Photoplethysmogram – Compression Methods

José Miguel Dias Simões
Departamento de Engenharia Informática
FCTUC
Coimbra
jds.work.one@gmail.com

Leandro Borges Pais
Departamento de Engenharia Informática
FCTUC
Coimbra
leandropaisslb1@gmail.com

João Francisco Borges Cruz
Departamento de Engenharia Informática
FCTUC
Coimbra
joaocruz63@live.com.pt

***Abstract* – In the field of medicine, Photoplethysmography is quickly becoming a very important technique for assessing vascular parameters (such as blood oxygen volume and blood pressure). [1] Because of this increase in use, it has become necessary to find an effective method for compressing the data. Lossless compression methods result in the best reproduction of the original signal, but their compression rates are lower, and they aren't as effective for storage. Lossy compression methods promise lower file size, at a cost in fidelity, regarding the original signal. In this paper, two methods are compared. Comparison involves calculating the square-difference (PRD) and the Compression Rates (CR) of both methods, using samples obtained via test groups. A combined method, using second-order Delta and Huffman encoding[2] (lossless) yields good reconstruction (PRD 0.042), albeit with a lower compression rate (CR 2.22) [2] when compared to a second proposed method, which involves Downsampling and the application of a grouping algorithm[3]. The latter produces better results as far as compression goes (CR 122.24) at a good reproduction rate of the original signal (PRD 0.02). Our own method, built around the lossless method we researched, has a CR of 1.288, after being exported as a .csv file, while being completely lossless, using a mix of second Delta encoding, Huffman codes and Sign-Byte redundancy-increasing algorithms. Better compression could be obtained using different storage methods.**

Keywords – PPG, Encoding, Compression

I. INTRODUÇÃO

O PPG (Photoplethysmogram) é uma técnica não-invasiva de medição ótica usada para medir a alteração da na pressão sanguínea através da medição do volume dos tecidos microvasculares. Tem sido explorada nos anos recentes devido à procura de procedimentos mais económicos, portáteis e simples, e devido à disponibilidade de materiais em elevadas quantidades, o PPG tem vindo a ser explorado como alternativa a procedimentos invasivos. Tem tido um grande número de utilizações nos hospitais e também no mercado de consumidores. [1]

Como a tecnologia está a ganhar mais utilização, surge o problema do volume dos dados. Para guardar um elevado número de dados, é necessário efetuar compressão dos mesmos. No tratamento de dados clínicos é portanto importante garantir um método de compressão que resulte na maior redução possível do tamanho, cuidando que as perdas sejam mínimas.

Enquanto que tecnologias como o ECG (Electrocardiogram) receberam mais atenção, e já possuem vários métodos de compressão extremamente eficazes, para o PPG ainda não houve tanta atenção. Serão demonstrados dois métodos de compressão para o sinal PPG, e será exposta uma solução possível.

II. MÉTODO DO Δ DE SEGUNDA ORDEM (CODEC LOSSLESS)

A. PASSO DO Δ DE SEGUNDA ORDEM

Este método de compressão é baseado numa combinação do método de Δ de segunda ordem e da codificação de Huffman. O principal objectivo deste método é garantir um algoritmo pouco complexo com níveis de compressão moderados.

Neste método, primeiro usa-se o método Δ de segunda ordem[2] [fig.1] num array de um sinal amostra PPG para codificação. Os valores delta representam a diferença entre amostras consecutivas, onde χ (i) são as amostras de PPG, Δ1 é a primeira diferença e Δ2 é a segunda. Como parte do algoritmo, no array Δ1 usa-se uma técnica onde se compara os valores absolutos do menor número e do maior número de forma a evitar que os números negativos afetem o algoritmo e modifica-se o array de acordo com os resultados. No array Δ2 foi aplicada a mesma técnica.

$$\Delta_1(i) = x(i+1) - x(i)$$
$$\Delta_2(i) = \Delta_1(i+1) - \Delta_1(i)$$

Fig. 1 – Equações para o Delta de Segunda Ordem[2]

B. PASSO DA CODIFICAÇÂO DE HUFFMAN

A codificação de Huffman foi aplicada no array modificado Δ2, calculando a probabilidade de cada símbolo[2]. Depois disso, utilizando símbolos da tabela de Huffman, o seu respectivo número de bits por símbolo e o código de Huffman foram comprimidos.

Os símbolos de Huffman foram comprimidos por nibble (agrupamento de bits quatro a quatro) ou comprimidos com um *offset* de valor conhecido, criando assim um array de símbolos

comprimidos. Finalmente o bit stream de símbolos de Huffman comprimidos serão convertidos em bytes através de sequencialização de oito bits. Será adicionado um zero, se necessário, para a cadeia ter comprimento par (zero padding). Esta informação será guardada em packets, e cada packets terá um header byte [fig.2] que permitirá a reconstrução do sinal. [2]

Header byte	Information	Bit allocation
hb 1	Number of bytes in the packet	5 bits
	Zero padding in Huffman symbol byte	3 bits
hb 2	Bias information for Δ_1 and Δ_2	2 bits
	Upper nibble of original sample	2 bits
	Number of Huffman symbols	4 bits
hb 3	Zero padding of Huffman code stream for $b\Delta_2$	3 bits
	First element of $b\Delta_1$	5 bits

Fig. 2 – Estrutura dos Header Bytes[2]

C. DESCODIFICAÇÂO

O array de dados codificados foi descodificado por uma sequência reversa à usada na sua codificação.

Primeiro descodificaram-se os header bytes para obter a informação essencial para descartar os zeros que foram adicionados para facilitar a codificação do sinal. Depois retiram-se dos header bytes a informação necessária para reverter a codificação de Huffman usada no array modificado Δ2. [2]

Finalmente obtemos os arrays originais Δ1 e Δ2, e usa-se o somatório de Δ1 para reconstituir a amostra do PPG.

D. ANÁLISE DA COMPRESSÃO

Usando 30 amostras de sinais PPG vindo de 30 pessoas diferentes[2]:

Os valores da compression ratio (CR), calculada dividindo o tamanho dos dados do input inicial pelo tamanho dos dados comprimidos, tendem para uma média de 2.223. Os valores da média da diferença percentual da raiz (PRD) e a sua versão normalizada (PRDN), para averiguar a diferença entre o sinal original e o sinal reconstruído, dão respectivamente 0.042 e 0.214.

$$CR = \frac{file\ \ size\ \ of\ \ original\ \ PPG\ \ data}{file\ \ size\ \ of\ \ compressed\ \ PPG\ \ data}$$

$$PRD\% = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\overline{y_i})^2}{\sum_{i=1}^{n} y_i^2}} \times 100\% \qquad PRDN = \sqrt{\frac{\sum_{i=1}^{n}(y_i-\overline{y_i})^2}{\sum_{i=1}^{n}(y_i-S_{mean})^2}} \times 100\%$$

Fig. 4 – Fórmulas de cálculo para a análise de dados[2]

O valor médio de CR é 2.22, de PRD é 0.042 e PRDN é 0.214. [2]

Conclui-se que, admitindo uma perda de dados média muito baixa, consegue-se uma compressão bastante eficaz. A perda de dados, neste caso, pode não ser significativa.

III. MÉTODO DO AGRUPAMENTO (CODEC LOSSY)

A. DE-NOISING

Numa fase inicial, é necessário eliminar o ruido do sinal, usando um filtro low-pass (25Hz). Acima de 15Hz, o sinal do PPG não contém informação relevante, logo, utiliza-se este filtro para remover ruídos como os da interferência eléctrica. [3]

B. DOWNSAMPLING E TRUNCAÇÃO

De seguida, o sinal é truncado com precisão de 2 casas decimais. Um sinal original de 500Hz a 16bits/sinal será reduzido a 100Hz, o que resultará numa redução significativa do volume dos dados. Esta redução tem precisão de até 3 casas decimais. Subsequentemente, o sinal será truncado para ter precisão até 2 casas decimais. Esta truncação é efectuada de modo a facilitar o agrupamento numa fase seguinte do método de compressão. [3]

C. CONSTRUÇÃO DO ARRAY DE 2ª DIFERENÇA e SIGNAL-BYTE

É importante notar que este método funciona de 8 em 8 samples, ou seja, num dado instante, os seguintes tratamentos são aplicados a 8 amostras.

É construído um array, de seguida, constituído pelas segundas-diferenças do sinal PPG truncado. Este array terá todos os seus valores amplificados de modo a representarem inteiros.

Agora, será gerado o byte de sinal. Todos os valores no array serão tornados positivos após ser guardada uma string binária com a informação relativa ao sinal dos valores no array.

D. APLICAÇÃO DO ALGORITMO DE AGRUPAMENTO

É feito um agrupamento, usando um algoritmo com cinco tipos [fig.4]: [3]

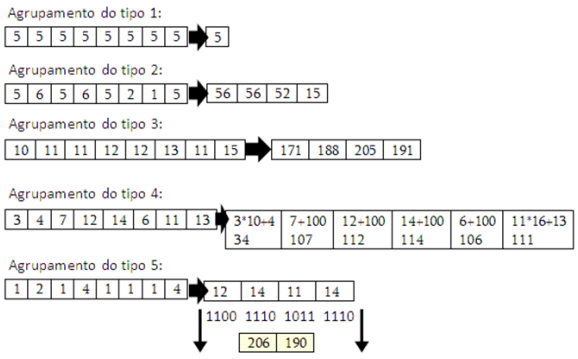

Fig. 5 – Exemplos do algoritmo de agrupamento

a) O primeiro tipo de agrupamento resulta numa compressão de conjuntos de símbolos iguais.

b) O segundo tipo de agrupamento agrupa símbolos dois a dois, se ambos forem inteiros menos que 10. O primeiro será multiplicado por 10, e o seu vizinho ser-lhe-á adicionado (o resultante será um valor entre 0 a 99).

c) O terceiro tipo de agrupamento ocorre quando os valores vizinhos estiverem entre 10-15. O primeiro inteiro será

multiplicado por 16 e adicionado ao vizinho (o resultante será um valor entre 170 a 255).

d) O quarto tipo é utilizado quando nenhuns dos 3 superiores são aplicados. Se, entre dois, apenas um dos vizinhos estiver entre 10-15, 100 será adicionado a ambos os valores. O valor deste inteiro agrupado será entre 100 e 115. Estes novos inteiros serão guardados num novo array.

e) O quinto tipo é uma extensão do segundo tipo. Se o segundo tipo ocorrer, e os inteiros agrupados forem todos menores ou iguais a 15, cada um desses inteiros será convertido no seu correspondente binário (4bits). Cada par de inteiros será representado pela sua combinação (2x4bits, 8bits, e será guardado o numero em decimal representado pelos 8bits).

E. ENCRIPTAÇÃO ASCII

Agora, é necessário encriptar o resultado. É escolhida uma Key (0-255) e uma operação XOR é utilizada em cada inteiro no array de inteiros agrupados. Estes dados encriptados são escritos como caracteres ASCII num ficheiro, juntamente com o sign-byte. [3]

O sinal foi guardado com sucesso. É necessário agora desencriptar, reverter o agrupamento, repor a informação do sinal e restaurar o sinal original. Por fim, será aplicada uma interpolação

F. RESTAURAÇÃO DO SINAL

A desencriptação é efectuada utilizando a mesma chave com que se efectuou a encriptação. O primeiro valor será o sign-byte, e os outros valores, serão os inteiros agrupados. [3]

De seguida, serão efectuadas as operações inversas às efectuadas no agrupamento. Através de uma verificação valor-a-valor, é decidido qual a operação a utilizar[4] (verifica-se a que conjunto de números pertence o inteiro, e deste modo conseguimos averiguar que operação a utilizar, visto que cada uma resulta num inteiro agrupado num conjunto específico, e único a essa operação.)

Agora, é reposto o sinal de cada inteiro. O sign-byte é convertido no seu binário (8bits), e se nessa string binária existir um 1, o seu correspondente (em posição) no array será multiplicado por -1.

Após a reposição do sinal, cada inteiro terá de ser dividido por 100, para reverter a amplificação efectuada anteriormente. Temos agora um array com as segundas-diferenças. Será convertida na primeira-diferença, e depois será convertida mais uma vez no sinal original (downsampled).

A seguir, o sinal terá de ser restaurado, após a sua redução de 500Hz para 100Hz. Através de algoritmos de interpolação cúbica, nearest-neighbour entre outros; o sinal é restaurado. É de notar que a interpolação linear é a mais eficaz[3]. Temos agora o sinal restaurado[4]

G. ANÁLISE DA COMPRESSÃO

Para avaliarmos a eficácia deste processo, temos de efetuar os cálculos já explicitados anteriormente.

O valor médio de CR é 122.24, PRD é 0.02 e PRDN é 0.03. [3]

IV. MÉTODO ELABORADO - COMPRESSÃO

A. BREVE DESCRIÇÃO DO MÉTODO

Para este método, foram efectuadas ligeiras modificações ao método primeiramente utilizado (método do delta de segunda ordem) [2]. Reduz-se à compressão do array de segundas diferenças, com um algoritmo "*Move-to-Front (MTF)*" subsequentemente aplicado, antes da codificação de Huffman. Este passo extra permite-nos comprimir mais os dados; e trata-se de um método completamente reversível.

B. DE-NOISING (LOW-PASS FILTER)

Os dados iniciais foram analisados, e obteve-se o cálculo da entropia do sinal, com o valor de 8.858339.

Neste passo é aplicado um filtro low-pass ao sinal lido do ficheiro '.csv'. Este filtro usa uma *cutoff-frequency* de 18MHz, que remove todos os ruídos indesejados do sinal. Acima de 18MHz, o sinal PPG não apresenta informação relevante para a reconstrução do sinal, mas pode apresentar ruídos provenientes da interferência eléctrica, como foi referido previamente.

Analisando o resultado verifica-se uma entropia de valor 15.872699, após a filtração.

C. ALGORITMO MTF (MOVE-TO-FRONT)

De seguida, aplicar-se-á uma interpretação do algoritmo MTF. Este algoritmo reestrutura os dados de maneira a que os dados transformados sejam mais compressíveis nos passos seguintes (nomeadamente na codificação de Huffman). É um método completamente reversível, que, quando aplicado eficientemente, é um passo extra extremamente útil na compressão por métodos de entropia. No entanto, usando o MTF somos também obrigados a guardar o dicionário original de símbolos.

Numa primeira fase, obtém-se o alfabeto ordenado dos

input_str chars	output_arr	list
p	15	abcdefghijklmnopqrstuvwxyz
a	15 1	pabcdefghijklmnoqrstuvwxyz
n	15 1 14	apbcdefghijklmnoqrstuvwxyz
a	15 1 14 1	napbcdefghijklmoqrstuvwxyz
m	15 1 14 1 14	anpbcdefghijklmoqrstuvwxyz

Fig. 5 – Funcionamento básico do MTF, usando a string "panam" e o alfabeto latino.

símbolos da fonte. Lendo o primeiro símbolo da fonte, é guardado o seu índice (no alfabeto); e subsequentemente o mesmo símbolo é colocado na primeira posição do alfabeto. Este processo repete-se até termos todos os índices.

Com este método, garante-se que os símbolos mais frequentes serão representados por um índice menor no alfabeto, o que facilitará a compressão com os métodos de compressão entrópicos, como por exemplo, a codificação de Huffman; facilitando o acesso aos símbolos mais frequentes.

Analisando o resultado verifica-se uma entropia de valor 14,561507 após o MTF.

D. MÉTODO DO DELTA DE SEGUNDA ORDEM

Neste passo, é aplicado o cálculo das segundas diferenças. É calculada a diferença entre valores adjacentes e essa diferença é guardada num array. De seguida, o mesmo procedimento é aplicado a este array de primeiras diferenças. [2]

$$\Delta_1(i) = x(i+1) - x(i)$$
$$\Delta_2(i) = \Delta_1(i+1) - \Delta_1(i)$$

Fig. 6 – Equações para o Delta de Segunda Ordem[2]

Analisando o resultado verifica-se uma entropia de valor 8,645734 após o Delta encoding.

E. SIGN-BYTES

De modo a reduzir a quantidade de símbolos existentes no alfabeto, aplica-se este método, convertendo todos os negativos em positivos, eliminando o uso de símbolos simétricos e aumentando a redundância do sinal [3].

Guarda-se num array binário a informação relativa ao sinal de cada elemento. Neste array binário, as posições correspondentes aos números negativos serão postas a 1, e as correspondentes a números positivos serão postas a zero. Na reconstrução, será facilitada a regeneração dos sinais, multiplicando por "-1" pelo valor do índice correspondente ao do sinal a reconstruir.

Sinal	1.32	1.0	0.5	-1	-2	0	-1	-2
Sign-Byte	0	0	0	1	1	0	1	1

Fig. 7 – Funcionamento e geração do Sign-Byte

Esta informação será incluída juntamente com o ficheiro principal depois da compressão, e o seu tamanho será também usado no cálculo final da compressão, pois trata-se de um dado essencial para a reconstrução[3].

Analisando o resultado verifica-se uma entropia de valor 7,692549 após a aplicação do Sign-Byte, que corresponde com o aumento da redundância.

F. CÒDIGOS DE HUFFMAN

Para terminar a compressão, utilizam-se os códigos de Huffman. Esta codificação *lossless* resulta numa tabela de códigos de comprimento variável, cada um representando um símbolo distinto na fonte. Baseia-se na frequência de cada símbolo e no alfabeto extraído da fonte.

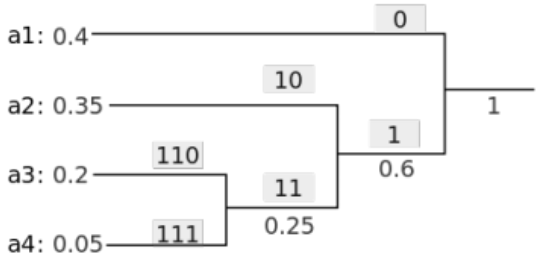

Fig. 8 – Exemplo da codificação de Huffman para quatro valores

Numa primeira abordagem, é calculada a frequência de cada símbolo no sinal. Aos símbolos mais comuns serão atribuídos símbolos com menor comprimento (geralmente com menos de dois bits para os símbolos mais comuns). É criada uma árvore binária, com os símbolos mais comuns sendo os que necessitam menos bits para ser representados, garantindo assim que o comprimento final do sinal será significativamente inferior ao comprimento original.

De seguida, o sinal comprimido será guardado num ficheiro, onde aguardará a descompressão [4]

Analisando o array binário devolvido verifica-se uma entropia de valor 0.998520 após o encoding de Huffman.

V. ANÁLISE DE RESULTADOS E DISCUSSÃO

Analisando o tamanho final do ficheiro, verificamos uma redução bastante significativa do tamanho do ficheiro. Visto que o sinal comprimido foi escrito bit a bit no ficheiro final; dividindo o tamanho final por 8 indicará o tamanho verdadeiro do ficheiro (que seria escrito byte a byte). Após isso temos ainda que somar o tamanho do ficheiro onde guardámos o alfabeto.

O ficheiro original tinha um tamanho de 753KB, com 60.000 amostras. Após todos os algoritmos serem aplicados, o ficheiro que contém as segundas diferenças codificadas tem um tamanho final de 990KB, que após ser dividido por 8 e adicionando 462KB do alfabeto, resulta num tamanho de 585KB, de onde podemos calcular o valor da CR, que é aproximadamente 1.288, que supera o valor obtido através do encoding das segundas diferenças com os códigos de Huffman, sem mais métodos aplicados.

Não foi possível guardar também o dicionário de Huffman, visto que este se encontra guardado num tipo especifico de ficheiro do *Matlab*, mas este teria um tamanho muito reduzido, visto que o formato .MAT guarda os dados formatados, e não alteraria de modo significativo a compressão do ficheiro.

Apesar de reduzirmos o tamanho, é de esperar que escrevendo os valores do array na forma de bits conseguiríamos um ficheiro muito mais comprimido. Da forma actual, escrevendo cada valor do array resultante da codificação de Huffman (um 0 ou 1 por célula) o ficheiro tem um tamanho de 585KB. Se os elementos deste array fossem entendidos como bits e escritos 8 a 8 na memória (como um byte), o tamanho final do ficheiro seria cerca de 65052B, ou seja, aproximadamente 64KB. O compression ratio para este sinal seria muito elevado.

A escrita sob o formato CSV comprimiu os dados necessários em 585KB, que, apesar de não ser muito pequeno, é uma

compressão satisfatória, que seria significativa em conjuntos de dados maiores.

O método desenvolvido é completamente reversível e lossless, utilizando métodos de optimização para a encoding de Huffman (nomeadamente, o Move-to-Front) que resultam numa compressão mais eficaz; e utilizando algoritmos de aumento da redundância do sinal, diminuindo a quantidade de símbolos do alfabeto, o que em troca, aumenta a eficácia da compressão de Huffman.

Este resultado é razoável, considerando que todos os métodos usados são lossless e reversíveis. Ainda poderíamos ter efectuado mais tratamento aos dados, aplicando-lhes mais algoritmos de optimização, para aumentar mais a CR. No entanto, os resultados teriam diferenças pouco significativas, relativamente ao que já foi conseguido. Aumentando a redundância, conseguimos optimizar efectivamente a compressão de Huffman; mas esta apresenta as suas limitações.

Apesar de termos conseguido aumentar a compressão usando métodos lossless, ainda ficamos muito distantes da compressão por métodos lossy, como a compressão através do Método de Agrupamento, previamente analisado[3]. Um sinal comprimido através de métodos lossy, apresenta uma compressão muito elevada, mas sacrifica a fidelidade, que é uma qualidade importante num sinal médico. [4]

VI. CONCLUSÕES E PESQUISA FUTURA

O sinal PPG possui poucos avanços na área da compressão, quando comparado com outros sinais, como o ECG, e durante a nossa análise e subsequente elaboração de um método de compressão, verificámos que há muitas vertentes adequadas à exploração de novos algoritmos para a compressão eficaz do sinal. O aumento crescente da necessidade de guardar grandes quantidades de dados relativos a este sinal leva a pesquisa a encontrar novos e mais eficazes métodos de compressão.

Foram apresentados dois métodos para a compressão de dados relativamente ao sinal PPG, que poderão facilitar a sua utilização no futuro, e foi apresentado um método, criado utilizando partes de cada um dos trabalhos previamente analisados.

Através da comparação, nota-se que o método de compressão com perdas (Método do Agrupamento e Downsampling) é o que resulta numa melhor compressão. As técnicas de interpolação utilizadas resultam numa grande proximidade ao sinal original, e a truncação das frequências não causa qualquer perda de dados relevantes ao sinal PPG. [3]

Podemos, com estes dados, entender o estado da arte da codificação de sinais PPG, estudando ambos os métodos.

Esta pesquisa levou-nos a misturar elementos de dois algoritmos previamente estudados[3]. Sabendo que todas as operações foram reversíveis, a CR de 1.22 é um resultado satisfatório.

Como vimos anteriormente, um sinal comprimido através de algoritmos lossy atingiu um CR de 122,24[3]. Apesar de ter sido comprometida a fidelidade do sinal, poderão ser feitos desenvolvimentos na área da descompressão de sinais codificados com métodos com perdas; e poderá ser possível recriar com elevada fidelidade o sinal original, com recurso a métodos matemáticos de interpolação sofisticados, capazes de restaurar, com erro mínimo, o sinal original.

A nossa análise de métodos que envolviam Downsampling mostrou que há já algoritmos de interpolação linear capazes de restaurar com uma grande precisão o sinal; e como vimos, o estudo das técnicas aplicadas a métodos lossy podem ser reutilizadas e aproveitadas nos métodos lossless, tal como foi reutilizado o método do Sign-Byte, que já fora utilizado no método de Agrupamento previamente estudado[3].

Portanto, uma abordagem futura que poderia ser explorada, seria a utilização de métodos de compressão muito pouco lossy, desde que estes se verificassem muito mais eficazes na compressão que métodos lossless. Ainda há outros algoritmos que poderiam ser aplicados na compressão lossless para melhorar mais a CR, como por exemplo, um algoritmo de agrupamento que conseguisse organizar todos os dados e reduzir, por exemplo, as áreas onde o sinal tem valores muito próximos.

Descartámos a ideia da truncação[3], pois resultava numa diferença muito significativa relativamente ao sinal original, mas poderíamos talvez adequar o método de agrupamento usado no Método de Agrupamento com Sign-Byte e aprimorar se o resultado final justificava a elevada complexidade computacional adicionada ao algoritmo[3].

Ainda, adicionando ao que já foi dito, poderíamos pesquisar um método de escrever os dados sob a forma binária num ficheiro binário, o que permitiria aumentar exponencialmente a compressão. Seria preciso um algoritmo bastante complexo de descodificação, mas esta parece ser uma via possível para aumentar a CR, num método lossless.

REFERÊNCIAS

[1] – Allen J "Photoplethysmography and its application in clinical physiological measurement" in Physiological Measurement Vol 28, Number 3, IOP Publishing 20th February 2007.

[2] – Gupta, Rajarshi "Lossless compression technique for real-time photoplethysmographic measurements" in IEEE Transactions on Instrumentation and Measurement, Vol 64, Number 4, April 2015.

[3] – S. Dhar, S.K. Mukhopadhyay, S. Pal and M. Mitra "An efficient data compression and encryption technique for PPG signal" in Measurement, Vol 116, pg 533-542, 2018.

[4] –S.K. Mukhopadhyay, M.O.Ahmad and M.N.S. Swamy "ASCII-character-encoding based PPG compression for tele-monitoring system" in Biomedical Signal Processing and Control, Vol 31, pg 470-482, 2017.