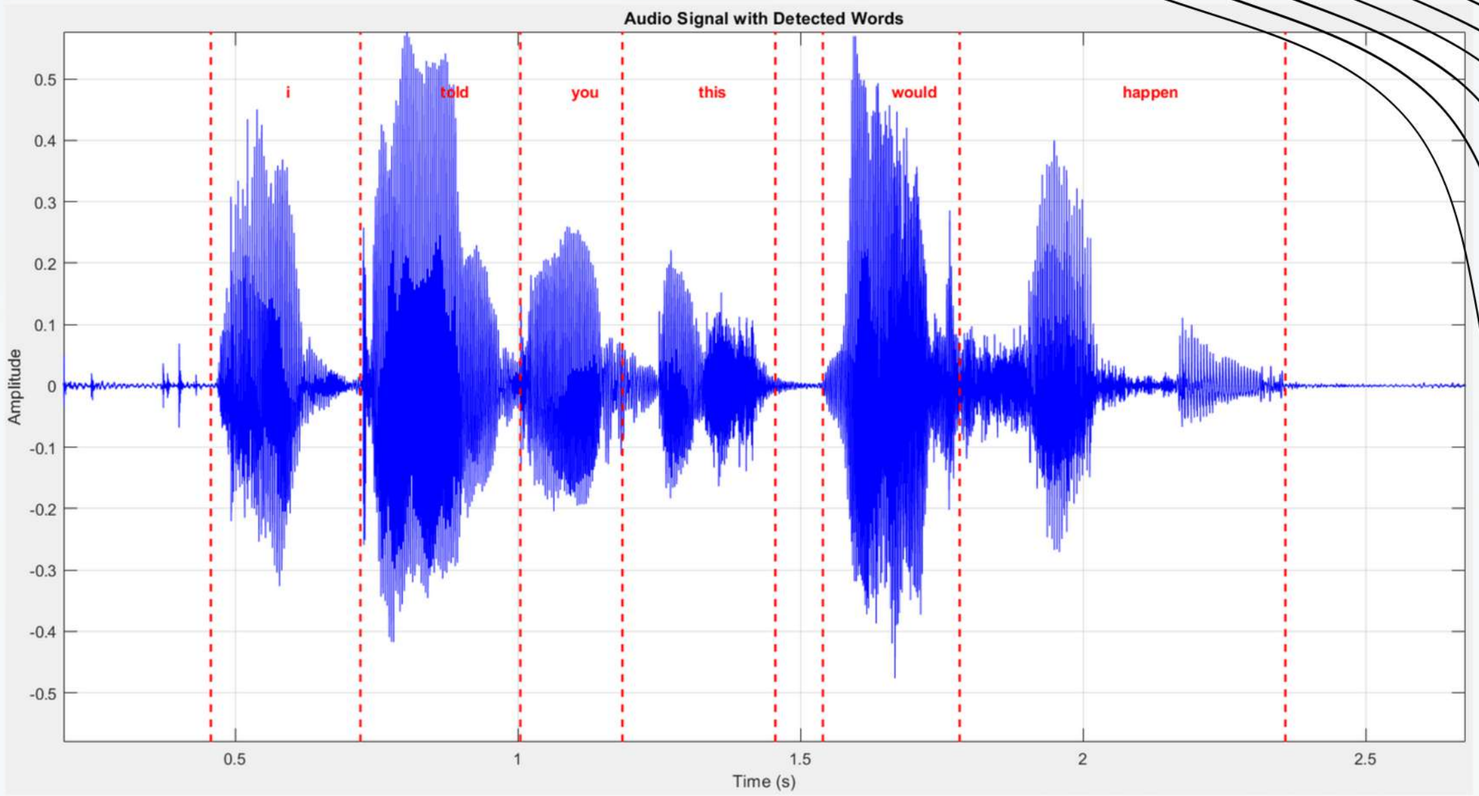
RMS values of each segments -
i - 0.0528
told - 0.2288
you - 0.0463
this - 0.0246
would - 0.0458
happen - 0.0267

Words louder than the threshold -
told - 0.2288

Words which are actually louder -
told - 0.2288

Sample 6-

1	i	0.457101	0.721523	0
2	told	0.721523	1.004392	0
3	you	1.004392	1.184773	0
4	this	1.184773	1.455344	0
5	would	1.539385	1.781259	1
6	happen	1.781259	2.357247	0



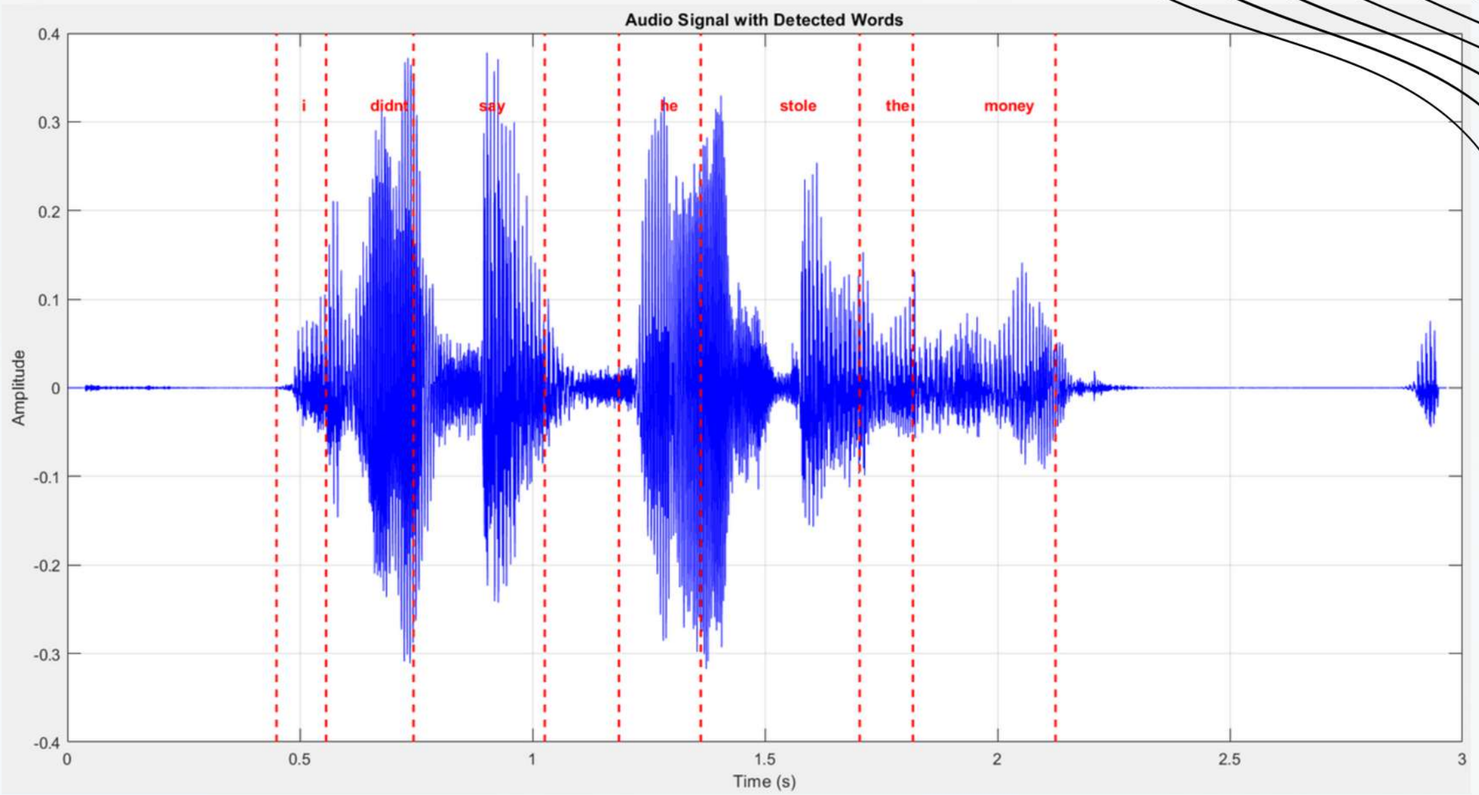

```
RMS values of each segments -
i - 0.0981
told - 0.1668
you - 0.0937
this - 0.0493
would - 0.1515
happen - 0.0541
```

```
Words louder than the threshold -
told - 0.1668
would - 0.1515

Words which are actually louder -
would - 0.1515
```

As we can see, from RMS ‘told’ and ‘would’ both are loud, but it is given that only ‘would’ is loud.
If we see the plot of the signal, told is comparitively louder than would, that means our method was correct and the answer given might be incorrect.

Sample 7-

1	i	0.449975	0.556333	0
2	didnt	0.556333	0.744504	0
3	say	0.744504	1.026761	0
4	he	1.186297	1.362197	1
5	stole	1.362197	1.703768	0
6	the	1.703768	1.818307	0
7	money	1.818307	2.125108	0



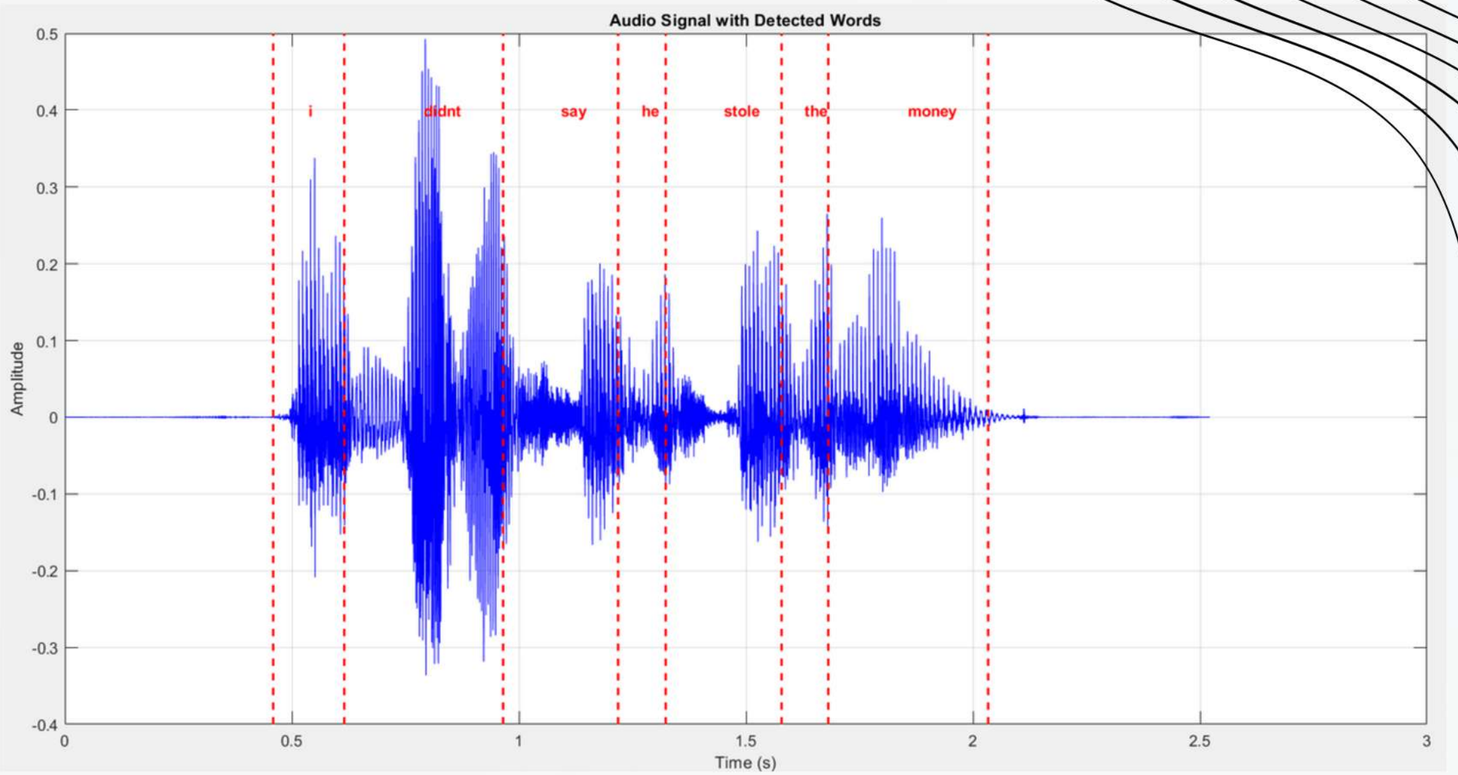

```
RMS values of each segments -
i - 0.0196
didnt - 0.1223
say - 0.0839
he - 0.1214
stole - 0.0893
the - 0.0361
money - 0.0321
```

```
Words louder than the threshold -
didnt - 0.1223
he - 0.1214

Words which are actually louder -
he - 0.1214
```

As we can see, from RMS ‘didnt’ and ‘he’ both are loud, but it is given that only ‘he’ is loud.
If we see the plot of the signal, ‘didnt’ is comparitively louder than would, that means our method was correct and the answer given might be incorrect.

Sample 8-

```
1    i   0.458962    0.615380    1
2    didnt   0.615380    0.965261    1
3    say 0.965261    1.218411    0
4    he  1.218411    1.323375    0
5    stole   1.323375    1.578583    0
6    the 1.578583    1.681489    0
7    money   1.681489    2.033429    0
```



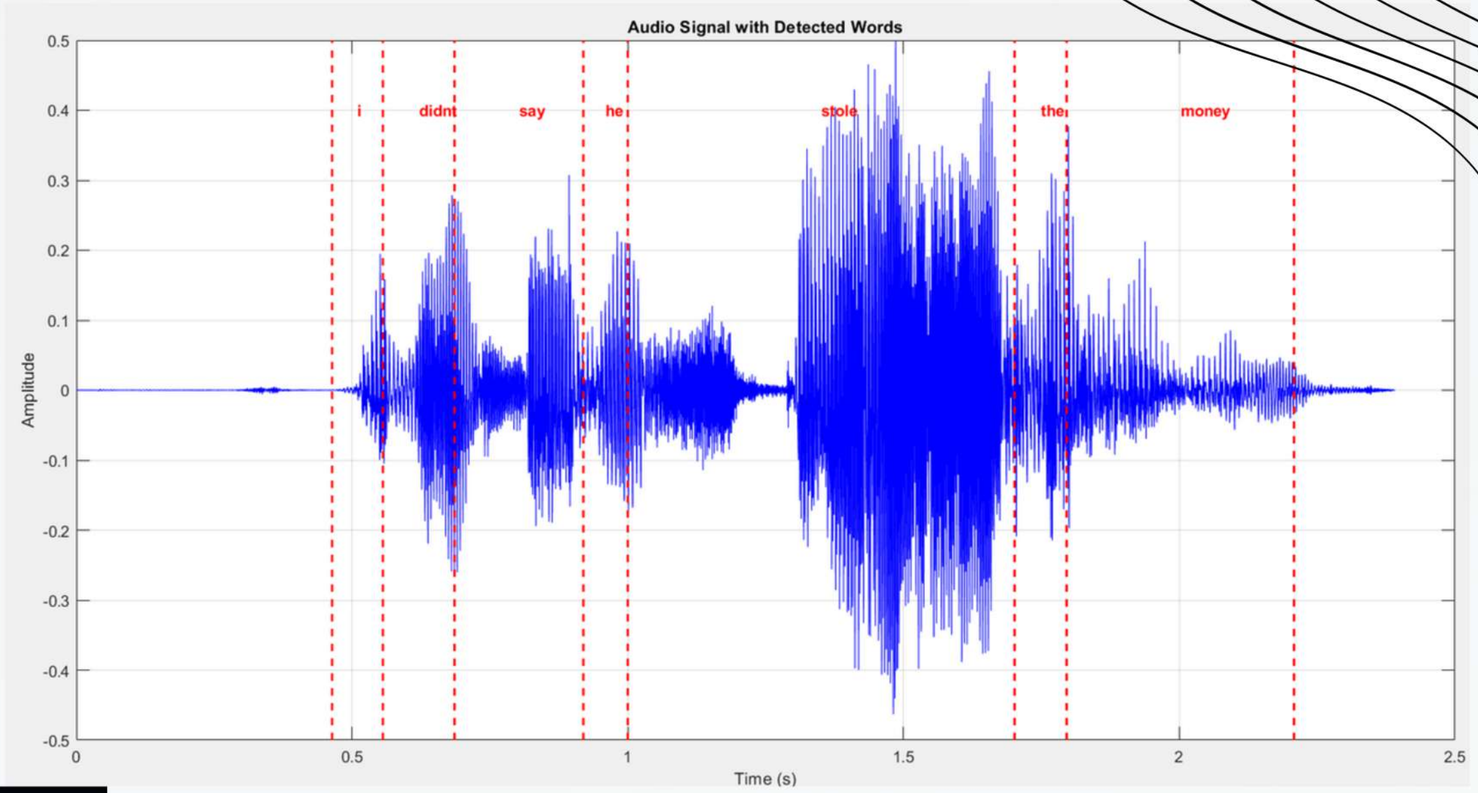

```
RMS values of each segments -
i - 0.0596
didnt - 0.1124
say - 0.0510
he - 0.0404
stole - 0.0478
the - 0.0595
money - 0.0388
```

```
Words louder than the threshold -
didnt - 0.1124

Words which are actually louder -
i - 0.0596
didnt - 0.1124
```

As we can see, from RMS ‘didnt’ is the only loud word but it is given that ‘i’ and ‘didnt’ both are loud. Which is ambigous, because from the graph also ‘i’ doesn’t seem that loud and it also have comparitively less RMS value.

Sample 9-

1	i	0.463641	0.555976	0
2	didnt	0.555976	0.685638	0
3	say	0.685638	0.919423	0
4	he	0.919423	0.999971	0
5	stole	0.999971	1.701326	1
6	the	1.701326	1.795626	0
7	money	1.795626	2.208187	0




```
RMS values of each segments -
i - 0.0294
didnt - 0.0837
say - 0.0755
he - 0.0735
stole - 0.1418
the - 0.0956
money - 0.0387
```

```
Words louder than the threshold -
stole - 0.1418

Words which are actually louder -
stole - 0.1418
```

Task 2 -

For given speech samples, find the louder words without using start and end time information

Approach

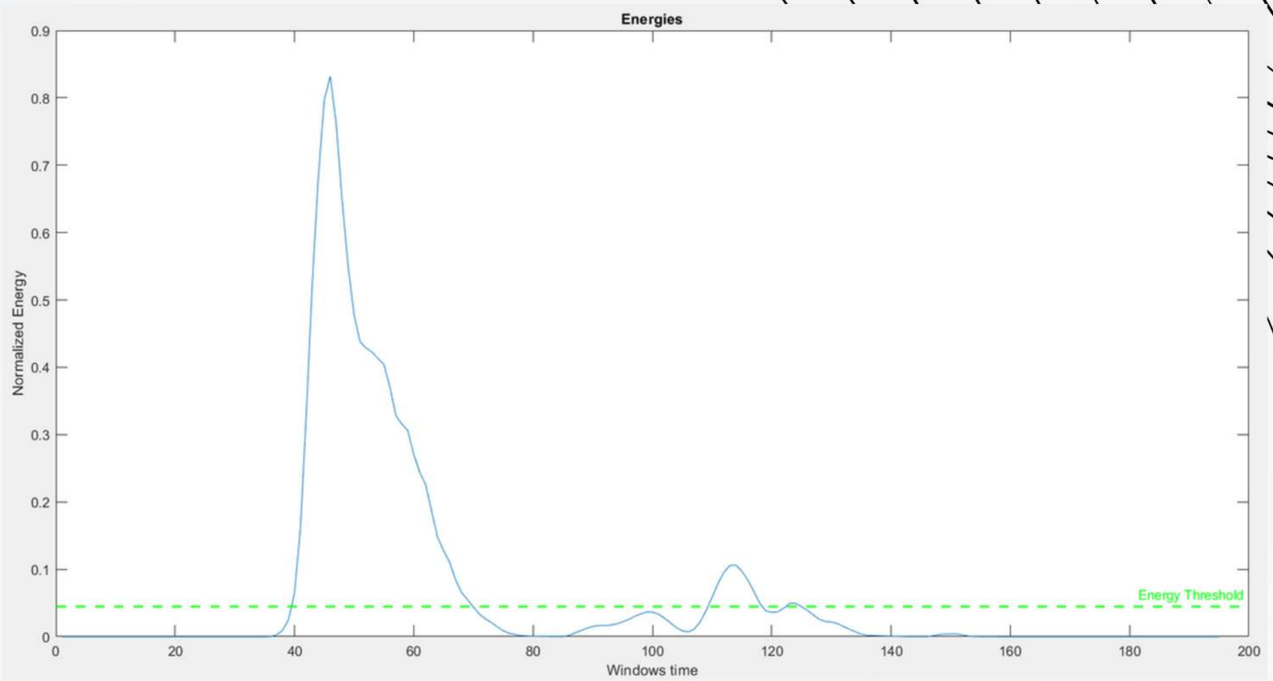
The solution processes an audio file to detect speech segments and analyze their loudness. Using energy-based Voice Activity Detection (VAD), speech regions are identified by detecting high-energy sections of the signal. The Root Mean Square (RMS) energy is calculated for each segment to measure loudness, and segments exceeding a set threshold are classified as "loud." The results, including RMS values and loud segments, are displayed with visualizations for clarity.

Here, we have varried the Threshold for the different audio signals.

Let’s now test our approach in the given speech samples and see what’s the efficiency of our approach

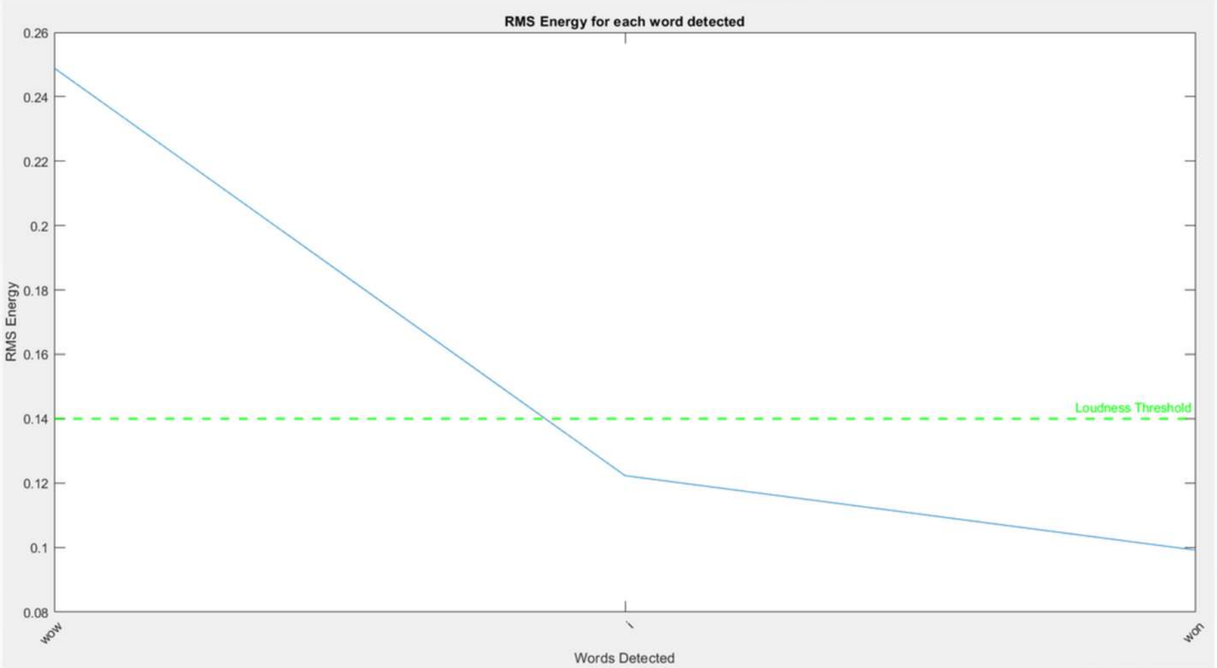
Sample 1-

```
Words detected and their corressponding RMS
wow - 0.248946
i - 0.122290
won - 0.099141

Words louder than the threshold

wow - 0.248946
```



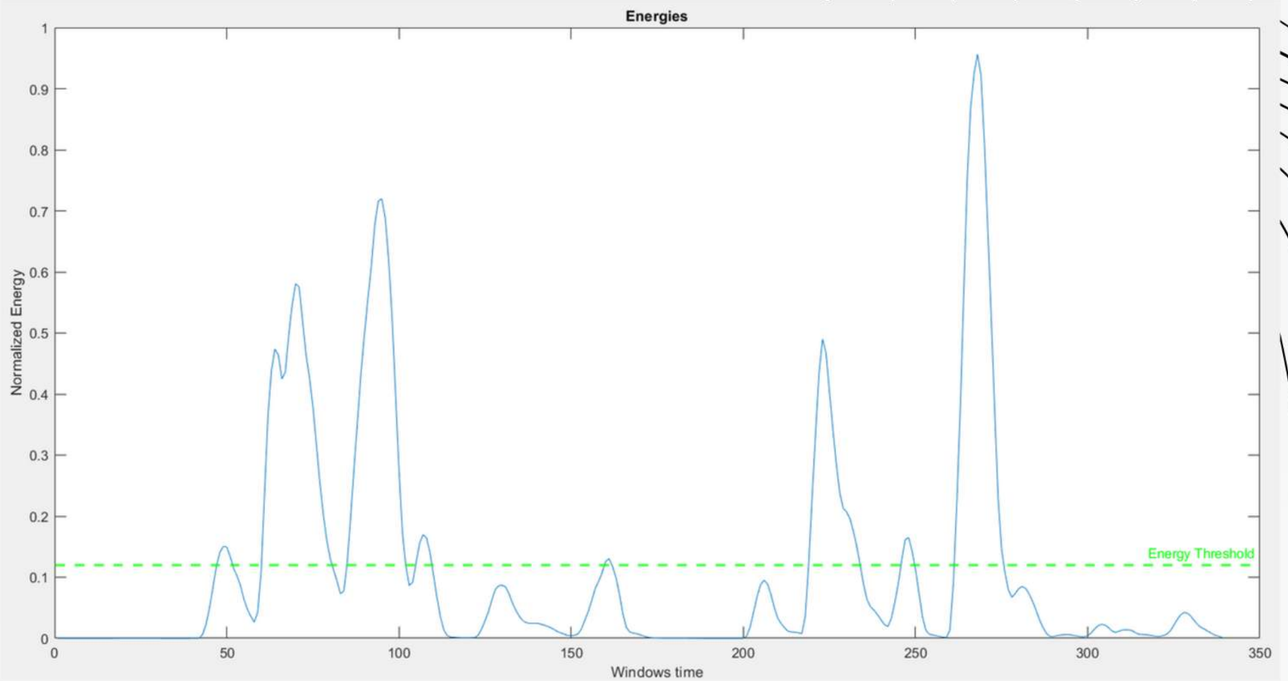
Lower Energy Threshold - 0.04

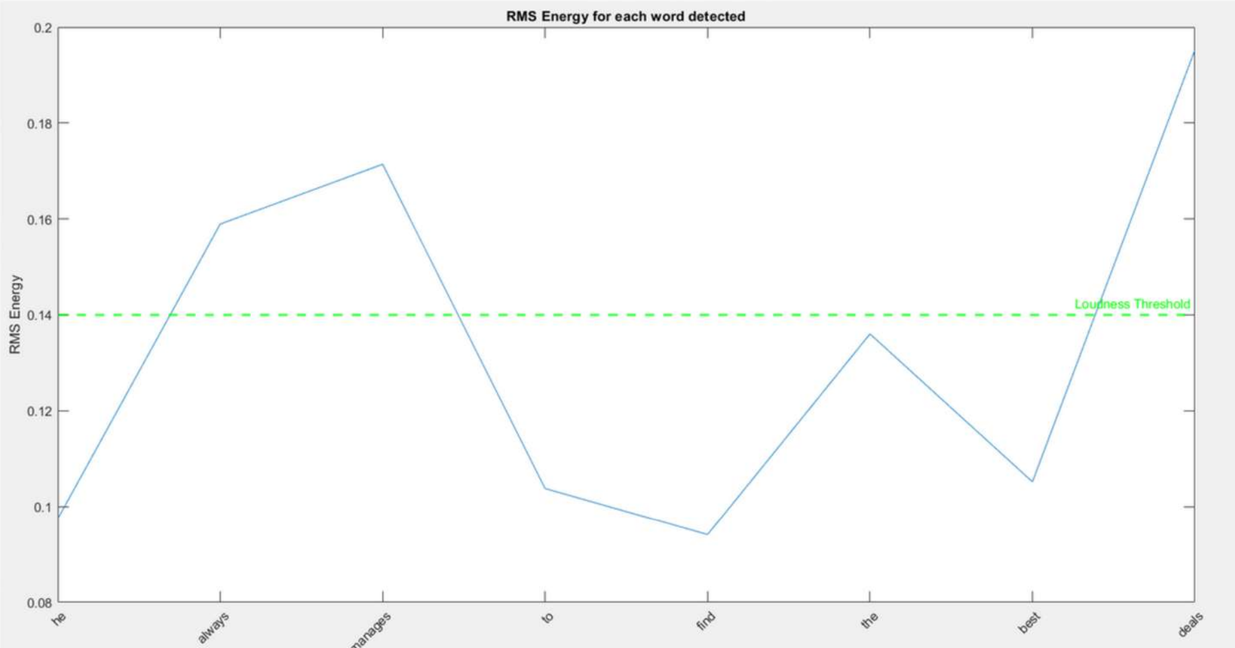
Higher RMS Threshold - 0.14



Sample 2-

```
Words detected and their corressponding RMS
he - 0.097475
always - 0.158947
manages - 0.171402
to - 0.103801
find - 0.094140
the - 0.136029
best - 0.105204
deals - 0.195210

Words louder than the threshold

always - 0.158947
manages - 0.171402
deals - 0.195210
```



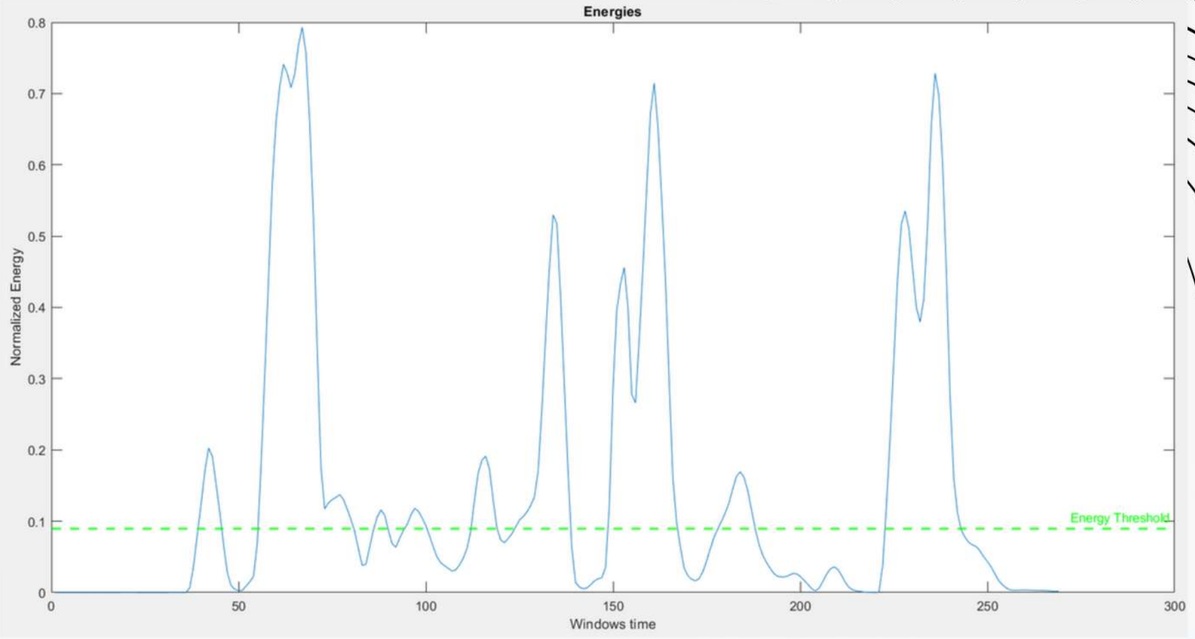


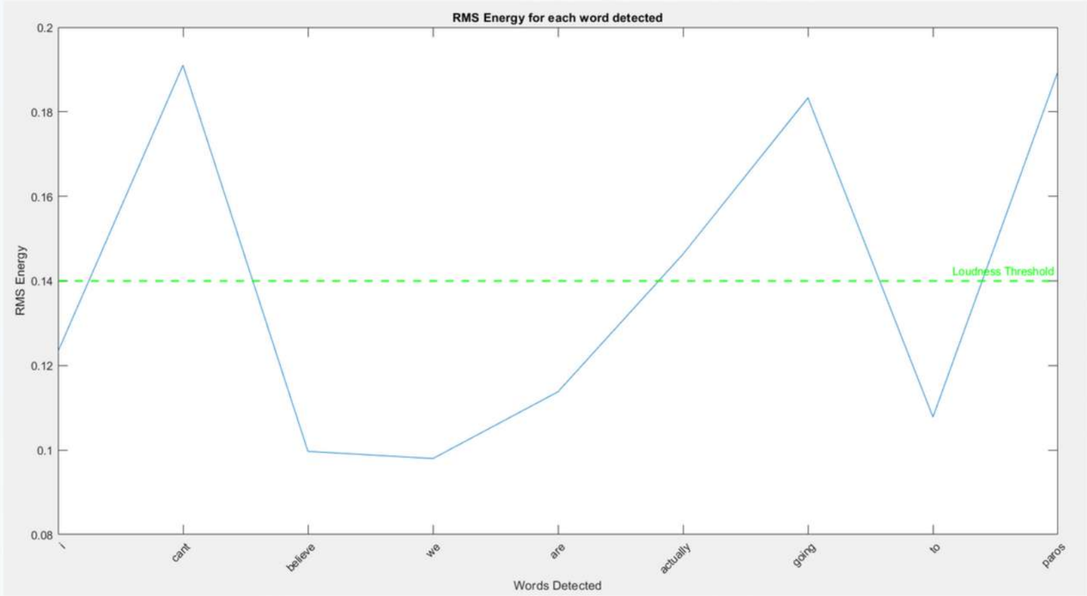
Lower Energy Threshold - 0.12

Higher RMS Threshold - 0.14

Sample 3-

```
Words detected and their corressponding RMS
i - 0.123273
cant - 0.191017
believe - 0.099685
we - 0.097989
are - 0.113780
actually - 0.146329
going - 0.183311
to - 0.107798
paros - 0.189592

Words louder than the threshold

cant - 0.191017
actually - 0.146329
going - 0.183311
paros - 0.189592
```



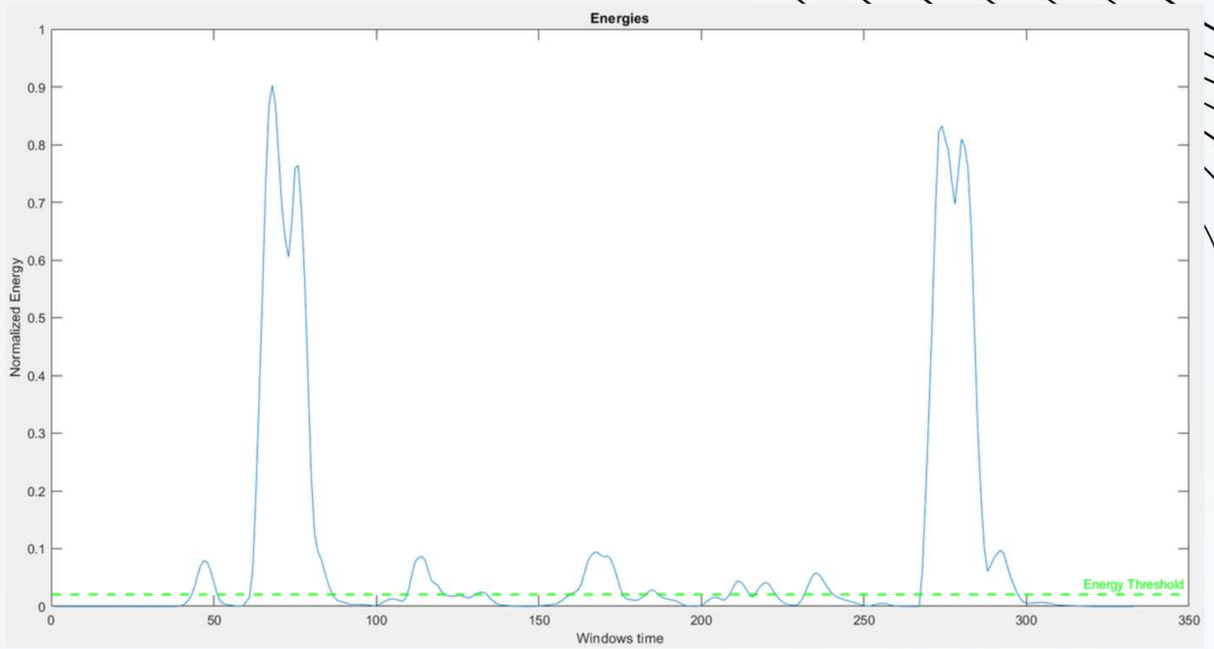


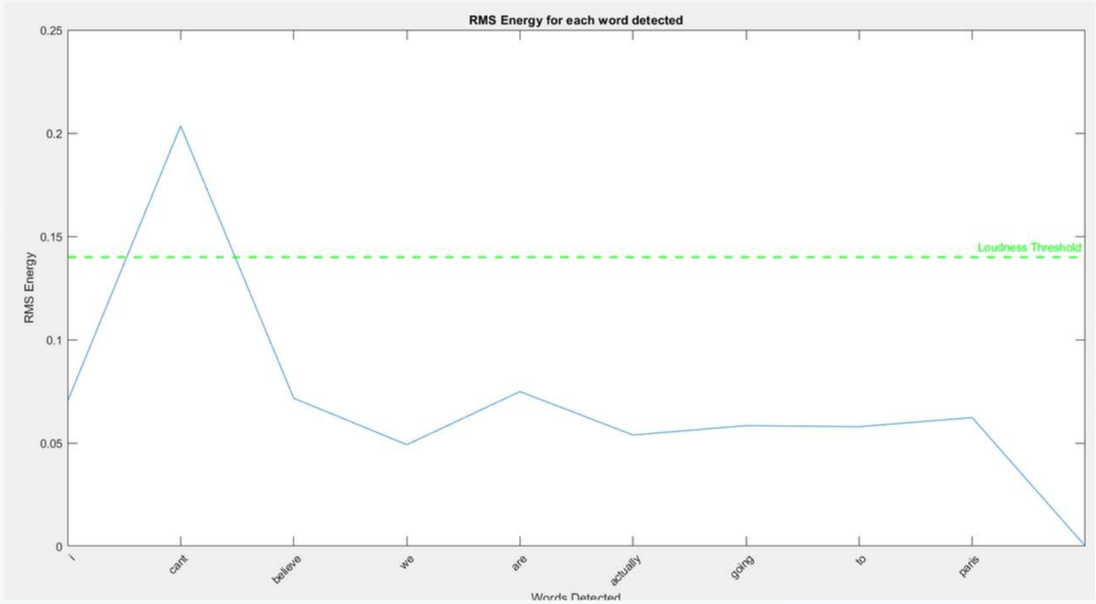
Lower Energy Threshold - 0.09

Higher RMS Threshold - 0.14

Sample 4-

```
Words detected and their corressponding RMS
i - 0.070149
cant - 0.203496
believe - 0.071651
we - 0.049115
are - 0.074855
actually - 0.053837
going - 0.058351
to - 0.057856
paris - 0.062248

Words louder than the threshold

cant - 0.203496
```



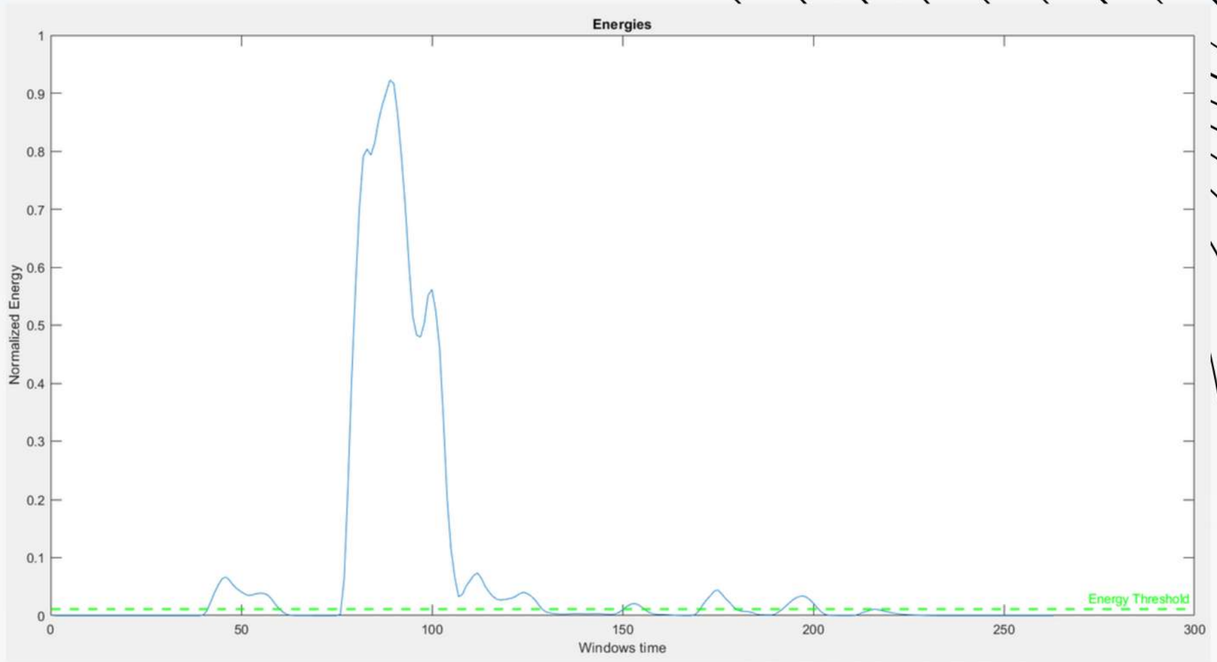


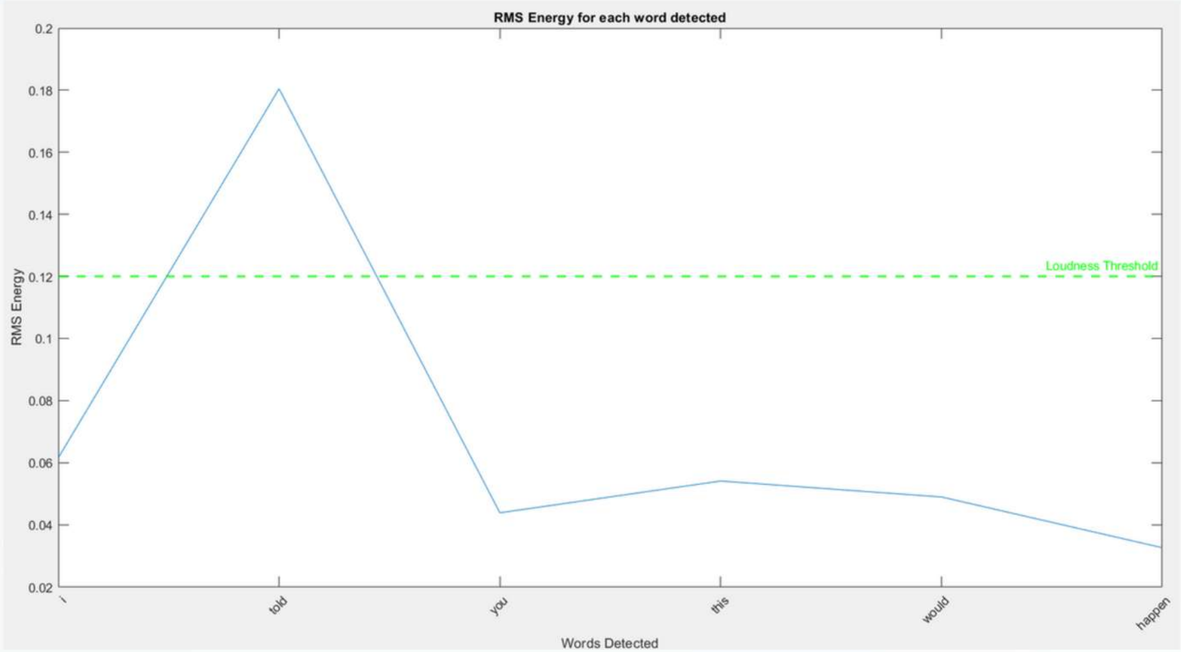
Lower Energy Threshold - 0.02

Higher RMS Threshold - 0.14

Sample 5-

```
Words detected and their corressponding RMS
i - 0.061533
told - 0.180452
you - 0.043832
this - 0.054065
would - 0.048930
happen - 0.032578

Words louder than the threshold

told - 0.180452
```



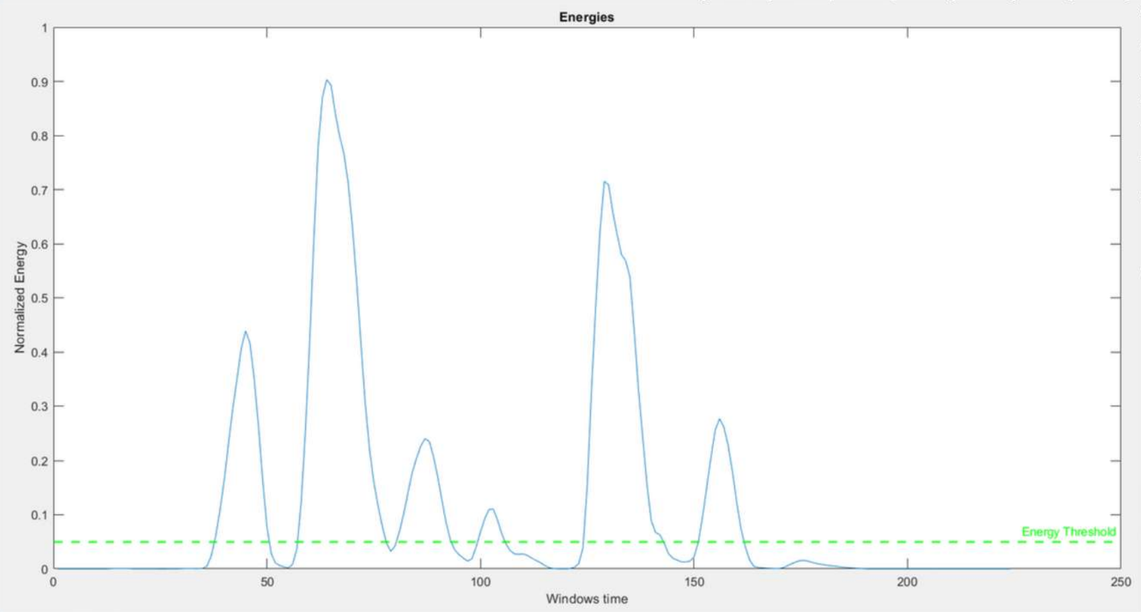


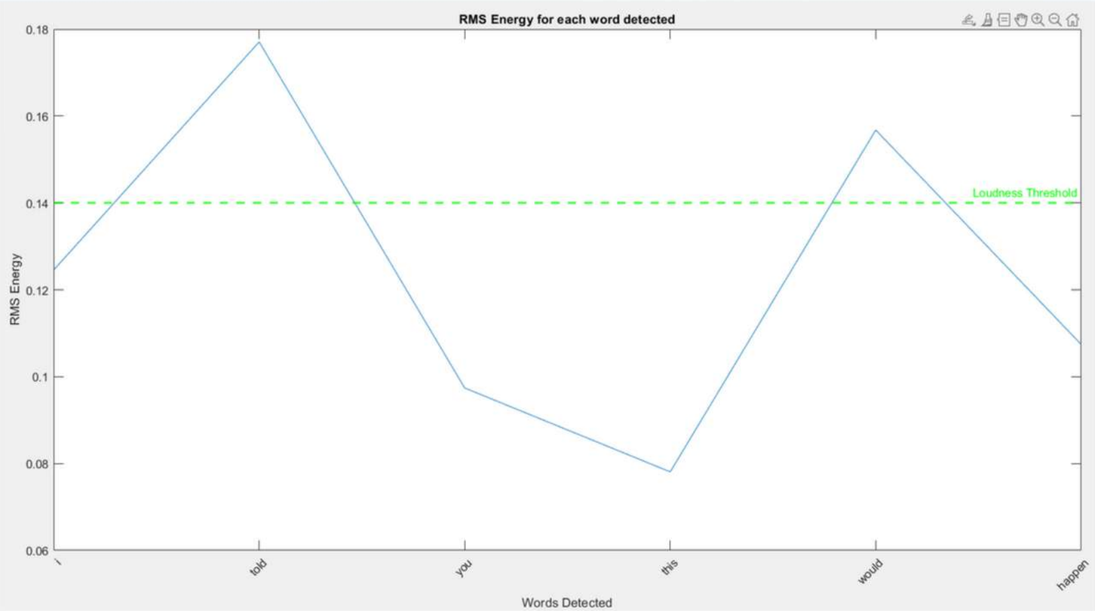
Lower Energy Threshold - 0.011

Higher RMS Threshold - 0.14

Sample 6-

```
Words detected and their corressponding RMS
i - 0.124537
told - 0.177065
you - 0.097399
this - 0.078080
would - 0.156778
happen - 0.107405

Words louder than the threshold

told - 0.177065
would - 0.156778
```



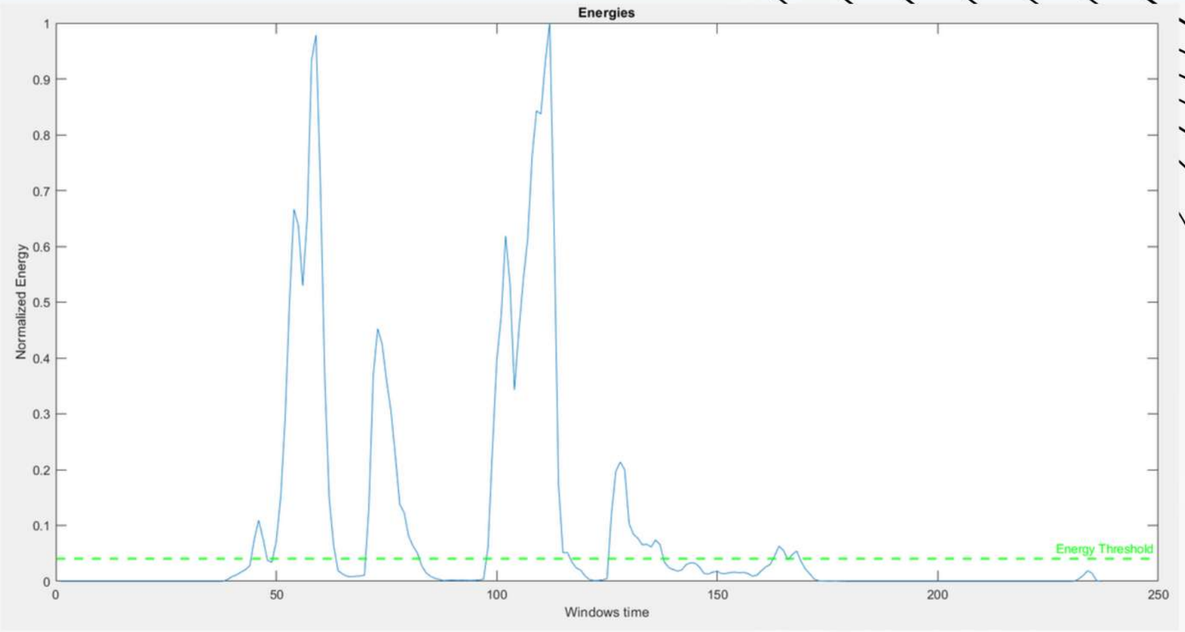


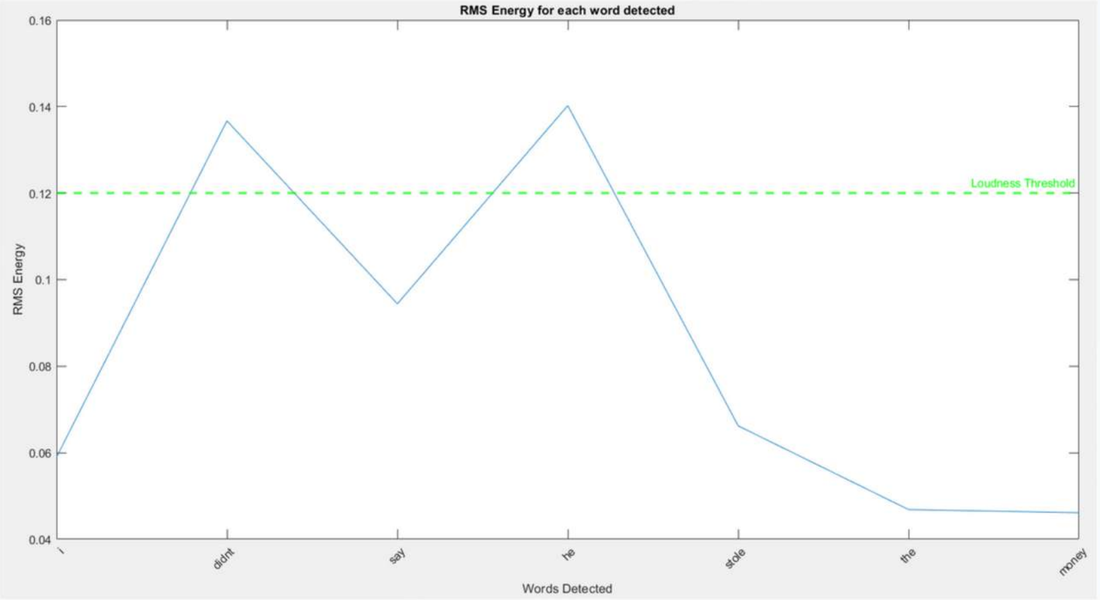
Lower Energy Threshold - 0.05

Higher RMS Threshold - 0.14

Sample 7-

```
Words detected and their corressponding RMS
i - 0.058994
didnt - 0.136707
say - 0.094421
he - 0.140228
stole - 0.066213
the - 0.046864
money - 0.046150

Words louder than the threshold

didnt - 0.136707
he - 0.140228
```



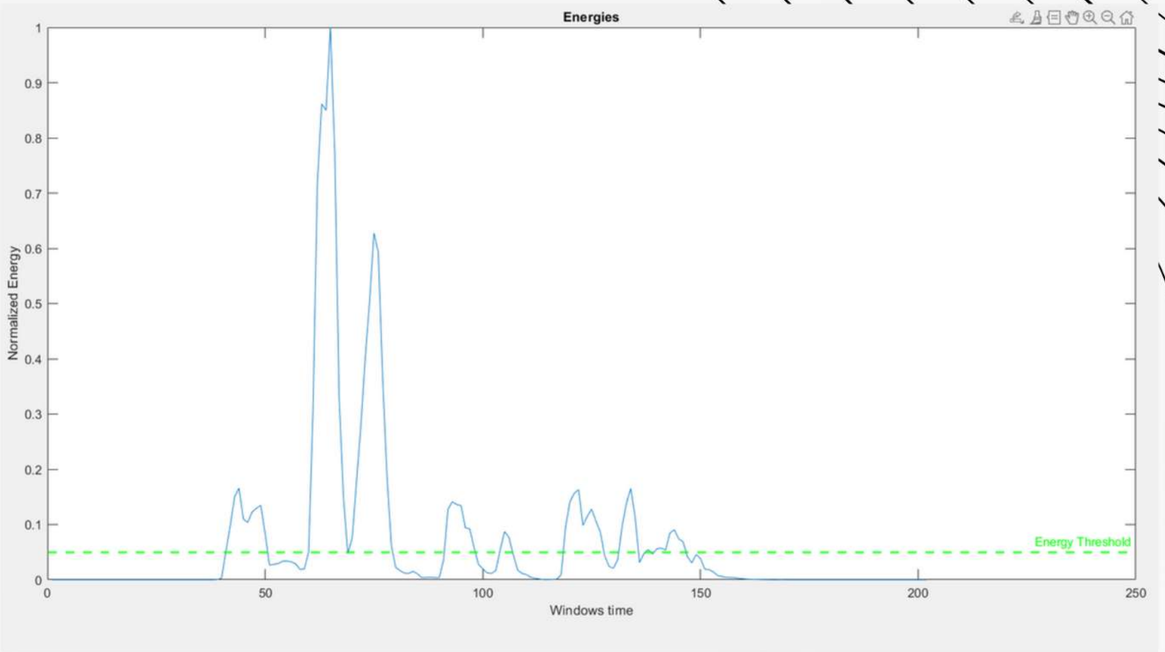


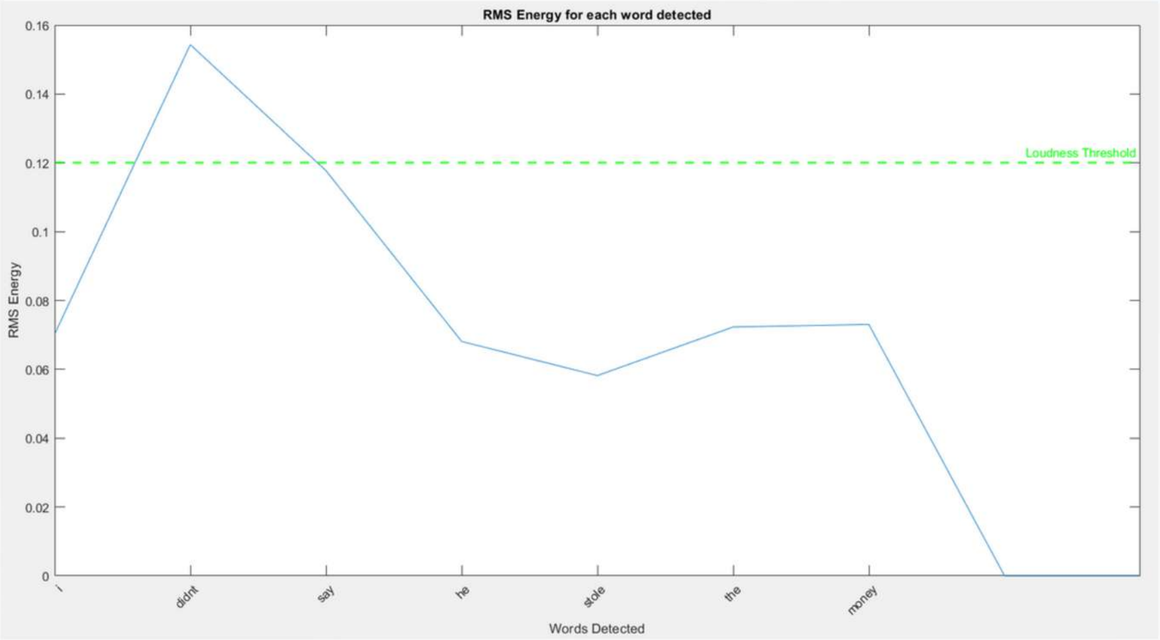
Lower Energy Threshold - 0.04

Higher RMS Threshold - 0.12

Sample 8-

```
Words detected and their corressponding RMS
i - 0.070197
didnt - 0.154348
say - 0.117714
he - 0.068090
stole - 0.058191
the - 0.072305
money - 0.073046

Words louder than the threshold

didnt - 0.154348
```



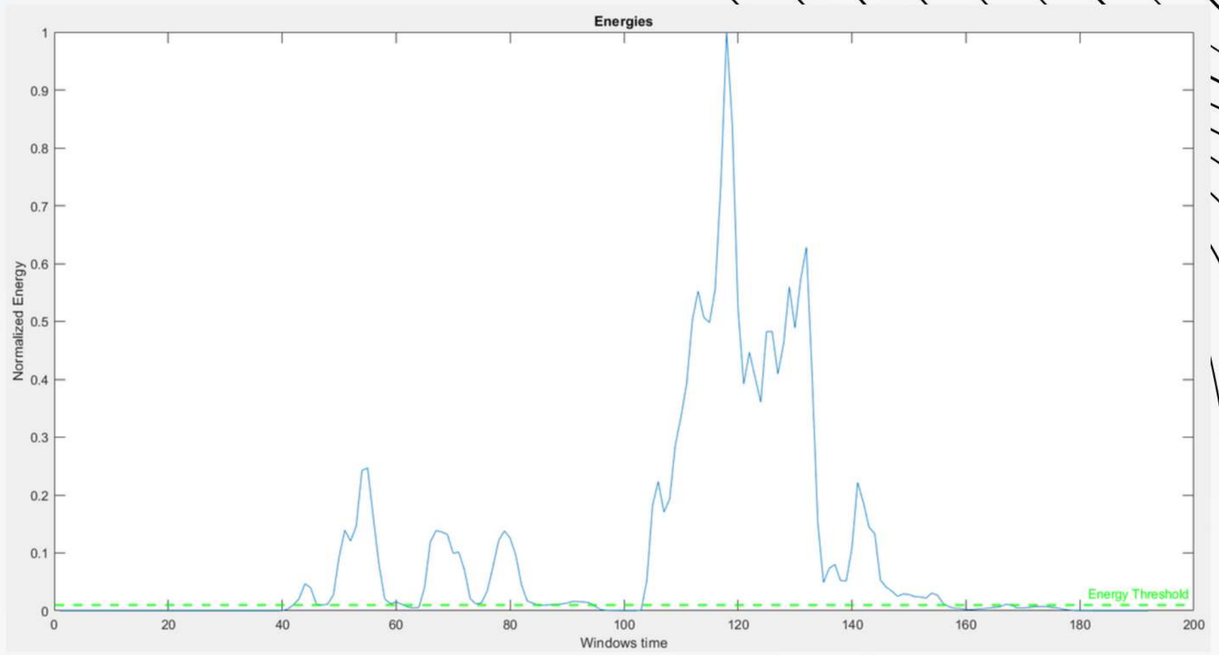


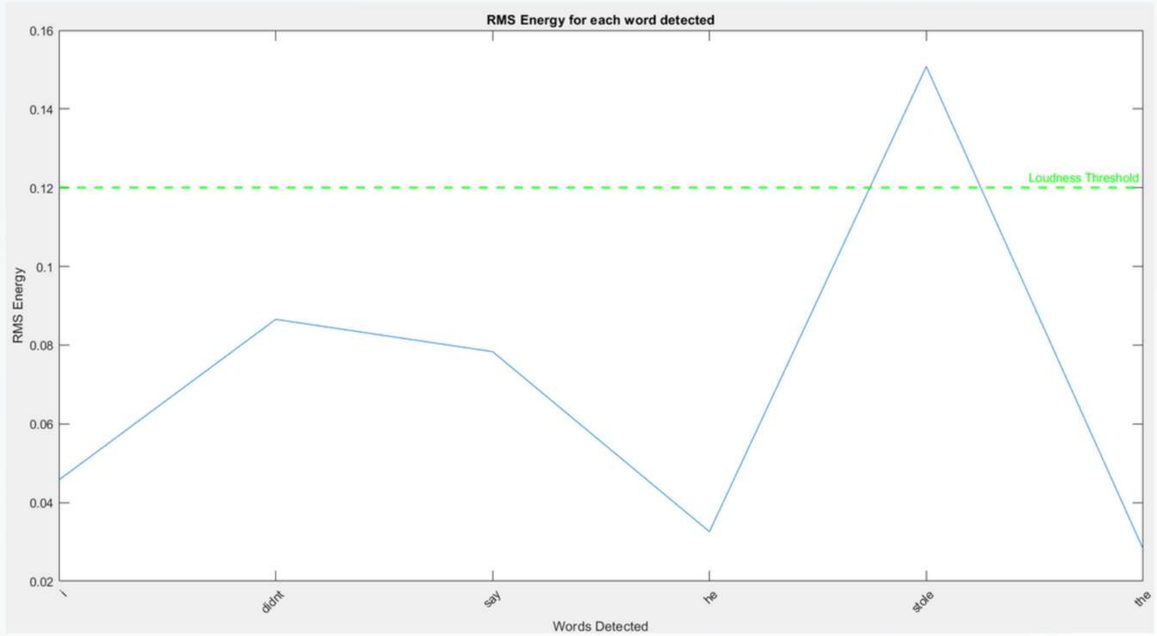
Lower Energy Threshold - 0.05

Higher RMS Threshold - 0.12

Sample 9-

```
Words detected and their corressponding RMS
i - 0.045733
didnt - 0.086548
say - 0.078348
he - 0.032592
stole - 0.150802
the - 0.028156

Words louder than the threshold

stole - 0.150802
```





Lower Energy Threshold - 0.01

Higher RMS Threshold - 0.12

Conclusion -

- In the task one we are getting some results as wrong if we compare it to what was given to us, but logically we were correct (we can see the plots and tell that we were correct)
- In the task 2 , we are able to detect almost every words, but the RMS energy that we were calculating / getting are different from what was there in task 1
    - **Limitations in Task 2**
        - We are required to change threshold for every speech
        - If there is some silent period in a single word, our algorithm will detect it as two words.
        - If some word's amplitude it very low, our algorithm will detect it as a silent period and won' t detect any word.