Análise de algoritmos de compressão em sinais de ECG utilizando um sistema embarcado

Tharlysson Breno Lima de Menezes[1], **Edward David Moreno Ordonez**[2]

[1] Departamento de Ciência da Computação
Universidade Federal de Sergipe (UFS) – São Cristóvão, SE – Brasil

{tharlyssonbreno,edwdavid}@gmail.com

Abstract. *Digital Systems aplicated in medicine are each day more common. Many times, it is necessary to decide the best form to capture, processs, store and transfer the referent data to the analysis made by each systems. In the present work, the focus will be the electrocardiograph, specificlally the decreasement of the size of the data captured.*
In this work we analysed which compression algorithm had the best behavior using a Raspberry Pi 3 about the memory consume, timing and processing applied. Besides that, we made a comparison between them and a discussion of which fitted best for each situation.

Resumo. *Sistemas digitais aplicados à medicina estão cada vez mais comuns. Muitas vezes, é preciso decidir qual a melhor forma de capturar, processar, armazenar e transferir os dados referentes às análises feitas por tais sistemas. O foco foi aparelhos de Eletrocardiograma, em específico a diminuição do tamanho dos dados capturados por eles.*
Neste trabalho, analisamos qual algoritmo de compressão teve melhor comportamento utilizando um Raspberry Pi 3 em relação ao consumo de memória, tempo e processamento utilizados. Além disso, foi feita uma comparação entre eles e uma discussão de qual melhor se adéqua para cada situação.

1. Introdução

Sistemas digitais que processam sinais biomédicos utilizados em clínicas, hospitais, laboratórios, ambulatórios, etc. São dispositivos responsáveis por capturar os dados, processá-los e armazená-los. Tais aparelhos podem ser utilizados por um longo período de tempo criando assim um amplo banco de dados que futuramente poderá servir como base de dados para pesquisas ou análises, o que leva à necessidade de planejarmos e otimizarmos o processamento e armazenamento destes dados, afim de poupar recursos computacionais e energéticos utilizados, tentando ao máximo minimizar os gastos, em busca da eficiência do processo como um todo.

Para nos auxiliar na diminuição do tamanho dos dados armazenados podemos utilizar técnicas de compressão. Atualmente classificamos os diversos algoritmos que fazem parte deste ramo de pesquisa computacional em duas categorias [Mahoney 2010], são elas:

- Compressão sem perda de dados (*lossless data compression*) [Holtz and Holtz 2002]: Esta classificação se refere aos métodos em que a

informação obtida após a descompressão é idêntica à informação original antes de ser comprimida. Exemplo arquivos ZIP, RAR, 7z, etc.

- Compressão com perda de dados (*lossy data compression*) [Salomon and Motta 2009]: analogamente à citada anteriormente, neste caso a informação obtida após a descompressão é diferente da original, o que não quer dizer que os dados que obteremos após este processo sejam inúteis, pois a perda é previamente estimada para que estes dados possam ser interpretados futuramente de uma forma útil. São exemplos de dados comprimidos com perdas: imagens em JPEG e áudios em MP3.

Ao sermos apresentados a essas duas formas de classificação uma pergunta que surge é: "Qual categoria utilizar?". Como se trata de um contexto critico que irá envolver diagnósticos médicos é prudente escolhermos métodos que se encaixem na categoria que não possua perda de dados após aplicarmos o algoritmo de compressão.

Para promover a portabilidade e o baixo consumo energético decidimos utilizar como plataforma computacional para aplicar os algoritmos de compressão nos dados coletados um *System on Chip* (SoC), que se trada de um sistema que agrupa todos os componentes de um computador em um circuito integrado portátil com um poder computacional superior ao de um microcontrolador [Martin and Chang 2002]. Além da possibilidade de compartilhar os dados capturados através da rede Wi-Fi permitindo assim o acesso remoto a esses dados promovendo uma comunicação entre os diversos dispositivos.

Em busca desse objetivo neste trabalho, analisamos algumas técnicas de compressão sem perda de dados que foram aplicadas em sinais reais de eletrocardiograma (ECG) tendo como objetivos fazer uma comparação de qual técnica melhor se comportaria com os dados. Além disto visando a portabilidade e diminuição do consumo energético executamos os testes destas técnicas em um dispositivo SoC.

1.1. Problematização

Para um diagnóstico médico eficaz é necessário um monitoramento intensivo do paciente, tomando por base um monitoramento cardíaco de 24 horas por dia, 7 dias por semana. Sabendo que é necessário o registro e armazenamento dos sinais coletados do ECG para futuros diagnósticos ou estudos de possíveis doenças cardíacas. O armazenamento dessas informações é um fator que pode ser minimizado afim de melhorar o desempenho deste sistema. A seguir será descrito um estudo de caso utilizando as métricas fornecidas por MIT-BIH *Arrhythmia Database* [Moody and Mark 2001].

Estudo de caso 1:

Um paciente em uma UTI é monitorado 24 horas por dia durante 7 dias, a taxa de coleta é de 360 amostras por segundo a uma resolução de 11 *bits*. Sabendo que um monitor de eletrocardiograma possui 12 canais a quantidade de dados gerados é dada por:

$$TamanhoTotal = T * Q * R * C \tag{1}$$

T = Tempo em segundos, Q = Taxa de amostras, R = Resolução da amostra em *bits*, C = Número de canais

Substituindo os valores na Equação 1, temos o tamanho total de 28.740.096.000 *bits* ou 3,59 GB.

É importante salientar que existem aparelhos que possuem tanto resolução, quanto taxas de amostragem superiores ao estudo de caso levantado anteriormente. A consequência é o aumento do número de *bits* necessários para armazenar esses dados. Logo é de grande importância que tentemos reduzir ao máximo o espaço necessário para armazenar essa quantidade de dados coletados, para isso aplicamos técnicas de compressão que visam diminuir o máximo possível a quantidade de *bits* necessários para armazenar o ECG. Com a redução do tamanho dos dados, podemos reduzir a memória de armazenamento local e a banda para transmissão destes.

1.2. Objetivos da Pesquisa

Este trabalho tem como objetivo geral, analisar o comportamento dos principais algoritmos de compressão de dados aplicados a sinais coletados por aparelhos de eletrocardiograma, utilizando um dispositivo embarcado para realizar este processo. E como objetivo específico eleger o algoritmo que melhor se comporta tendo como entrada sinais de eletrocardiograma, informar os resultados obtidos nos testes e compará-lo com os demais algoritmos apresentados no decorrer do trabalho.

1.3. Metodologia

A metodologia adotada para o trabalho envolveu, inicialmente, a revisão da literatura acerca do que é um ECG, da importância dele para o diagnóstico médico. Além do estudo dos algoritmos tidos como base para compreensão de como funciona o processo de compressão, são eles: Deflate, LZ77 e a codificação de Huffman. Em seguida foram estudados os algoritmos que seriam utilizados na realização dos testes, que são: Gzip, Bzip2, LZMA, 7-zip, XZ e o Zpaq. A escolha ocorreu devido a constante frequência da utilização destes algoritmos durante a pesquisa bibliográfica sobre compressão de dados.

2. Resultados e Análises

Para podermos elencar qual o algoritmo teve mais sucesso no quesito de compressão, calculamos a razão de compressão, RC, que nada mais é que a razão entre o tamanho do arquivo após a compressão e o tamanho do arquivo original, como pode ser visto na equação a seguir, lembrando que quanto menor for a razão de compressão melhor o algoritmo se saiu neste quesito.

$$RC = \frac{TamanhoComprimido}{TamanhoOriginal} X100\%$$

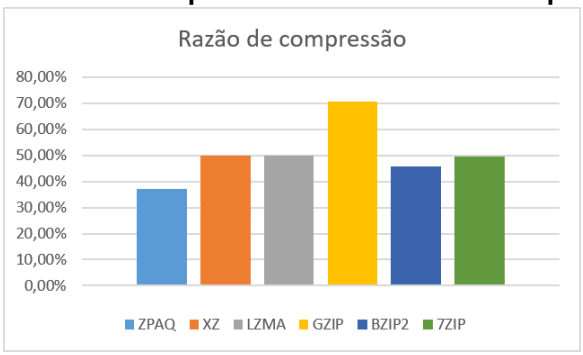
Table 1. Razão de compressão

Algoritmo	Tamanho original (bytes)	Tamanho comprimido (bytes)	Razão de compressão
Zpaq	93 969 920	34 920 402	37,16%
XZ	93 969 920	46 761 608	49,76%
LZMA	93 969 920	46 761 172	49,76%
Gzip	93 969 920	66 235 008	70,48%
Bzip2	93 969 920	42 960 803	45,71%
7-zip	93 969 920	46 622 343	49,61%

Como podemos observar na Tabela 1 três algoritmos (XZ, LZMA e 7-zip) tiveram comportamento semelhante com uma razão de compressão equivalente entre si, este resultado era esperado pois os três algoritmos utilizam como base o algoritmo LZMA para compressão. O Zpaq, por sua vez foi o que teve a menor razão de compressão com 37,16%, foi o que se comportou melhor em relação aos demais com quase 10% de vantagem para o segundo melhor que foi o Bzip2. A que teve o pior resultado foi o Gzip, com uma razão de compressão de 70,48%, aproximadamente o dobro do Zpaq que obteve a melhor razão de compressão e aproximadamente 20% superior aos demais concorrentes.

O gráfico apresentado na Figura 1 ilustra como se comportaram os algoritmos em relação à razão de compressão; importante ressaltar que quanto menor a razão, melhor o algoritmo se comportou.

Figure 1. Gráfico representando a razão de compressão



Além da razão de compressão também foi monitorado o consumo de memória RAM que os algoritmos utilizaram durante o processo. Como podemos ver na Tabela 2, os algoritmos XZ e LZMA tiveram comportamento semelhante: utilizando aproximadamente 97MB de memória, cada um deles. Isso dar-se pelo mesmo fato citado anteriormente, em que ambos utilizam o LZMA como método de compressão, a surpresa vem quando analisamos a memória utilizada pelo 7-zip, que é mais que o dobro da utilizada pelos XZ e LZMA. Apesar de, como citado anteriormente, eles utilizarem o mesmo processo de compressão.

Table 2. Memória utilizada na compressão

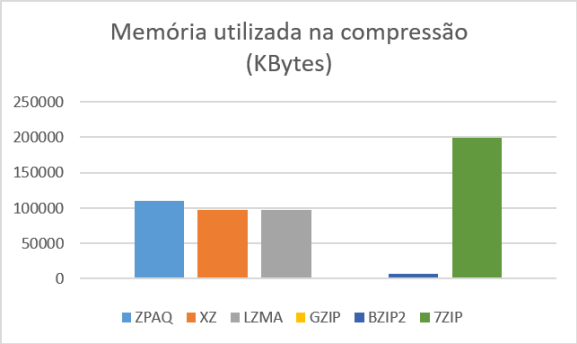
Algoritmo	Média (KBytes)	Desvio padrão	Intervalo de confiança
Zpaq	110771,00	66,42	± 69,70
XZ	96839,83	7,33	± 7,69
LZMA	97150,50	4,67	± 4,91
Gzip	1214,16	26,11	± 27,40
Bzip2	7200,16	40,57	± 42,58
7-zip	198941,70	104,44	± 109,60

O Zpaq manteve-se em um consumo próximo aos XZ e LZMA, já o Gzip e o Bzip2 utilizaram pouca memória em relação aos demais algoritmos citados, o destaque

ficando para o Gzip que teve o menor consumo de memória RAM, aproximadamente 1MB.

O gráfico apresentado na Figura 2 ilustra como se comportaram os algoritmos em relação ao consumo de memória do sistema enquanto executavam o processo de compressão, é relevante ressaltar que a memória é um recurso importante quando se trata de dispositivos embarcados, ver Tabela 2.

Figure 2. Gráfico representando a memória utilizada na compressão



No quesito tempo gasto na compressão, os algoritmos XZ e LZMA também se comportaram de maneira similar, precisando de aproximadamente 250 segundos para concluir a compressão. Porém, como na análise anterior, o 7-zip teve uma discrepância em relação aos dois, precisando de aproximadamente 165 segundos para efetuar a tarefa. Saindo um pouco dos algoritmos que tem como base a compressão utilizando o LZMA, vamos para o Zpaq: ele precisou de aproximadamente 1140 segundos para concluir a tarefa, mais do quádruplo de tempo que o segundo pior necessitou.

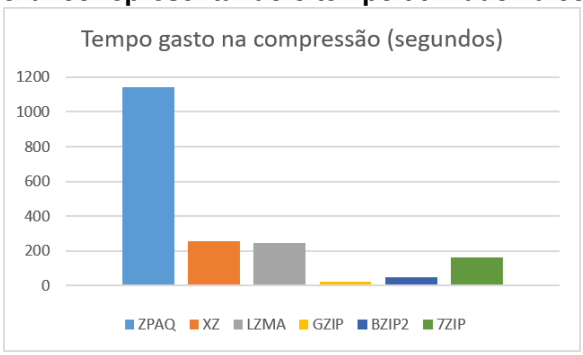
Table 3. Tempo gasto na compressão

Algoritmo	Média (segundos)	Desvio padrão	Intervalo de confiança
Zpaq	1139,50	1,87	± 1,96
XZ	255,83	1,47	± 1,54
LZMA	244,50	0,54	± 0,57
Gzip	24,83	0,40	± 0,42
Bzip2	51,33	1,21	± 1,27
7-zip	164,66	0,51	± 0,54

Ao contrário do Zpaq, os algoritmos Gzip e Bzip2 tiveram bons desempenho, tendo os dois melhores tempos, com destaque para o Gzip que demorou apenas 25 segundos para realizar a compressão.

No gráfico representado na Figura 3 fica bastante clara a discrepância de tempo que o Zpaq possui em relação aos demais algoritmos analisados, resultado do tipo de compressão utilizado pelo Zpaq, que necessita de diversas verificações durante o processamento dos dados antes de serem comprimidos, diferentemente de outras técnicas usadas para compressão de dados que trabalham com um fluxo contínuo de *bytes* tratados um a um, ver Tabela 3.

Figure 3. Gráfico representando o tempo utilizado na compressão



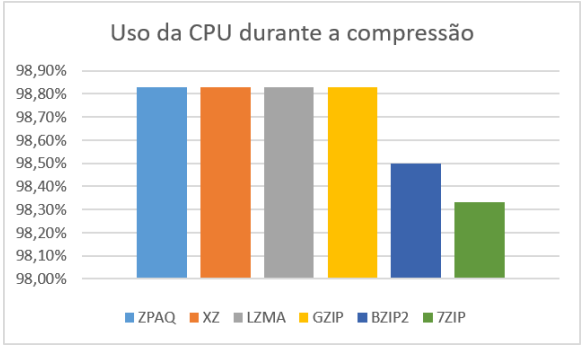
O último quesito analisado durante o processo de compressão foi o uso da CPU. Neste quesito, todos os algoritmos ficaram muito parecidos utilizando todo o recurso disponível para realizar a tarefa dada. O ponto fora da curva foi o consumo de CPU do 7-zip que, apesar de ter um consumo um pouco inferior aos demais, foi o único que utilizou *thread* do processador para executar a tarefa. Mesmo com este ponto, todos os algoritmos tiveram um desempenho bastante similar.

Table 4. Uso da CPU durante a compressão

Algoritmo	Média por CPU	Desvio padrão	Intervalo de confiança
Zpaq	98,83%	0,40	± 0,42
XZ	98,83%	0,40	± 0,42
LZMA	98,83%	0,40	± 0,42
Gzip	98,83%	0,40	± 0,42
Bzip2	98,50%	0,83	± 0,87
7-zip	98,33%	0,51	± 0,54

No gráfico representado na Figura 4 apesar de visualmente o 7-zip e o BZIP terem uma escala menor que os demais, se olharmos a porcentagem relativa de cada um podemos ver que a diferença entre todos os algoritmos analisados é mínima, ver Tabela 4.

Figure 4. Gráfico representando o uso da CPU durante a compressão



No processo de descompressão os algoritmos utilizaram pouca memória, com exceção do Zpaq que consumiu aproximadamente a mesma quantidade de memória que

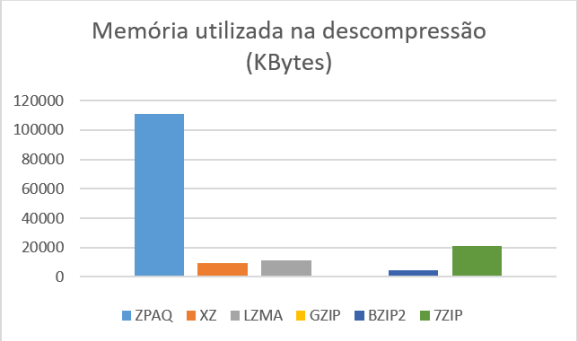
utilizou para a compressão, cerca de 110MB. O que necessitou de mais memória para poder realizar a descompressão foi o 7-zip. O destaque neste quesito fica novamente para o Gzip que, assim como durante o processo de compressão, foi o que utilizou menos memória RAM para realizar a tarefa, aproximadamente 1MB.

Table 5. Memória utilizada na descompressão

Algoritmo	Média (KBytes)	Desvio padrão	Intervalo de confiança
Zpaq	110751,50	40,58	± 42,59
XZ	9721,33	9,02	± 9,47
LZMA	11061,67	12,24	± 12,84
Gzip	1060,33	45,03	± 47,25
Bzip2	4286,66	30,76	± 32,28
7-zip	20991,67	37,92	± 39,80

No gráfico representado na Figura 5 vemos o quanto de memória o algoritmo Zpaq utilizou a mais em relação aos demais algoritmos analisados, ver Tabela 5.

Figure 5. Gráfico representando a memória utilizada na descompressão



Assim como o item anterior o que teve um desempenho inferior foi o Zpaq, que precisou de aproximadamente 1140 segundos para efetuar o processo de descompressão do arquivo. Como descrito na página do algoritmo [Mahoney 2009], ele necessita do mesmo tempo para descomprimir o arquivo que ele levou para comprimir. Os demais algoritmos se comportaram semelhantes no quesito de tempo gasto para descompressão, vale ressaltar que o Gzip teve o menor tempo, aproximadamente 10 segundos.

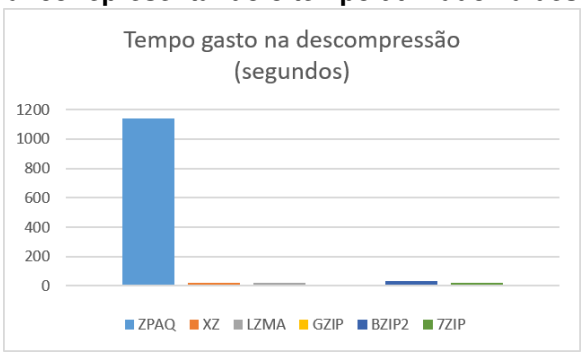
Como pode ser observado nos dados contidos nas Tabelas 3 e 6 o motivo do tempo de descompressão ser menor que o tempo gasto na compressão, na maioria dos algoritmos analisados, deve-se a como ele age durante o modo de compressão. A necessidade de criar as estruturas necessárias para comprimir demanda mais tempo, já no processo de descompressão só é necessário a leitura dessas estruturas tornando o processo mais rápido. A exceção fica com o Zpaq que a natureza do seu algoritmo impõe a necessita de diversas verificações durante o processo de descompressão.

Table 6. Tempo gasto na descompressão

Algoritmo	Média (segundos)	Desvio padrão	Intervalo de confiança
Zpaq	1139,33	2,16	± 2,26
XZ	19,66	1,21	± 1,27
LZMA	19,16	0,98	± 1,03
Gzip	9,50	1,37	± 1,44
Bzip2	31,83	1,47	± 1,54
7-zip	19,33	1,21	± 1,27

O gráfico representado pela Figura 6 ilustra como o Zpaq necessitou de um intervalo de tempo superior aos demais para realizar o processo de descompressão, ver Tabela 6.

Figure 6. Gráfico representando o tempo utilizado na descompressão



O último quesito para análise será o uso da CPU para descompressão, diferente do processo de compressão, com exceção do Zpaq que necessitou do máximo de CPU disponível para realizar o processo, os demais não atingiram o recurso máximo disponibilizado pelo hardware para realizar a tarefa, mostrando que o processamento não é um gargalo quando o objetivo é a descompressão. Isso dar-se ao fato de que durante o processo de descompressão só é necessária uma leitura das estruturas comprimidas para obter o arquivo descomprimido, diferente do processo de compressão que é necessário criar tais estruturas, como pode ser visto nas Tabelas 4 e 7.

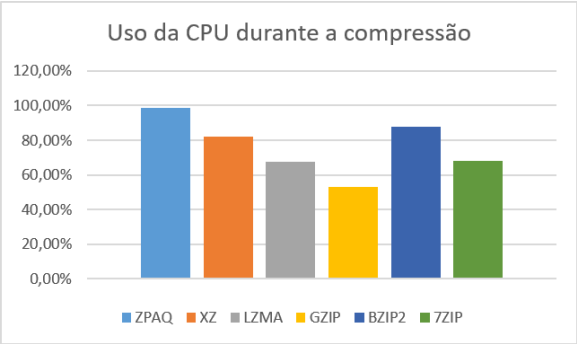
Table 7. Uso da CPU durante a descompressão

Algoritmo	Média	Desvio padrão	Intervalo de confiança
Zpaq	98,83%	0,40	± 0,42
XZ	82,16%	6,94	± 7,28
LZMA	67,50%	6,12	± 6,42
Gzip	53,16%	6,52	± 6,84
Bzip2	87,83%	5,07	± 5,32
7-zip	67,83%	5,63	± 5,91

O gráfico representado na Figura 7 ilustra o consumo de CPU dos algoritmos

analisados como nos critérios anteriores o Gzip foi o que menos utilizou recurso para realizar a tarefa, ver Tabela 7.

Figure 7. Gráfico representando o uso da CPU durante a descompressão



Abaixo podemos analisar a Tabela 8 que é um compilado dos resultados obtidos durante os testes aplicados.

Table 8. Resultados dos testes executados

	Zpaq	XZ	LZMA	Gzip	Bzip2	7-zip
Razão de compressão	37,16%	49,76%	49,76%	70,48%	45,71%	49,61%
Memória utilizada na compressão (KBytes)	110771,00	96839,83	97150,50	1214,16	7200,16	198941,70
Tempo gasto na compressão (segundos)	1139,50	255,83	244,50	24,83	51,33	164,66
Uso da CPU na compressão	98,83%	98,83%	98,83%	98,83%	98,50%	98,33%
Memória utilizada na descompressão (KBytes)	110751,50	9721,33	11061,67	1060,33	4286,66	20991,67
Tempo gasto na descompressão (segundos)	1139,33	19,66	19,16	9,50	31,83	19,33
Uso da CPU na descompressão	98,83%	82,16%	67,50%	53,16%	87,83%	67,83%

3. Conclusão

Após escolher o banco de dados que iria ser utilizado, elencar os algoritmos que seriam interessantes na realização dos testes e escolha da plataforma embarcada que seria utilizada na aplicação dos testes, podemos concluir que cada algoritmo se comportou de uma forma particular com os dados utilizados, com exceção do XZ e LZMA que tiveram desempenhos semelhantes em todos os quesitos analisados. Era esperado isto ocorrer, quando elegemos os algoritmos que faríamos os testes, colocamos os dois para confirmar que ambos utilizavam o mesmo algoritmo para compressão.

Quanto aos algoritmos, vimos que eles possuíam vantagens em determinados quesitos e desvantagens em outros. Um exemplo disto foi o Zpaq que se analisássemos somente a taxa de compressão ele seria nossa escolha, porém quando vamos analisar o tempo necessário para realizar o processo de compressão e descompressão ele torna-se inviável em relação aos demais. Com tempo de cerca de quatro vezes maior que os demais durante o processo de compressão e de vinte vezes maior durante o processo de descompressão.

Na linha intermediária ficaram os algoritmos XZ, LZMA e 7-zip, estes algoritmos possuem a mesma base de compressão: o algoritmo LZMA. Porém o 7-zip, como descrito no decorrer do trabalho, implementa outros algoritmos que o ajudam a tirar um pouco de vantagem dos outros dois citados. No processo de compressão, a taxa de compressão dos três foi próxima, porém no tempo gasto o 7-zip levou vantagem dos demais, mesmo que para isso ele tenha utilizado mais memória RAM, no processo de descompressão essa vantagem ficou evidente no tempo que ele demorou para realizar a tarefa, quase que a metade do tempo gasto pelos outros dois.

O Gzip é um caso particular, se destacou pelo seu aproveitamento dos recursos do hardware, sendo o que menos utilizou memória RAM, CPU e tempo para efetuar a compressão e a descompressão, porém sua taxa de compressão foi disparada a pior em relação aos outros.

O destaque ficou para o Bzip2 que possuiu a maior consistência nos testes; no processo de compressão ele ficou em segundo no quesito tempo, teve a segunda melhor taxa de compressão e o segundo melhor uso de memória RAM. No processo de descompressão não foi diferente, teve pouco consumo de memória e um leve aumento no tempo necessário para descomprimir, mas nada que impactasse sua eficiência em relação aos outros.

Analisando os resultados das simulações o algoritmo do compressor Bzip2 foi o algoritmo que apresentou ser mais viável, devido ao baixo consumo dos recursos computacionais, a sua razão de compressão em relação aos demais e o baixo tempo necessário para realizar o processo de compressão e descompressão.

Importante destacar que após a realização de todos os testes, notamos que o gargalo ficou na capacidade de processamento, apesar dos resultados satisfatórios que levantamos, ficou claro que durante o processo de compressão o uso da CPU esteve sempre no máximo.

Como trabalho futuro, uma sugestão é aplicar o algoritmo Bzip2 que melhor se destacou para transmitir os dados do dispositivo embarcado para a rede. Fazendo com que a análise dos dados coletados por profissionais da área de saúde seja mais rápida e eficiente.

References

Holtz, K. and Holtz, E. (2002). Lossless data compression techniques. *WESCON/94. Idea/Microelectronics. Conference Record.*

Mahoney, M. (2009). Incremental journaling backup utility and archiver.

Mahoney, M. (2010). Data compression explained.

Martin, G. and Chang, H. (2002). System-on-chip design. *ASIC, 2001. Proceedings. 4th International Conference on*.

Moody, G. and Mark, R. (2001). The impact of the mit-bih arrhythmia database. *IEEE Eng in Med and Biol*, pages 45–50.

Salomon, D. and Motta, G. (2009). Handbook of data compression, 5th edition. *Springer*, pages 16–18.