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 SHANNON-FANO CODING FOR ECG SIGNALS USING MATLAB

A report submitted by

Name of the Student	***Daniel Dencil J, Gladwin Jebas J***
Register Number	***URK22EC1010, URK22EC1031***
Subject Name	***Digital Communication***
Subject Code	***22EC2016***
Date of Report submission	***28/10/24***

Total marks: _____/ 40 Marks

Signature of Faculty with date:

TABLE OF CONTENTS

CHAPTER 1

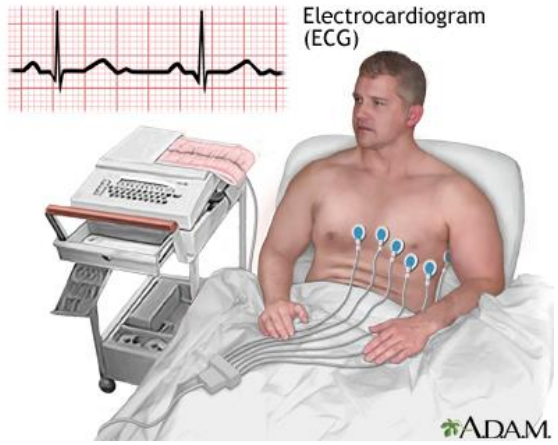
INTRODUCTION

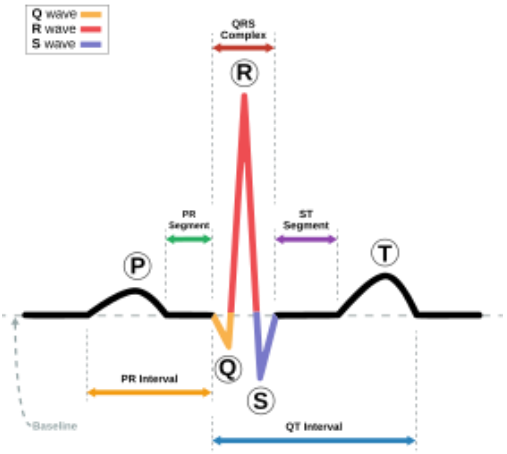
What is ECG?

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.

The efficient encoding of electrocardiogram (ECG) signals is critical for medical diagnostics, telemedicine, and wearable health devices, where minimizing storage and transmission costs is essential. This project focuses on implementing Shannon-Fano coding, a foundational entropy-based encoding algorithm, for compressing ECG signals. Shannon-Fano coding reduces data redundancy by assigning shorter codes to more frequent signal values, making it an ideal approach for compressing ECG data without compromising accuracy.

In this project, MATLAB is used to process ECG signal data due to its powerful signal processing capabilities, while Python handles the coding logic, facilitating cross-platform compatibility and future scalability. By leveraging Shannon-Fano coding, this implementation aims to provide an effective solution for ECG data compression, ultimately supporting more efficient medical data management.





CHAPTER 2

PROBLEM STATEMENT

- Current data storage and transmission methods for ECG signals are often inadequate due to the large sizes of the datasets involved.
- Conventional compression techniques may either lead to loss of crucial diagnostic information or fail to exploit the statistical properties of the ECG signals effectively.
- As a result, there is a significant gap in the availability of robust, lossless compression algorithms tailored specifically for ECG data.
- This project seeks to address these challenges by implementing Shannon-Fano coding, a lossless data compression algorithm known for its efficiency in encoding variable-length codes based on the probabilities of occurrence of symbols in a dataset.
- By applying this method to ECG signals, we aim to achieve a substantial reduction in data size while retaining the original signal's integrity, thus facilitating more efficient storage and transmission.

CHAPTER 3

MARKET SURVEY

- With the growing demand for efficient healthcare data management and remote patient monitoring, the compression of ECG data is becoming increasingly important.
- Traditional ECG systems generate large volumes of data, posing challenges in storage, transmission, and processing costs.
- Wavelet Transform-Based Compression, Discrete cosine Transform Compression, are some of the existing encoding techniques in market.
- Shannon-Fano coding is also an encoding technique which is used in this project.

Technique	Compression Type	Compression Ratio	Complexity	Lossless/Lossy	Applications
Shannon-Fano Coding	Entropy-based	Moderate	Moderate	Lossless	Suitable for moderate compression in simple ECG
Run-Length Encoding (RLE)	Entropy-based	Low–Moderate	Low	Lossless	Repetitive ECG data
Huffman Coding	Entropy-based	Moderate–High	High	Lossless	Complex and varying ECG data
Wavelet Transform (DWT)	Transform-based	High	High	Lossy	High compression in storage/transmission
Discrete Cosine Transform	Transform-based	High	Moderate	Lossy	Data reduction with acceptable signal quality
Arithmetic Coding	Entropy-based	High	Very High	Lossless	Applications needing maximum compression
AZTEC	ECG-specific	Very High	Moderate	Lossy	Real-time ECG monitoring, wearable devices

Figure1: comparison table of existing models with SHANNON-FANO

CHAPTER 4

SHANNON-FANO IMPLEMENTATION

Mathematical model

Problem:

Suppose we have a set of symbols S={A,B,C,D,E} with the following probabilities:

P(A)=0.4,P(B)=0.2,P(C)=0.2,P(D)=0.1,P(E)=0.1

Step 1: Sort Symbols by Probability

List the symbols in descending order of their probabilities:

A:0.4, B:0.2, C:0.2, D:0.1, E:0.1

Step 2: Recursive Partitioning

Split the list into two parts such that the sum of probabilities in each part is approximately equal.

1. **First Split:** Divide the list into two groups with near-equal probability sums:
    - Group 1: A (0.4)
    - Group 2: B,C,D,E(0.6)

    Assign:

    - Group 1: Code "0" for A
    - Group 2: Code "1" for B, C, D, E
2. **Second Split (on Group 2):** Now, take Group 2 and divide it further:
    - Group 2.1: B, C (0.4)
    - Group 2.2: D, E (0.2)

    Assign:

    - Group 2.1: Code "10" for B and "11" for C
    - Group 2.2: Code "110" for D and "111" for E

Step 3: Assign Codes to Each Symbol

Symbol	Probability	Code
A	0.4	0
B	0.2	10
C	0.2	11
D	0.1	110
E	0.1	111

Step 4: Calculate Average Code Length

The average code length L can be calculated using:

L=$\sum_{i=1}^{N} P(si)$×length of C(si)

Substituting in values:

L= (0.4×1) +(0.2×2) +(0.2×2) +(0.1×3) +(0.1×3)

L=0.4+0.4+0.4+0.3+0.3=1.8 bits

Step 5: Calculate Entropy H

The entropy H provides the theoretical minimum for the average code length:

H= $\sum_{i=1}^{N} P(si)$(log P(si)/log2)

Calculating for each symbol

H= (0.4×log0.4/log2) + (0.2× log0.2/log2) + (0.2× log0.2/log2) + (0.1× log0.1/log2) + (0.1× log0.1/log2)

After calculating each term, we get:
H≈1.85

Step 6: Calculate Efficiency
The efficiency of the Shannon-Fano coding is:
Efficiency=H/L=1.85/1.8≈1.03

CHAPTER 5

SHANNON FANO ALGORITHM

1. Data Acquisition

- **Objective:** Obtain ECG signal data that will be compressed using Shannon-Fano coding.
- **Details:** Use publicly available ECG datasets or generated ECG signals (e.g., MIT-BIH Arrhythmia Database) to ensure a standardized approach for testing and comparison.
- **Tools:** MATLAB for data loading and processing.

2. Preprocessing of ECG Signals

- **Objective:** Prepare the raw ECG signal for effective compression.
- **Steps:**
    - Normalize the ECG signal values if necessary, to fit within a specific range.
    - Segment the signal into manageable data blocks if it's a continuous long-duration recording.
    - Quantize the signal to reduce redundancy, mapping it to discrete amplitude levels, which will be treated as symbols for coding.

3. Probability Calculation

- **Objective:** Calculate the probability of each unique symbol (quantized amplitude level) within the ECG data.
- **Steps:**
    - Calculate the frequency of each unique symbol.
    - Determine the probability P(si) of each symbol as

$$P(s_i) = \frac{\text{frequency of } s_i}{\text{total number of symbols}}.$$

4. Shannon-Fano Coding Implementation

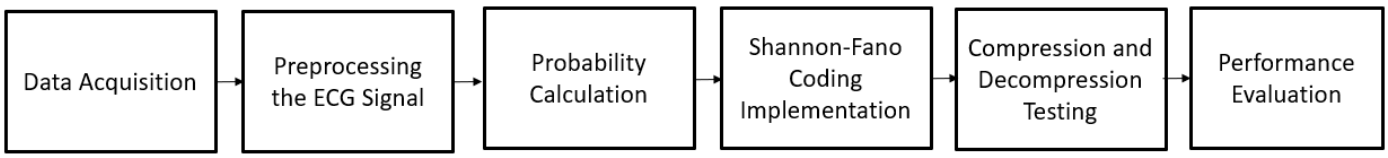
- **Objective:** Encode the ECG symbols using the Shannon-Fano algorithm.
- **Steps:**
    - Sort symbols by probability in descending order.
    - Recursively partition symbols into two groups with equal probability sums, and assign binary codes ("0" or "1") to each group.
    - Repeat until each symbol has a unique binary code.
- **Tools:** Write a MATLAB script to implement the Shannon-Fano algorithm, assigning codes to each symbol in the ECG dataset.

5. Compression and Decompression Testing

- **Objective:** Test the efficiency of the Shannon-Fano coding by compressing and then decompressing the ECG signal.
- **Steps:**
  - Apply the Shannon-Fano codes to compress the ECG signal.
  - Reconstruct the original ECG signal by decoding the compressed data.
- **Performance Metrics:** Track the compression ratio and bit rate.

6. Performance Evaluation

- **Objective:** Evaluate the effectiveness of Shannon-Fano coding for ECG data.
- **Metrics to Calculate:**
  - **Compression Ratio:** Ratio of the compressed size to the original size.
  - **Reconstruction Error:** Measure the difference between the original and decompressed signals to assess fidelity.
  - **Entropy and Efficiency:** Calculate the entropy HHH and compare it with the average code length LLL to assess how close the coding is to theoretical limits.
- **Tools:** MATLAB functions for error calculation, comparison, and analysis.



CHAPTER 6

CODE

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

sampling_frequency = 360;
time_axis = (0:length(data)-1) / sampling_frequency;
plot(time_axis, data);
xlabel('Time (seconds)');
ylabel('Voltage (mV)');
title('ECG Signal');
grid on;

% Unique symbols and their probabilities
symbols = unique(data);
symbols = reshape(symbols, [], 1); % Ensure symbols is a column vector
bin_edges = [symbols; max(symbols) + 1];

% Calculate probabilities based on the occurrence of each symbol
probabilities = histcounts(data, bin_edges) / numel(data);

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

% Sort nodes by probability in descending order
[~, sort_idx] = sort(cellfun(@(x) x.probability, nodes), 'descend');
nodes = nodes(sort_idx);

% Generate Shannon-Fano codes using containers.Map to avoid issues with field names
function codes = shannon_fano_codes(nodes)
   codes = containers.Map('KeyType', 'double', 'ValueType', 'char');
   generate_codes(nodes, '', codes);
end

function generate_codes(nodes, code, codes)
   if numel(nodes) == 1
      % If there's only one node, assign the code to that symbol
      codes(nodes{1}.symbol) = code;
   else
      % Find the splitting point to balance the probabilities
      total_prob = sum(cellfun(@(x) x.probability, nodes));
      cumulative_prob = 0;
      split_idx = 0;

      for i = 1:numel(nodes)
```

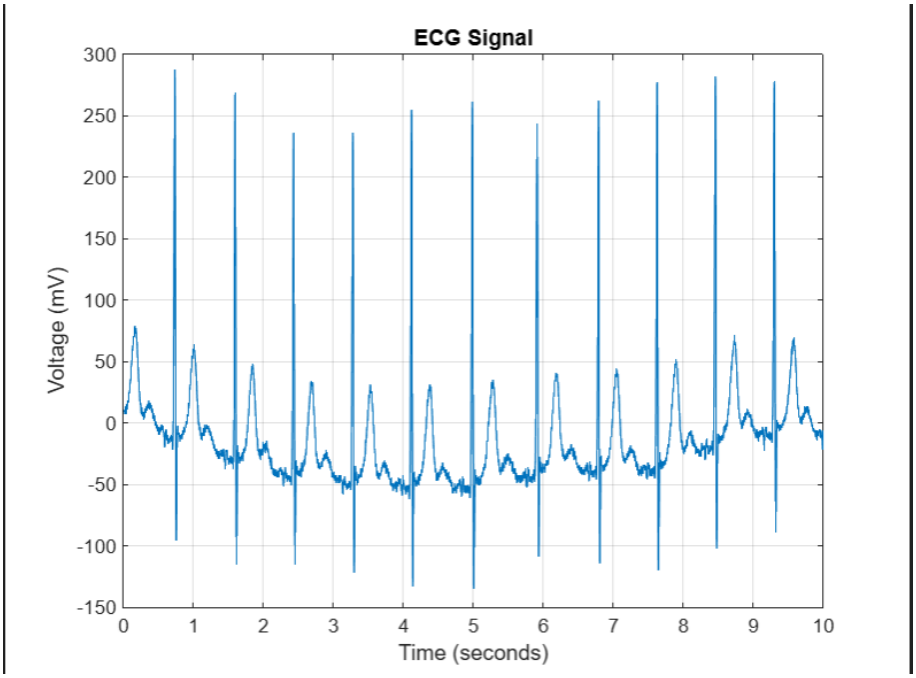
```
            cumulative_prob = cumulative_prob + nodes{i}.probability;
            if cumulative_prob >= total_prob / 2
                split_idx = i;
                break;
            end
        end

        % Split into two groups
        left_nodes = nodes(1:split_idx);
        right_nodes = nodes(split_idx+1:end);

        % Recursively assign '0' to the left group and '1' to the right group
        generate_codes(left_nodes, [code '0'], codes);
        generate_codes(right_nodes, [code '1'], codes);
    end
end

% Generate Shannon-Fano codes
codes = shannon_fano_codes(nodes);

% Compress the data using the generated Shannon-Fano codes
compressed_data = arrayfun(@(x) codes(x), data, 'UniformOutput', false);

% Sort symbols and probabilities from most probable to least probable
[sorted_probabilities, sorted_indices] = sort(probabilities, 'descend');
sorted_symbols = symbols(sorted_indices);

% Display the Shannon-Fano codes for each symbol, sorted by probability
for i = 1:numel(sorted_symbols)
    fprintf('Symbol: %d, Probability: %.5f, 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

% Redundancy calculation
redundancy = average_code_length - entropy;

% Efficiency calculation
efficiency = (entropy/average_code_length)*100;

% Display results
```

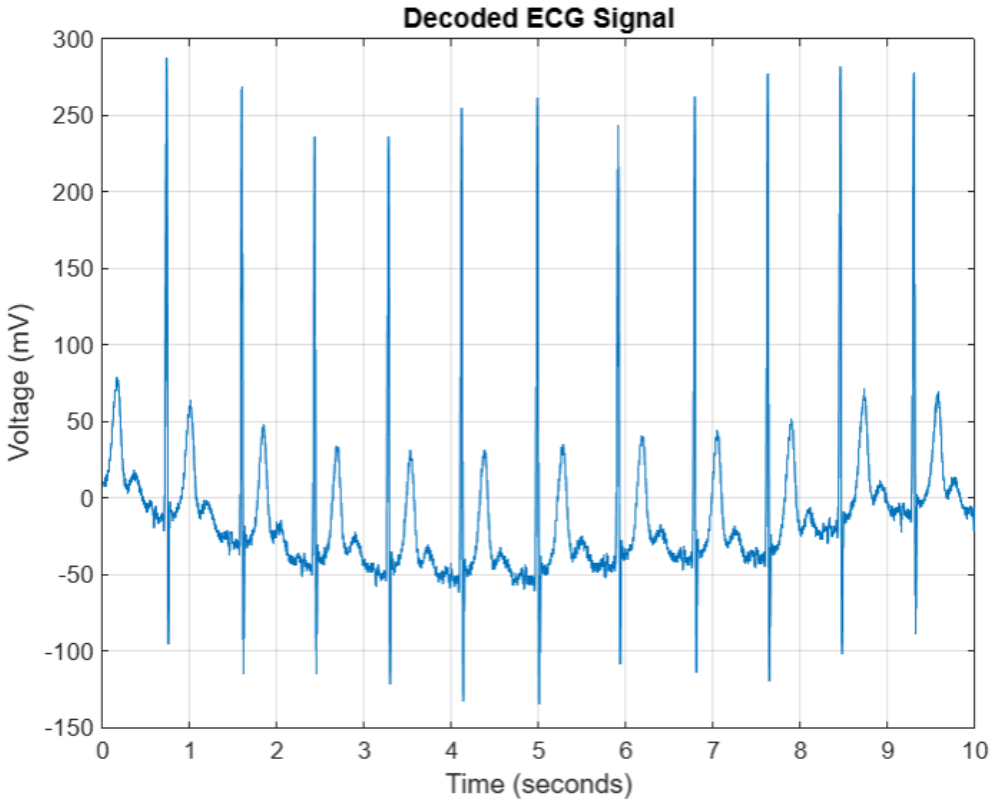
```
fprintf('Entropy: %.5f bits\n', entropy);
fprintf('Efficiency: %.5f %%\n', efficiency);
fprintf('Average Code Length: %.5f bits\n', average_code_length);
fprintf('Redundancy: %.5f bits\n', redundancy);

% Save the compressed data as a cell array of binary strings
save('compressed_data.mat', 'compressed_data');

% Load the compressed data and the original symbols
load('compressed_data.mat'); % Load the compressed data
symbols = unique(data);       % Reuse the symbols from the original data
symbols = reshape(symbols, [], 1);  % Ensure it's a column vector

% Reverse lookup: Build a map of binary codes to symbols
reverse_codes = containers.Map('KeyType', 'char', 'ValueType', 'double');
for i = 1:numel(symbols)
    reverse_codes(codes(symbols(i))) = symbols(i);
end

% Decode the compressed data using the reverse lookup table
decoded_data = zeros(1, numel(compressed_data));  % Pre-allocate the array for speed
for i = 1:numel(compressed_data)
    decoded_data(i) = reverse_codes(compressed_data{i});  % Map the binary string back to its
symbol
end

% Verify if the decoded data matches the original data
if isequal(decoded_data, 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

% Optional: Plot the decoded signal to visually compare with the original signal
figure;
plot(time_axis, decoded_data);
xlabel('Time (seconds)');
ylabel('Voltage (mV)');
title('Decoded ECG Signal');
grid on;

% Create a bar graph of sorted symbols and their probabilities
figure;
bar(sorted_symbols, sorted_probabilities);
xlabel('Symbols');
ylabel('Probability');
title('Probability Distribution of Symbols');
grid on;
```

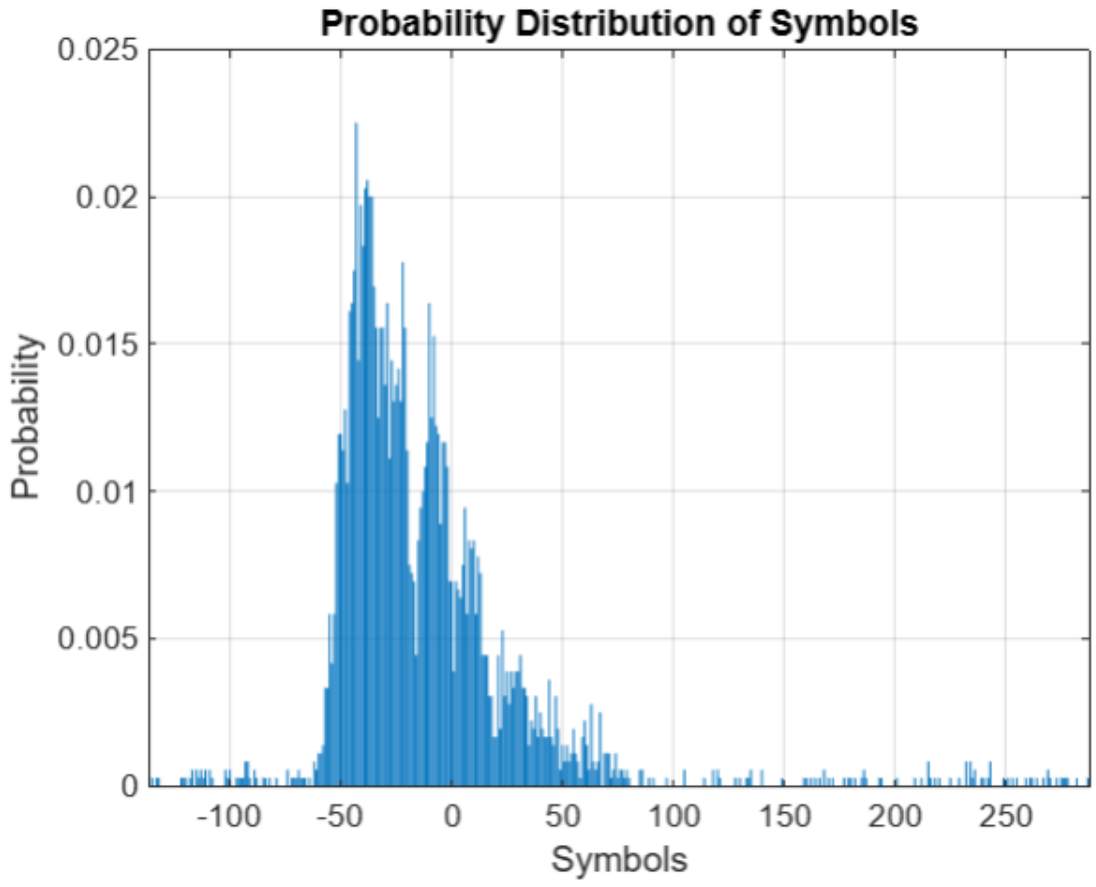
CHAPTER 7

RESULTS AND DISCUSSION

```
Entropy: 6.83886 bits
Efficiency: 99.01033 %
Average Code Length: 6.90722 bits
Redundancy: 0.06836 bits
Decoding successful. The decoded data matches the original data.
```




Decoded ECG Signal
Voltage (mV)
Time (seconds)
300
250
200
150
100
50
0
-50
-100
-150
0
1
2
3
4
5
6
7
8
9
10


Probability Distribution of Symbols
Probability
Symbols
0.025
0.02
0.015
0.01
0.005
0
-100
-50
0
50
100
150
200
250

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

The implementation of Shannon-Fano coding for ECG signal compression successfully reduces data storage requirements and transmission costs while preserving signal fidelity and minimizing error.

Successful decoding confirms that the reconstructed signal closely matches the original, validating Shannon-Fano coding as a reliable approach for ECG data compression with reduced error.

Overall, this project highlights the potential of Shannon-Fano coding to enhance ECG data management and serves as a basis for future research and applications in medical data compression.

CHAPTER 9

REFERENCES

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

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