Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

Division of Electronics and Communication Engineering

III IA EVALUATION REPORT

for

BLENDED LEARNING PROJECT BASED LEARNING

Implementation of Huffman Method AHAP for ECG Signal in MATLAB

A report submitted by

Name of the Student	*Godwin Sagai V, Aldrin Infant Raj F*
Register Number	*URK22EC1014, URK22EC1069*
Subject Name	*Digital Communication*
Subject Code	*22EC2016*
Date of Report submission	*26/10/24*

Total marks: _____/ 40 Marks

Signature of Faculty with date:

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

TABLE OF CONTENTS

CHAPTER – 1
INTRODUCTION

The need for reliable, real-time monitoring of cardiovascular health has increased with the growth of wearable health devices and telemedicine. Electrocardiograms (ECGs) are crucial for diagnosing heart conditions, generating vast amounts of data during continuous monitoring.

However, transmitting and storing ECG data in high volume can be costly and inefficient. Therefore, efficient data compression methods are essential to reduce data size while preserving diagnostic information.

Huffman coding, a lossless data compression technique, is a widely applied method that reduces data size efficiently without compromising data integrity. By assigning variable-length codes to different data symbols based on their frequency of occurrence, Huffman coding minimizes the overall storage requirements of a dataset. This is achieved by representing frequently occurring symbols with shorter codes and less common symbols with longer ones, resulting in a compressed data stream. This method's efficiency and simplicity make it a popular choice across a range of fields, from text compression to multimedia and scientific data storage.

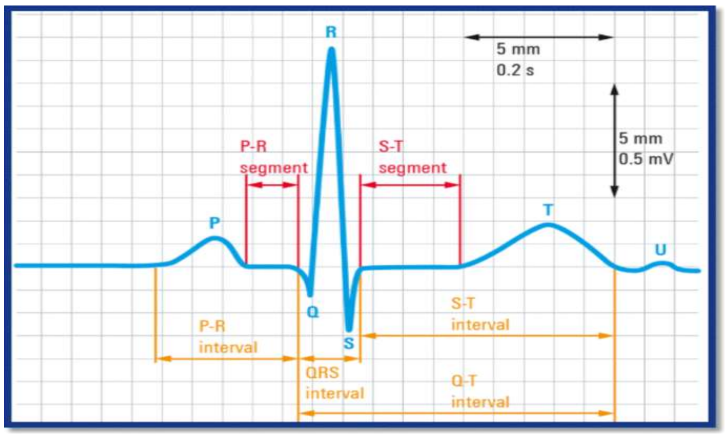
In this project, Huffman coding is applied to electrocardiogram (ECG) signals, a crucial data source in healthcare monitoring, diagnostics, and research. ECG data captures the electrical activity of the heart over time, often requiring high-frequency sampling and generating large amounts of data. Compressing this data efficiently is critical in both clinical and remote healthcare settings, as it enables better storage management and faster transmission, supporting real-time analysis and continuous monitoring in wearable devices. By applying Huffman coding to ECG signals, we aim to reduce the data volume significantly while maintaining signal accuracy and integrity, ensuring that the compression process does not impact the quality of information available for medical diagnosis.

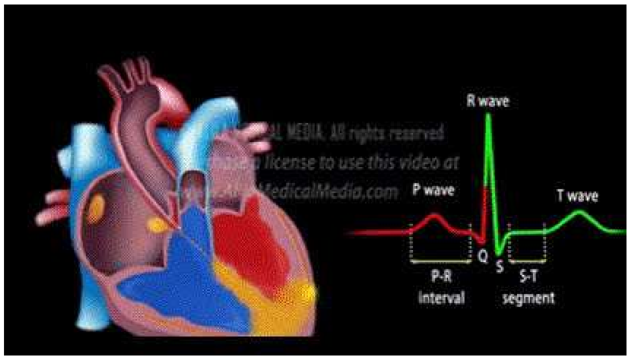
This report explores the effectiveness of Huffman coding in compressing ECG data, providing a detailed analysis of the compression ratio, storage savings, and computational efficiency achieved. In evaluating its feasibility in healthcare applications and more efficient digital healthcare solutions

What is ECG?

An ECG (electrocardiogram) is a test that records the electrical activity of the heart over a period of time. By placing electrodes on the skin, an ECG measures the electrical signals that cause the heart muscles to contract and relax, producing a trace that shows heart rate and rhythm.

ECG Signal:


R
5 mm
0.2 s
5 mm
0.5 mV
P-R
segment
S-T
segment
P
T
U
Q
S
P-R
interval
S-T
interval
QRS
interval
Q-T
interval


R wave
P wave
T wave
Q
S
P-R
interval
S-T
segment

CHAPTER - 2
PROBLEM STATEMENT

The increasing use of ECG (electrocardiogram) data in health monitoring and diagnostics has led to a significant growth in data storage and transmission requirements. Given that ECG signals are often recorded over extended periods, efficient data compression is crucial to reduce storage costs and enhance transmission speed without compromising signal quality.

CHAPTER - 3
MARKET SURVEY

Wavelet transform with Huffman coding, modified run-length encoding (RLE), and distributed source coding (DSC) are advanced techniques used for compressing ECG signals. Each method has unique characteristics and applications in current medical devices.

Wavelet Transform with Huffman Coding

This technique involves preprocessing the ECG signal by segmenting it into beats and applying the Discrete Wavelet Transform (DWT). The resulting wavelet coefficients are then encoded using Huffman coding schemes, such as optimal selective, adaptive, and modified adaptive Huffman coding. The primary goal is to assign shorter codewords to more frequent symbols, enhancing compression efficiency. This approach has shown promising results in terms of compression ratios and time efficiency, making it suitable for real-time applications in Holter monitoring systems and wireless ECG devices 14.

Modified Run-Length Encoding (RLE)

Modified RLE is a compression technique that focuses on reducing the size of digitized ECG signals while maintaining their integrity. It works by encoding sequences of identical data points as a single value and count, which is particularly effective for ECG signals that often contain long runs of similar values. Recent studies have demonstrated the successful implementation of modified RLE in various ECG compression frameworks, highlighting its efficiency in long-term monitoring applications like Holter systems 23.

Distributed Source Coding (DSC)

DSC exploits redundancy in ECG signals for low-complexity compression, allowing for efficient encoding without extensive computational resources. This method is particularly useful in real-time monitoring scenarios where quick data processing is essential. Current research indicates that DSC can significantly enhance the performance of ECG data handling by reducing the overall data size while ensuring the quality of the transmitted information 3.

These techniques are actively used in modern medical devices and research applications:

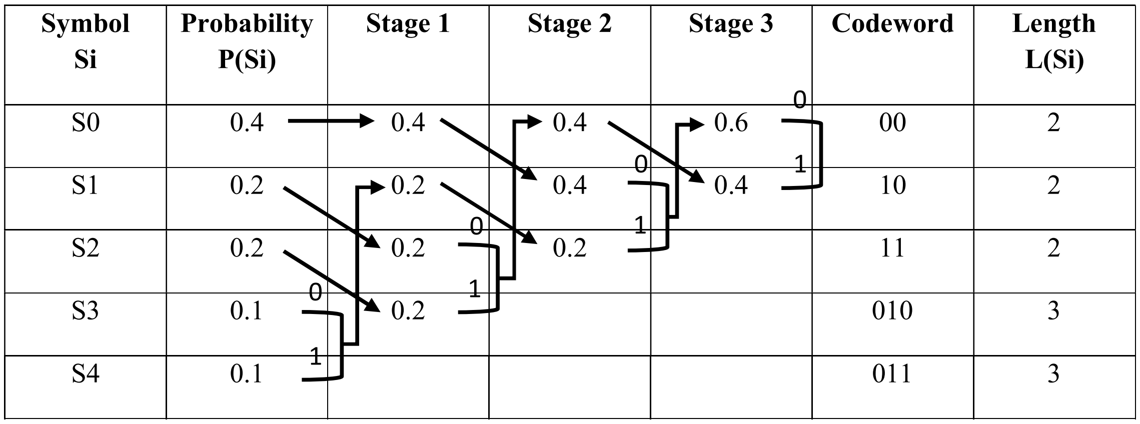
Wavelet Transform with Huffman Coding: Integrated into wireless ECG sensors and Holter monitors to optimize data transmission and storage.

Modified RLE: Applied in Holter monitoring systems to compress long-term ECG recordings effectively.

DSC: Utilized in real-time monitoring systems to manage bandwidth efficiently while maintaining signal integrity.

Overall, these methods are not only theoretical but are also being implemented in various medical technologies aimed at improving patient care through efficient ECG signal processing.

CHAPTER - 4
MATHEMATICAL MODEL



Symbol Si	Probability P(Si)	Stage 1	Stage 2	Stage 3	Codeword	Length L(Si)
S0	0.4	0.4	0.4	0.6	00	2
S1	0.2	0.2	0.4	0.4	10	2
S2	0.2	0.2	0.2		11	2
S3	0.1	0.2			010	3
S4	0.1				011	3

Average Codeword Length:

Formula:

$$\overline{\mathbf{L_{avg}}} = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{P(Si) \times L(Si)}$$

Substituting values:

$$\overline{\mathrm{L_{avg}}} = (0.4\times2) +(0.2\times2) +(0.2\times2) +(0.1\times3) +(0.1\times3)$$

$$\overline{\mathrm{L_{avg}}} = 0.8+0.4+0.4+0.3+0.3$$

$$\overline{\mathbf{L_{avg}}} = \mathbf{2.2\ bits\ /\ symbol}$$

Entropy:

Formula:

$$\mathbf{H} = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{P(Si)\log_2\left(\frac{1}{P(Si)}\right)}$$

Substituting values:

$$\mathrm{H} = 0.4\log_2\left(\frac{1}{0.4}\right) + 0.2\log_2\left(\frac{1}{0.2}\right) + 0.2\log_2\left(\frac{1}{0.2}\right) + 0.1\log_2\left(\frac{1}{0.1}\right) + 0.1\log_2\left(\frac{1}{0.1}\right)$$

H = 2.1219 bits / Symbol

Coding Efficiency:

$$\boldsymbol{\eta} = \left(\frac{\mathbf{H}}{\overline{\mathbf{L_{avg}}}}\right) \times \mathbf{100}$$

Substituting values:

$$\eta = \frac{2.1219}{2.2} \times 100$$

$$\boldsymbol{\eta} = \mathbf{96.45\%}$$

Variance:

$$\boldsymbol{\sigma^2} = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{P(Si)}\left(\mathbf{L(Si)} - \overline{\mathbf{L_{avg}}}\,\right)^{\mathbf{2}}$$

Substituting values:

$$\sigma^2 = 0.4(2-2.2)^2 + 0.2(2-2.2)^2 + 0.2(2-2.2)^2 + 0.1(3-2.2)^2 + \ 0.1(3-2.2)^2$$

$$\boldsymbol{\sigma^2} = \mathbf{0.16}$$

CHAPTER - 5
HUFFMAN CODING AHAP ALGORITHM

1) Loading the ECG Signal Data:

We load the ECG signal data from a .mat file, containing a sampled signal with a frequency of 360 Hz. The time axis is calculated based on the length of the data to facilitate plotting.

2) Symbol and Probability Extraction

Each unique amplitude level in the signal (represented as unique symbols) is identified, and a probability distribution is computed for these symbols by dividing each symbol's occurrence by the total number of samples.

3) Building the Huffman Tree

Using the computed probabilities, a Huffman tree is constructed. The process includes:

- Sorting symbols by probability.
- Merging the two least probable nodes iteratively until a single root node remains.
- Assigning binary codes recursively to each symbol, creating a mapping from symbols to unique binary codes.

4) Data Compression

The data is compressed by substituting each symbol in the original signal with its corresponding Huffman code, reducing the storage space required.

5) Entropy and Redundancy Calculation

We calculate the entropy of the signal, the average code length, redundancy, and the compression efficiency. These metrics provide insights into the compression's effectiveness and the information density in the signal.

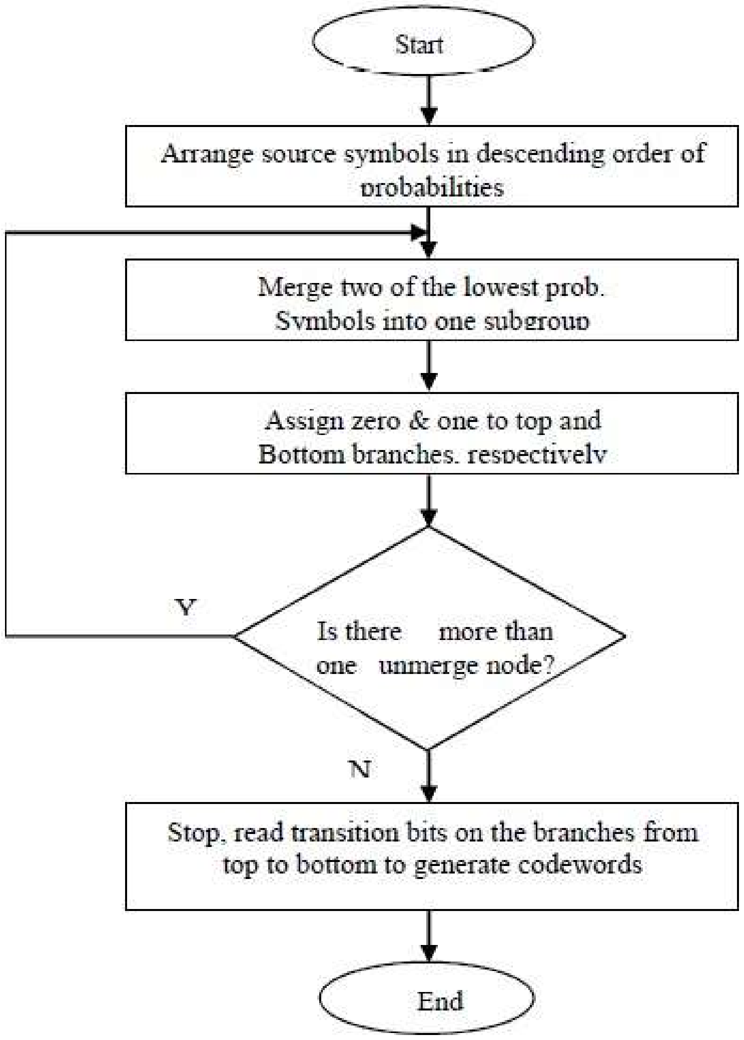
6) Decoding and Verification

The compressed data is decoded by using the Huffman codes in reverse. The decoded data is then compared with the original signal to verify the accuracy of the compression and decompression process.

7) Visualization

The original and decoded signals are plotted to visually confirm that the decoded signal closely matches the original. Additionally, a bar graph displays the probability distribution of symbols in the original signal, demonstrating which symbols are more frequent.

Flowchart:



Start
Arrange source symbols in descending order of probabilities
Merge two of the lowest prob. Symbols into one subgroup
Assign zero & one to top and Bottom branches, respectively
Is there more than one unmerge node?
Y
N
Stop, read transition bits on the branches from top to bottom to generate codewords
End

CHAPTER - 6
CODE

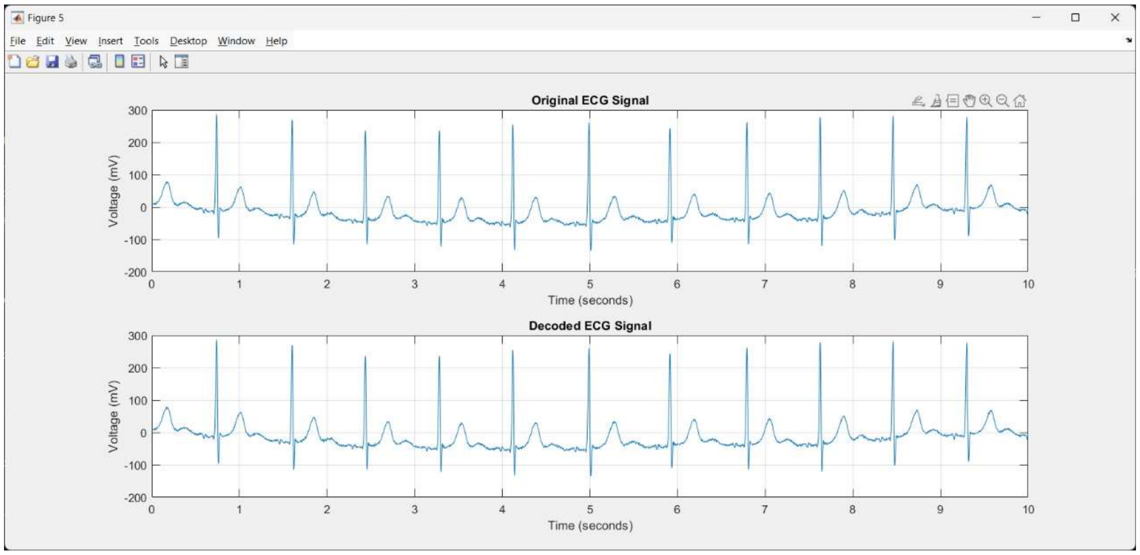
```
% Load the ECG data
load('data.mat');
sampling_frequency = 360;
time_axis = (0:length(data)-1) / sampling_frequency;

% Plot the original ECG signal
figure;
subplot(2,1,1);
plot(time_axis, data);
xlabel('Time (seconds)');
ylabel('Voltage (mV)');
title('Original ECG Signal');
grid on;

% Unique symbols and their probabilities
symbols = unique(data);
symbols = reshape(symbols, [], 1);
bin_edges = [symbols; max(symbols) + 1];
probabilities = histcounts(data, bin_edges) / numel(data);

% Create nodes for each symbol
nodes = cell(1, numel(symbols));
for i = 1:numel(symbols)
    nodes{i} = struct('symbol', symbols(i), 'probability', probabilities(i));
end

% Build the Huffman Tree
while numel(nodes) > 1
    [~, sort_idx] = sort(cellfun(@(x) x.probability, nodes));
    node1 = nodes{sort_idx(1)};
    node2 = nodes{sort_idx(2)};
    new_node = struct('symbol', [], 'probability', node1.probability + node2.probability, 'left', node1, 'right',
node2);
    nodes = [nodes(sort_idx(3:end)), new_node];
end

% Generate Huffman codes
function codes = huffman_codes(node)
    codes = containers.Map('KeyType', 'double', 'ValueType', 'char');
    traverse_tree(node, '', codes);
end

function traverse_tree(node, code, codes)
    if isempty(node.symbol)
        traverse_tree(node.left, [code '0'], codes);
        traverse_tree(node.right, [code '1'], codes);
    else
        codes(node.symbol) = code;
    end
```

```
end

codes = huffman_codes(nodes{1});

% Compress the data
compressed_data = arrayfun(@(x) codes(x), data, 'UniformOutput', false);

% Display Huffman codes for each symbol
[sorted_probabilities, sorted_indices] = sort(probabilities, 'descend');
sorted_symbols = symbols(sorted_indices);
for i = 1:numel(sorted_symbols)
  fprintf('Symbol: %d,\t Probability: %.5f,\t Code: %s\n', sorted_symbols(i), sorted_probabilities(i),
codes(sorted_symbols(i)));
end

% Entropy calculation
entropy = -sum(probabilities(probabilities > 0) .* log2(probabilities(probabilities > 0)));

% Average code length calculation
average_code_length = 0;
for i = 1:numel(symbols)
  code_length = length(codes(symbols(i)));
  average_code_length = average_code_length + probabilities(i) * code_length;
end

% Variance calculation
variance = 0;
for i = 1:numel(symbols)
  code_length = length(codes(symbols(i)));
  variance = variance + (probabilities(i) * (code_length-average_code_length)^2);
end

% Redundancy and Efficiency calculation
redundancy = average_code_length - entropy;
efficiency = (entropy / average_code_length) * 100;

% Display results
fprintf('Entropy: %.5f bits\n', entropy);
fprintf('Average Code Length: %.5f bits\n', average_code_length);
fprintf('Redundancy: %.5f bits\n', redundancy);
fprintf('Efficiency: %.5f %%\n', efficiency);

% Save compressed data
save('compressed_data.mat', 'compressed_data');

% Decode the compressed data
decode_map = containers.Map(values(codes), keys(codes));
encoded_str = strjoin(compressed_data, '');
decoded_data = zeros(1, length(data));
current_code = '';
decoded_idx = 1;
```

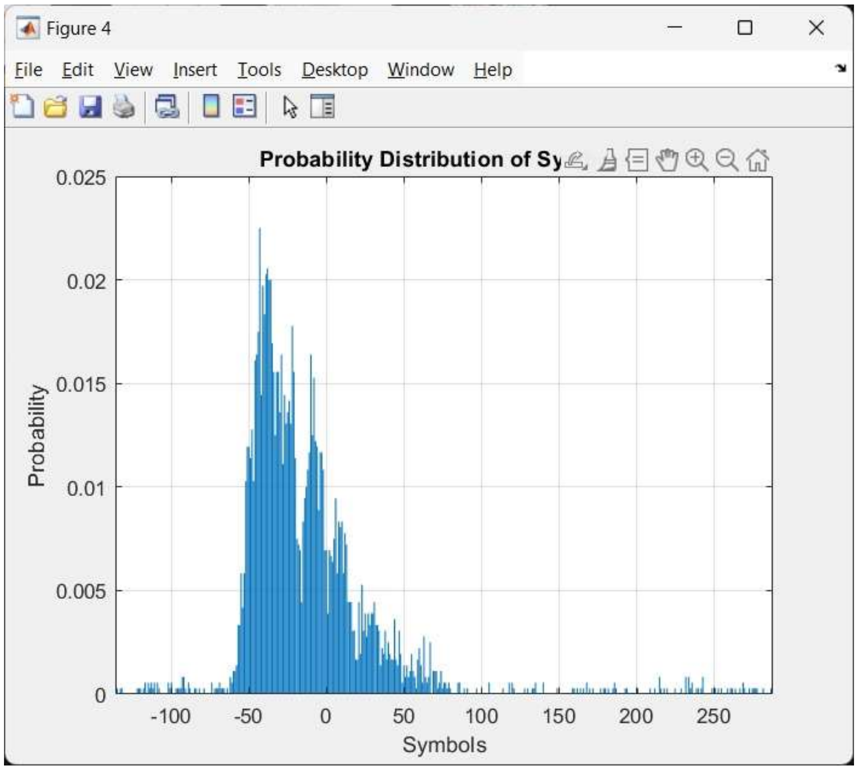
```
for i = 1:length(encoded_str)
    current_code = [current_code, encoded_str(i)];
    if isKey(decode_map, current_code)
        decoded_symbol = decode_map(current_code);
        decoded_data(decoded_idx) = decoded_symbol;
        decoded_idx = decoded_idx + 1;
        current_code = '';
    end
end

% Verify decoding success
if isequal(data, decoded_data)
    disp('Decoding successful. The decoded data matches the original data.');
else
    disp('Decoding failed. The decoded data does not match the original data.');
end

% Plot the decoded signal
subplot(2,1,2);
plot(time_axis, decoded_data);
xlabel('Time (seconds)');
ylabel('Voltage (mV)');
title('Decoded ECG Signal');
grid on;

% Probability distribution bar graph
figure;
bar(sorted_symbols, sorted_probabilities);
xlabel('Symbols');
ylabel('Probability');
title('Probability Distribution of Symbols');
grid on;
```

CHAPTER - 7
RESULTS AND DISCUSSION

```
Entropy: 6.83886 bits
Average Code Length: 6.86667 bits
Variance: 1.88500
Redundancy: 0.02780 bits
Efficiency: 99.59510 %
```



INPUT AND DECODED GRAPH



PROBABILITY OF SYMBOLS

PhysioNet

About the data of the ECG Signal used

The data comprises 18 long-term recordings of single-lead ECGs and associated 3-axis accelerometer data, collected from 15 subjects (9 female, 6 male) aged between 21 to 83 years. Recordings were carried out between August 2018 and October 2019 while the subjects were undertaking ordinary everyday activities.

Three signals were fully annotated in terms of ECG signal quality. The remaining 15 signals were annotated in two selected segments, each of 20 minutes in duration. Furthermore, five additional segments of poor signal quality were also annotated.

Methods: We recorded 18 signals longer than 24 hours using the Bittium Faros 180 device (ECG recorder with integrated accelerometer) under free-living conditions. By "free-living conditions", we mean that the subjects were carrying out ordinary everyday activities. The only limitation was that water activities such as bathing, showering, and swimming were avoided to protect the Bittium Faros 180 recorder. The database is broadly balanced in terms of gender (10 female records and 8 male records) and age (21 to 83 years, mean 41 years, median 37 years). The signals were recorded from 15 subjects (9 women and 6 men) with 13 subjects being monitored only once, one woman being monitored twice, and one man being monitored three times.

The database is intended for the development and objective comparison of algorithms designed to assess the quality of ECG records. One of the unique features of this database is that the quality of the ECG signals is annotated sample-by-sample (the signals have varying quality during time). Besides ECG signals, the dataset includes information about motion of the subjects - data from 3-axis accelerometer.

CHAPTER - 8
CONCLUSION

This MATLAB implementation demonstrates effective ECG signal compression through Huffman coding, achieving high compression efficiency. By reducing redundancy and average code length close to the signal's entropy, the Huffman coding method retains data integrity in compression and decompression. The results confirm Huffman coding's suitability for ECG data, offering efficient storage with no information loss.

CHAPTER - 9
REFERENCES

https://ieee-dataport.org/documents/ecg-signals-744-fragments

https://physionet.org/content/butqdb/1.0.0/100001/#files-panel