

**COMPRESSÃO DE DADOS DE MEDIDORES INTELIGENTES: UMA SOLUÇÃO
PARA REDUZIR E OTIMIZAR OS RECURSOS DO MEDIDOR DE ENERGIA EM
REDES NBIOT**

 <https://doi.org/10.56238/sevened2025.011-017>

Gleison Guardia

Matemático e Cientista de Dados (Mestrado)
Instituto de Ciência e Tecnologia da Evolução/Instituto Federal de Rondônia – IFRO
E-mail: gleison.guardia@ifro.edu.br
ORCID: orcid.org/0000-0003-1402-0777
LATTES: lattes.cnpq.br/3081488341816997

Rogério Guerra Diógenes

Engenheiro de Telecomunicações (Mestrado)
Núcleo de Estudos e Pesquisas do Norte e Nordeste – NEPEN
E-mail: rogerio.diogenes@nepen.org.br
ORCID: orcid.org/0009-0007-5461-7937
LATTES: lattes.cnpq.br/3782482898648331

Iury Gonçalves França

Analista e Desenvolvedor de Sistemas (em andamento)
Instituto de Ciência e Tecnologia da Evolução
E-mail: iurygfranca@gmail.com
ORCID: orcid.org/0009-0007-8535-0139
LATTES: lattes.cnpq.br/5608507491741074,

Lucas Noé dos Santos Santana

Engenharia Mecatrônica (em andamento)
Núcleo de Estudos e Pesquisas do Norte e Nordeste – NEPEN
E-mail: lucas.santana@nepen.org.br
ORCID: orcid.org/0009-0009-6236-1753
LATTES: lattes.cnpq.br/6086974571374063

Jamilly Cristina de Sousa

Engenheira de Produção Mecânica (Pós-Graduação)
Instituto de Ciência e Tecnologia da Evolução
E-mail: jamillycristina90@gmail.com
ORCID: orcid.org/0009-0001-8820-3165
LATTES: lattes.cnpq.br/6374960584977744

Brunna Conceição de Paulo

Engenharia da Computação (Cursando)
Instituto de Ciência e Tecnologia da Evolução
E-mail: brunnacdepaulo@gmail.com
ORCID: orcid.org/0009-0003-6448-1309
LATTES: lattes.cnpq.br/5168826144828472



Antônio Ébano Rafael Machado de Oliveira

Engenharia Mecatrônica (em andamento)

Instituto de Ciência e Tecnologia da Evolução

E-mail: eebanorafael@gmail.com

ORCID: orcid.org/0009-0006-7576-2722

LATTES: lattes.cnpq.br/9847201110211140

Alberto Alexandre Moura de Albuquerque

Engenheiro Eletricista (Mestrado)

Instituto de Ciência e Tecnologia da Evolução

alberto.alexandre@evolucaoinstituto.org.br

ORCID: orcid.org/0009-0003-1742-8652

LATTES: lattes.cnpq.br/1776411644533237

RESUMO

Este estudo propõe uma estratégia para otimizar a compressão de dados em dispositivos embarcados baseados no microcontrolador STM32WBA. O principal objetivo foi reduzir o volume de dados transmitidos em redes NB-IoT, com o objetivo de reduzir o volume desses dados sem comprometer a integridade da informação, ao mesmo tempo que respeita as restrições de memória e capacidade computacional destes sistemas. A pesquisa utilizou a biblioteca HEATSHRINK, baseada no algoritmo LZSS, ajustando parâmetros como o tamanho da janela de compressão (`window_size`) e o tamanho do lookahead (`lookahead_sz2`) integrado ao algoritmo BZ2 baseado na técnica Burrows-Wheeler Transform (BWT), para alcançar um equilíbrio entre taxa de compressão, tempo de processamento e consumo de memória. Os experimentos revelaram que configurações intermediárias, como HSWB06_HSLB04 e HSWB06_HSLB05, alcançaram taxas de compressão de até 46,93% para conjuntos de 96 amostras. Combinadas com uma redução de 21,15% no cabeçalho DLMS, essas configurações resultaram em um ganho total de compressão de 68,08%. Esses parâmetros também apresentaram tempos de processamento moderados, na faixa de 525 ms, e consumo de memória de 200 bytes, tornando-os adequados para dispositivos com recursos limitados. Por outro lado, configurações mais extremas, como HSWB10_HSLB09, apresentaram desempenho inferior, com taxas de compressão abaixo de 40% e tempos de processamento superiores a 7.900 ms, tornando-as inviáveis para sistemas que exigem baixa latência. Uma abordagem estratégica para alocar recursos de microcontroladores também foi implementada. Os dados originais foram armazenados em memória FLASH, os dados compactados na RAM e os buffers temporários foram gerenciados pelo STACK, garantindo estabilidade operacional mesmo sob condições de alta carga. A viabilidade dessas configurações foi confirmada por meio de análises de memória realizadas com a ferramenta Build Analyzer integrada ao STM32CubeIDE, que mostrou impacto mínimo nas operações do dispositivo. Os resultados obtidos destacam o potencial da biblioteca HEATSHRINK como uma solução eficiente para compressão de dados em dispositivos embarcados, desde que devidamente ajustada às especificidades da aplicação. Além disso, o estudo abre caminhos para avanços futuros, como a integração de técnicas de compressão híbrida com aprendizado de máquina, validações em ambientes reais e adaptações para sistemas de transmissão intermitente, como redes NB-IoT e LoRaWAN. Essas melhorias têm o potencial de expandir significativamente as aplicações de compressão em diversos setores, incluindo energia, saúde e transporte, promovendo maior eficiência energética, robustez e confiabilidade.

Palavras-chave: Compressão de dados. Dispositivos incorporados. Redes IoT. STM32WBA52CE.

1 INTRODUÇÃO

Este estudo foi conduzido pelo Instituto Evolução de Ciência e Tecnologia, em colaboração com o Núcleo de Estudos e Pesquisas Norte e Nordeste (NEPEN) e o Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, com o objetivo de investigar e implementar soluções mais eficazes para compressão de dados em redes de dispositivos embarcados.

A pesquisa se concentrou no uso do microcontrolador STM32WBA, utilizado em uma NIC com tecnologia NB-IoT adaptada para o aMeter WASION. O principal objetivo é minimizar o volume de dados transmitidos diariamente sem comprometer a integridade das informações, promovendo menor uso do espectro de frequências e economia no uso de dados, principal parâmetro utilizado pelas operadoras de telefonia para cobrança de tarifas.

O Wasion Group é líder em soluções avançadas de medição, distribuição inteligente e gerenciamento de eficiência energética, operando em mais de 50 países. Fundada em 2000 e listada na Bolsa de Valores de Hong Kong desde 2005, a empresa emprega mais de 6.000 pessoas e investe cerca de 9,4% de sua receita em P&D, destacando-se por sua inovação tecnológica. Seu portfólio abrange medidores de eletricidade, água e gás, sistemas de telecomunicações, energia renovável e distribuição inteligente, com mais de 100 milhões de dispositivos entregues globalmente. A Wasion possui fábricas em locais estratégicos como México, Brasil e Hungria, garantindo atendimento próximo e suporte técnico 24 horas. Reconhecida por sua qualidade e certificações como CE e KEMA, a empresa colabora com marcas renomadas como Siemens e ABB, reforçando seu compromisso com a eficiência energética e a inovação sustentável em escala global.

O projeto propõe uma abordagem inovadora para a redução significativa e sem perdas do tráfego de dados, utilizando o conceito de memória de massa de energia. Isso envolve a aplicação de técnicas de pré-processamento para reduzir o volume de informações antes da compactação, a adoção de bibliotecas específicas para compactação e descompactação de dados e a integração com uma aplicação web para transmissão e recuperação das informações.

A estrutura do artigo está organizada em seis seções principais. **A Seção 2** apresenta uma revisão da literatura, explorando estudos relevantes que sustentam as técnicas e soluções discutidas. **A Seção 3** descreve as características do hardware embarcado usado nos medidores, suas limitações e as especificidades dos dados coletados e transmitidos.

A seção 4 descreve em pormenor os desenvolvimentos técnicos realizados, incluindo os algoritmos e as bibliotecas utilizados, bem como a modelização e a execução das experiências. **A Seção 5** discute os resultados obtidos, analisando a eficácia das soluções propostas. Por fim, **a Seção 6** apresenta as conclusões do estudo, destacando suas contribuições e sugerindo possíveis direções para trabalhos futuros.

2 REVISÃO DA LITERATURA

A compressão de dados para dispositivos embarcados tem sido amplamente investigada em diferentes contextos, com foco na melhoria da eficiência energética, redução do consumo de recursos computacionais e otimização da transmissão de informações. Ketshabetswe et al. (2021) realizaram uma análise comparativa de algoritmos de compressão voltados para redes de sensores sem fio (WSN), destacando ALDC (Adaptive Lossless Data Compression) e métodos de compressão distribuída e agregação de dados. Este estudo resultou no desenvolvimento de um algoritmo capaz de reduzir o consumo de energia em até 76,8%.

No campo da compressão de dados textuais, Kodituwakku e Amarasinghe (2010) avaliaram algoritmos sem perdas, identificando as limitações do LZW em arquivos grandes e o desempenho robusto dos métodos de Huffman e Adaptive Huffman. Em paralelo, Sandoval et al. (2020) introduziram a decomposição tensorial para identificar padrões e anomalias em sistemas de potência, contribuindo para a operação eficiente desses sistemas. Zhang et al. (2023) e Zhang (2023) apresentaram RSDC (Real-Time Synchrophasor Data Compression), uma técnica em tempo real que aumentou as taxas de compressão e reduziu os atrasos nas redes elétricas.

Estudos de medidores inteligentes destacam abordagens práticas. Santos et al. (2023) investigaram medidores NB-IoT para reduzir fraudes em áreas urbanas e rurais, registrando um aumento de 134 kWh por unidade após a instalação de proteções, com 99% de eficiência na transmissão de dados. Meffe et al. (2023) implementaram uma solução de baixo custo para monitoramento remoto com LoRaWAN, proporcionando melhor cobertura em locais de difícil acesso e maior eficácia na detecção de fraudes.

Outras contribuições exploraram soluções específicas para compressão em dispositivos embarcados. Lee et al. (2016) propuseram um método sem perdas para tabelas de dados móveis, reduzindo o tamanho do arquivo sem comprometer a precisão. Moon et al. (2018) analisaram a compactação com perdas em dados espaço-temporais de IoT, examinando as compensações entre a redução de dados e a integridade das informações. Qin et al. (2020) introduziram o algoritmo CZ-Array para longos fluxos de dados em sensores com recursos limitados, superando gzip e bzip2 em taxas de compactação.

A adaptação de algoritmos clássicos para redes IoT também foi explorada em Júnior e Oyamada (2023) e Júnior et al. (2023) onde aplicaram métodos como LZ77, LZ78 e Huffman, alcançando reduções de até 22% no consumo de energia e 70% na compressão de dados.

Finalmente, Malandrino et al. (2024) contribuíram com abordagens para cidades inteligentes e redes neurais profundas. Gomes aplicou o aprendizado de máquina à compressão de sensores, melhorando a eficiência energética em 22% e reduzindo a latência. Malandrino desenvolveu o PACT

(Predictive Adaptive Compression Technique), ajustando dinamicamente a compressão em DNNs e reduzindo significativamente o consumo de energia.

Embora esta pesquisa progrida na otimização da compactação de dados e na eficiência energética, ela não aborda totalmente as restrições de memória, um aspecto crítico para dispositivos embarcados. Com base nesses estudos, este trabalho propõe uma solução inovadora para reduzir o tamanho dos arquivos transmitidos pelas placas controladoras, garantindo que o consumo de memória e a capacidade de armazenamento permaneçam compatíveis com as limitações de hardware durante a compressão e transmissão de dados.

3 INFRA-ESTRUTURA

3.1 SOBRE O HARDWARE

O STM32WBA é um microcontrolador sem fio de 32 bits desenvolvido pela STMicroelectronics, projetado para atender às necessidades de aplicações que exigem conectividade Bluetooth® Low Energy (BLE) e alta eficiência energética. Equipado com o núcleo Arm® Cortex-M33®, operando a até 100 MHz, o dispositivo combina desempenho de computação robusto com recursos avançados de segurança, como TrustZone® e Unidade de Proteção de Memória (MPU). Além disso, ele suporta instruções DSP e incorpora uma Unidade de Ponto Flutuante (FPU), recursos que o tornam adequado para processamento digital de sinais, veja a **figura 01**:

Figura 01: Medidores de wasion: À esquerda está o medidor completo, no centro está a parte superior do microcontrolador STM32WBA, no canto superior direito está a frente do microcontrolador e no canto inferior direito está o microcontrolador traseiro.



Fonte: Próprio autor

No campo da conectividade, o STM32WBA integra um transceptor de rádio de 2,4 GHz compatível com BLE 5.4 e protocolos adicionais como Thread, Matter, Zigbee® e soluções proprietárias. A sensibilidade de recepção do dispositivo é de -96 dBm para BLE a 1 Mbps e -97,5 dBm para IEEE 802.15.4 a 250 kbps, enquanto a potência de saída é programável até +10 dBm, em incrementos de 1 dB, oferecendo flexibilidade para várias condições de operação.

No que diz respeito à memória, o STM32WBA oferece até 1 Mbyte de memória flash com correção de erros (ECC), dos quais 256 Kbytes podem suportar 100.000 ciclos de gravação. Ele

também possui 128 Kbytes de SRAM, 64 Kbytes dos quais são verificados por paridade. A versão implementada em medidores inteligentes, no entanto, tem uma configuração simplificada, com 512 Kbytes de memória flash e 96 Kbytes de RAM, adequada para aplicações mais restritivas.

O microcontrolador foi projetado para operar em ambientes de baixo consumo de energia. No modo de espera, ele consome apenas 160 nA com 16 pinos de ativação e 6,5 μ A no modo Stop com 64 Kbytes de SRAM, tornando-o ideal para dispositivos alimentados por bateria.

Além do baixo consumo de energia, o STM32WBA é equipado com uma ampla gama de periféricos. Isso inclui um ADC de 12 bits com uma taxa de amostragem de até 2,5 Msps, comparadores de baixa potência, temporizadores de 16 e 32 bits e interfaces de comunicação como I2C, SPI, USART e SAI. Além disso, o dispositivo possui um controlador de toque capacitivo que suporta até 20 sensores, ampliando sua versatilidade para aplicações de interface sensíveis ao toque.

Essas especificações tornam o STM32WBA uma solução altamente adaptável e confiável, ideal para aplicações que vão desde a Internet das Coisas (IoT) até sistemas industriais. Sua combinação de conectividade robusta, eficiência energética e suporte para uma variedade de protocolos o posiciona como uma escolha estratégica para projetos que exigem alto desempenho e operação eficiente em dispositivos embarcados.

3.2 SOBRE OS DADOS

O estudo foi baseado no medidor trifásico Wasion, no qual os dados são transmitidos por meio de uma placa de interface de rede (NIC) para um módulo de coleta de dados (MDC). Durante esse processo, os dados relativos às medições do medidor são enviados na forma de um quadro de memória de massa, estruturado como uma matriz de bytes. Essa matriz encapsula as informações relacionadas às medições feitas pelo dispositivo.

O medidor de energia trifásico DTSD341, desenvolvido pela Wasion, é uma solução para consumidores residenciais e pequenos estabelecimentos comerciais, oferecendo medições precisas e confiáveis, veja a **figura 01**. Certificado com precisão classe 1.0 para energia ativa e classe 2.0 para energia reativa, o dispositivo suporta faixas de tensão de 3x120/240 V e 3x127/220 V, com tolerância de -20% a +20%.

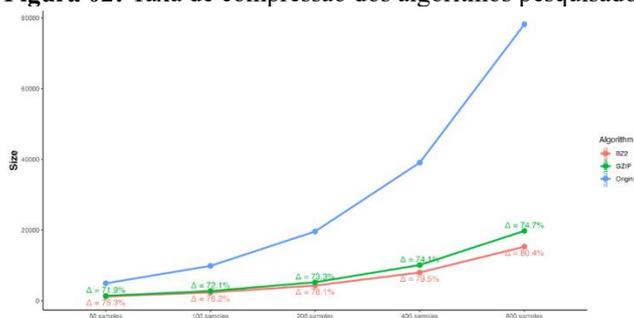
Equipado com um módulo de RF integrado para comunicação sem fio e uma opção de porta óptica, o DTSD341 facilita a leitura remota e a configuração de parâmetros, garantindo eficiência operacional. Seus recursos incluem suporte para até quatro tarifas diferentes, registro detalhado do perfil de carga com armazenamento de dados por 60 dias (em intervalos de 15 minutos) e detecção de eventos como abertura de tampa, reversão de corrente e falhas de fase. Seu display LCD grande e de fácil leitura, com suporte opcional para bateria interna ou supercapacitor, garante a visibilidade dos dados mesmo na ausência de energia. Além disso, a segurança dos dados é aprimorada pela

4 DESENVOLVIMENTO

Com o objetivo de avaliar e melhorar o desempenho deste trabalho, foi realizada uma extensa pesquisa bibliográfica em fontes renomadas, incluindo a Data Compression Conference (DCC), a base de dados IEEE (Institute of Electrical and Electronics Engineers), o portal de periódicos da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e os anais do SENDI (Seminário Nacional de Distribuição de Energia Elétrica) 2024. A pesquisa se concentrou exclusivamente em temas relacionados à compressão de dados em sistemas embarcados.

Com base nos estudos identificados, foram selecionados algoritmos para os testes iniciais, priorizando aqueles com potencial para atender às necessidades do sistema em termos de eficiência e desempenho. Com base nessa seleção, foi realizada uma análise detalhada da compressão aplicada ao armazenamento em massa, variando a quantidade de dados processados. O objetivo era encontrar o melhor equilíbrio entre volume de dados, taxa de compressão e desempenho do algoritmo. Os resultados obtidos para cada configuração testada são mostrados na **figura 02**.

Figura 02: Taxa de compressão dos algoritmos pesquisados



Fonte: Próprio autor

A análise revelou uma redução no volume de dados transmitidos variando de 75% a 80%, destacando a relevância da implementação proposta. Observou-se também que um aumento no número de arquivos acumulados resulta em ganhos incrementais na taxa de compressão. No entanto, o intervalo de tempo necessário para o envio dos arquivos mostrou-se um fator determinante nesse processo.

De acordo com os dados do sistema, acumular 100 leituras na memória de massa requer um período de aproximadamente 24 horas. Como o ganho percentual adicional na compactação não justifica o tempo adicional necessário para acumular mais dados, optou-se por padronizar em um intervalo de 24 horas como base para o envio. Essa decisão busca equilibrar a eficiência da compressão e o tempo de processamento, mantendo a viabilidade prática do sistema.

4.1 SOBRE O ALGORITMO BZ2

O algoritmo de compactação BZ2 é amplamente reconhecido por sua eficiência na compactação de dados sem perdas e é uma escolha frequente em sistemas Unix/Linux. Desenvolvido

na década de 1990, é baseado na técnica Burrows-Wheeler Transform (BWT), que reorganiza subcordas para facilitar a compressão. Comparado aos métodos tradicionais, como ZIP e GZIP, o BZ2 tem taxas de compactação mais altas em muitos casos, tornando-o uma solução eficaz para armazenar e transmitir dados \cite{burrows1994bwt}. Sua eficácia vem da combinação de técnicas avançadas, como BWT, Move-to-Front (MTF), codificação Run-Length (RLE) e codificação de Huffman, que otimizam o processo de compressão Seward (2024).

O funcionamento do BZ2 é composto por estágios estruturados que maximizam a compressão. Em primeiro lugar, a Transformada de Burrows-Wheeler (BWT) reorganiza os caracteres no texto, agrupando padrões repetitivos e preparando os dados para as fases subsequentes. Em seguida, o Move-to-Front (MTF) converte caracteres em índices com base em sua frequência, maximizando a repetição de símbolos Cleary e Witten (1984). A codificação de comprimento de execução (RLE) compacta sequências repetitivas, como transformar 'aaaaa' em '5a'. Finalmente, a codificação de Huffman atribui sequências mais curtas aos caracteres mais frequentes, reduzindo ainda mais o tamanho dos dados Knuth (1998). Apesar dessas vantagens, o BZ2 requer mais memória e poder de processamento, limitando sua aplicação em sistemas com recursos limitados.

O BZ2 é amplamente utilizado em práticas que exigem compactação eficiente. Em sistemas Unix/Linux, é a escolha padrão para a criação de arquivos compactados com extensão .bz2, especialmente em backups e logs, devido à sua capacidade de reduzir o tamanho de arquivos grandes e facilitar sua transferência Python Software Foundation (2024). Além disso, o BZ2 é ideal para compactar grandes volumes de dados antes da transmissão, reduzindo custos e otimizando processos. Ferramentas como tar oferecem suporte nativo para BZ2, integrando-o de forma eficiente aos fluxos de trabalho.

Neste estudo, o BZ2 foi avaliado quanto à sua viabilidade em sistemas embarcados, analisando o uso de ROM e memória RAM, bem como o custo operacional. No entanto, a natureza dos dados a serem compactados, armazenados em formato de byte, apresentou desafios significativos. Muitos algoritmos de compressão são otimizados para trabalhar com strings, valores hexadecimais ou números de ponto flutuante, enquanto os dados neste projeto exigiram uma abordagem específica. Essa limitação levou à experimentação com diferentes bibliotecas, buscando soluções que atendessem às restrições do sistema.

Após análise criteriosa, optou-se por adotar a biblioteca HEATSHRINK, disponível no repositório <https://github.com/atomicobject/heatshrink>. Desenvolvido para dispositivos embarcados, o HEATSHRINK é leve e eficiente, projetado para operar com formatos de dados como bytes, atendendo às necessidades específicas deste projeto. A escolha do HEATSHRINK permitiu contornar os desafios impostos pelos dados originais e possibilitou a implementação de uma compressão eficiente de acordo com as limitações e demandas do sistema.

4.2 SOBRE O HEATSHRINK

A biblioteca Heatshrink é uma solução projetada especificamente para compactação e descompactação de dados em sistemas embarcados e ambientes em tempo real. Sua principal característica é o baixo consumo de memória, com requisitos mínimos de apenas 50 bytes, bem como a capacidade de processar dados de forma incremental, otimizando o uso da CPU de forma controlada e eficiente. Essa flexibilidade torna o Heatshrink ideal para dispositivos com recursos limitados, permitindo alocações de memória estática ou dinâmica de acordo com as necessidades do projeto. As configurações podem ser personalizadas diretamente no *arquivo heatshrink_config.h*, fornecendo ajustes precisos para diferentes aplicações.

A biblioteca oferece duas formas de integração. O desenvolvedor pode incorporar diretamente as funções de codificação e decodificação no projeto ou usar a linha de comando autônoma programada para operações independentes. A operação segue um modelo simples baseado em uma máquina de estado. Os dados de entrada são fornecidos usando o método `sink`, processados e recuperados usando o método `poll` e, finalmente, a entrada é fechada com o método `finish`. Este ciclo pode ser repetido quantas vezes forem necessárias, permitindo a manipulação incremental dos dados.

Com sua eficiência e facilidade de uso, o Heatshrink se destaca como uma ferramenta dinâmica com baixo custo computacional, tornando-o uma escolha popular para sistemas embarcados que requerem compactação de dados sem comprometer os recursos limitados do dispositivo.

SOBRE O CÓDIGO

Para iniciar a implementação dos testes, estruturamos nossos algoritmos de forma a integrar a codificação do algoritmo BZ2 com as bibliotecas fornecidas pelo Heatshrink. Essa abordagem combinada visa aproveitar as vantagens de ambos os métodos, otimizando a compactação de dados em sistemas embarcados.

A **Tabela 03** apresenta o pseudocódigo que descreve a lógica usada para implementar o processo de compactação de dados. Essa estrutura detalha os estágios fundamentais do algoritmo, demonstrando como os dados são inicialmente processados pelo BZ2 e depois refinados usando o Heatshrink. Essa integração foi projetada para maximizar a eficiência, respeitando as limitações de memória e CPU típicas de dispositivos embarcados.

Tabela 03: Pseudocódigo de compactação

<pre> 01 Use <i>stdio.h</i> 02 Use <i>string.h</i> 03 Use <i>heatshrink_encoder</i> 04 Use <i>data_compress_app_layer.h</i> 05 Definir <i>max_compressed_data</i> 06 Criar <i>mass_memory_samples</i> 07 Criar <i>compressed_data</i> ← 0xff 08 Função compress_data (<i>p_from*</i>, <i>from_size</i>, <i>p_to*</i>, <i>p_to_size*</i>): </pre>

```

09 Inicializar p_from*
10 Inicialize from_size
11 Inicializar p_to*
12 Artimanha :( $x_i \leq x_n$ )
13      $p_{to}^* \leftarrow x_i$ 
14 Se  $p_{to}^* \geq max\_compressed\_data$ 
15 Imprimir (erro)
16 Intervalo
17 Atualização p_to_size*
18 Retorno: p_to_size*

19 Função compress_data_test():
20 Inicializar compressed_data
21     compressed_data compress_data(mass_memory_samples)←
22 Imprimir compressed_data

```

Fonte: Próprio autor

O processo busca detalhar a implementação do sistema de compressão de dados utilizando a biblioteca *Heatshrink*, com ênfase na simplicidade e eficiência. Inicialmente, são definidas constantes fundamentais, como o tamanho máximo permitido para os dados compactados, bem como estruturas básicas, incluindo o array *mass_memory_samples* para armazenar as amostras de memória que serão compactadas e o array *compressed_data* para conter os resultados compactados, inicialmente preenchidos com valores padrão.

A função principal, *compress_data*, é responsável por realizar a compactação iterativamente. Ele usa como entrada os dados originais, ponteiros para armazenar os resultados compactados e uma variável para registrar o tamanho final da compactação. O fluxo dessa função envolve a inicialização do codificador, o processamento dos dados originais em blocos e a extração contínua dos dados compactados até que todo o conteúdo tenha sido processado. Em caso de erros durante a execução, mensagens específicas são exibidas para garantir a transparência e facilitar a depuração. Quando a compactação é concluída, o tamanho final é atualizado e a função retorna o índice dos últimos dados processados.

Para validação e teste, a função *compress_data_test* usa a matriz *mass_memory_samples* como entrada, exibindo o índice retornado e os dados compactados em formato hexadecimal diretamente no console. Essa visualização facilita a análise e a verificação dos resultados obtidos.

A implementação também inclui funções auxiliares baseadas nos componentes da biblioteca, como *heatshrink_encoder* e *heatshrink_decoder*, que são usadas para compactação e descompactação de dados. Além disso, mecanismos robustos de tratamento de erros foram integrados para garantir que o processo seja confiável e seguro, mesmo em cenários adversos.

Com sua estrutura modular e bem definida, essa abordagem é altamente adequada para aplicações embarcadas que exigem eficiência e confiabilidade na compactação de dados, alinhando desempenho com simplicidade de implementação.

A função *decompress_data* **tabela 04** é responsável por descompactar os dados processados anteriormente, garantindo que o conteúdo compactado seja restaurado dentro dos limites especificados. Ele usa como entrada os dados compactados, o tamanho desses dados, um ponteiro para armazenar os dados descompactados, o tamanho final dos dados descompactados e o tamanho máximo permitido para a saída, garantindo que o processo permaneça dentro das restrições de memória.

Tabela 04: Pseudocódigo de descompactação

```

01 Use stdio.h
02 Use string.h
03 Use heatshrink_decoder
04 Use datacompressaplayer.h

05 Definir max_decompress_size
06 Função decompress_data(p_from*, from_size, p_to_size*, max_decompress_size):
07 Inicializar p_from*
08 Inicializar from_size
09 Inicializar p_to*
10 Inicializar p_to_size*
11 Artimanha :  $x_i \leq x_n$ 
12  $p\_to^* \leftarrow x_i$ 
13 Se  $p\_to^* \geq \text{max\_decompress\_size}$ 
14 Imprimir (erro)
15 Intervalo.
Atualização 16 p_to*
Atualização 17 p_to_size*
18 Retorno: 0

```

Fonte: Próprio autor

A função começa configurando o decodificador e inicializando as variáveis de controle, como a taxa de processamento e o tamanho pretendido para os dados descompactados. Para garantir um estado consistente, o decodificador é reiniciado antes de receber os dados compactados, que são gradualmente alimentados em partes menores. Essa abordagem de divisão em fragmentos reduz o risco de sobrecarga de memória, garantindo um processamento eficiente e compatível com as limitações do sistema.

Durante cada etapa de processamento, a função realiza verificações rigorosas para identificar possíveis erros. Se uma falha for detectada, mensagens de erro são exibidas, facilitando o diagnóstico e a correção. Ao mesmo tempo, o índice de processamento é atualizado continuamente, acompanhando o progresso do tratamento de dados. À medida que o fluxo progride, os dados descompactados são extraídos em cada iteração até que todo o conteúdo compactado tenha sido completamente processado.

Ao final do processo de descompactação, o tamanho final dos dados descompactados é registrado no ponteiro fornecido, garantindo que a memória alocada corresponda à quantidade de informações recuperadas. Por fim, a função retorna o valor 0, sinalizando que a operação foi concluída com êxito e que o conteúdo original foi restaurado sem erros ou interrupções.

4.3 IMPLEMENTAÇÃO

A biblioteca HEATSHRINK utiliza um conjunto de parâmetros configuráveis que impactam diretamente sua eficiência em termos de compactação e uso de recursos. Conforme descrito na documentação, o *parâmetro* `window_size` define o tamanho da janela, expresso como 2^W bytes, e tem uma influência direta no consumo de memória e na eficiência do algoritmo de compactação. Outro parâmetro relevante é o `lookahead_sz2`, que especifica o comprimento máximo de padrões repetidos, também definido como 2^W bytes. Além disso, o parâmetro `window_size - 1input_buffer_size` determina o tamanho do buffer de entrada no decodificador, influenciando a quantidade de dados processados em cada estágio.

Nos experimentos realizados, a configuração inicial para o parâmetro HEATSHRINK_STATIC_WINDO_BITS-(W) foi definida como 4, aumentando gradualmente para 10. Da mesma forma, o parâmetro HEATSHRINK_STATIC_LOOKAHEAD_BITS-(L) foi ajustado de 3 para $n-1$, levando em consideração o valor máximo de `window_size`. Cada bateria experimental incluiu 30 combinações diferentes de parâmetros. O cenário de teste foi desenhado para avaliar o desempenho em dois contextos de coleta de dados: o primeiro com 96 medições, equivalentes a 24 horas de leituras realizadas em intervalos de 15 minutos; o segundo com 144 medidas, correspondendo a leituras a cada 10 minutos no mesmo intervalo de 24 horas. Este último cenário foi denominado Tempo de Coleta (T_c).

O objetivo principal dos experimentos foi identificar a configuração que proporcionaria o melhor equilíbrio entre o tempo necessário para o envio dos dados e a eficiência da compressão, maximizando a relação custo-benefício. A estrutura matemática utilizada para modelar as configurações e analisar os resultados é formalizada na **equação 01**, que descreve a abordagem metodológica aplicada durante os testes.

$$T_{ij} = W_i L_j, (1)$$

onde:

- $i = \{4, 5, 6, \dots, 15\}$
- $j = \{3, \dots, i - 1\} \quad j \in Z.$
- T representa o teste

4.4 EXPERIÊNCIAS

Para a organização dos experimentos, foi adotado um planejamento de teste fatorial, representado por:

$$T_{30} \times Tc_2,$$

resultando em 60 testes diferentes, cobrindo todas as combinações possíveis de configurações.

A biblioteca HEATSHRINK foi usada para avaliar sua eficácia na compressão de quadros de memória de massa de um medidor trifásico. Cada quadro contém 52 bytes, dos quais 41 bytes são compostos de dados úteis após a remoção do cabeçalho DLMS, resultando em uma redução inicial de 21,42% no volume de dados transmitidos. Esses quadros incluem informações críticas, como o carimbo de data/hora UNIX, totalizadores de energia ativa e reativa sob diferentes condições de operação, bem como as tensões e correntes correspondentes às fases A, B e C.

A implementação da biblioteca HEATSHRINK, baseada no algoritmo LZSS, possibilitou ajustar parâmetros que equilibram a eficiência da compressão com o consumo de recursos do sistema. O valor máximo para o parâmetro `window_size` foi limitado a 10, devido às restrições de memória do dispositivo, o que equivale a um buffer de 2.048 bytes. O experimento foi projetado para testar várias combinações de parâmetros, avaliando seu impacto no uso de memória, taxa de compressão e tempo de execução.

O dispositivo incorporado usado tem limitações significativas de memória: 4.096 bytes disponíveis para STACK, 2.048 bytes alocados para HEAP e 4.102 bytes livres. Na memória FLASH, o dispositivo possui 24.316 bytes. Esses recursos foram alocados estrategicamente para garantir a estabilidade do sistema durante os testes. Os dados originais foram armazenados na memória FLASH (seção `.rodata`), os dados compactados foram alocados na RAM (seção `.noinit`) e o buffer necessário para a operação da biblioteca foi alocado localmente no STACK.

A análise de uso de memória, conduzida com a ajuda do Build Analyzer do STM32CubeIDE, revelou que a inclusão da biblioteca HEATSHRINK implicou um aumento no tamanho da seção `.text`. Antes da incorporação da biblioteca, esta seção ocupava 146.204 bytes. Após a inclusão, o uso aumentou para 148.912 bytes, representando um aumento de 2.708 bytes. Desse total, 494 bytes foram atribuídos às funções criadas especificamente para os testes, enquanto 2.214 bytes corresponderam à própria biblioteca. Além disso, identificou-se uma variação no consumo de memória à medida que o parâmetro `window_size` foi ajustado: para o valor 6, houve um aumento de 44 bytes; para 7, o aumento foi de 212 bytes; e para valores entre 8 e 10, o aumento variou entre 216 e 220 bytes.

A análise estática do STACK confirmou que a alocação de buffer local, quando configurada corretamente, não comprometeu a integridade da memória do sistema. Essa validação foi essencial para garantir que as configurações testadas fossem adequadas para dispositivos reais, garantindo a confiabilidade das operações sem risco de falha. Os resultados obtidos estão detalhados na **Tabela 05**, fornecendo uma visão abrangente do impacto das configurações propostas.

Tabela 05: Resultados dos testes realizados, onde representa o tamanho dos dados originais, representa o tamanho dos dados compactados, representa a taxa de compactação, t representa o tempo necessário para compactação e SSA representa Análise de Pilha Estática. $\alpha\beta\Delta$

Configuração	96 Amostras				144 amostras				SSA
	α_1	β_1	$\Delta_{\alpha_1, \beta_1}$	t	α_2	β_2	$\Delta_{\alpha_2, \beta_2}$	t	
	(bytes)	(bytes)	(%)	(MS)	(Bytes)	(Bytes)	(%)	(MS)	
HSWB04_HSLB03	3840	2405	0,3737	439	5760	3674	0,36215	664	104
HSWB05_HSLB03	3840	2258	0,4120	435	5760	3341	0,41997	661	136
HSWB05_HSLB04	3840	2311	0,3982	452	5760	3520	0,38889	687	136
HSWB06_HSLB03	3840	2120	0,4479	541	5760	3179	0,44809	809	200
HSWB06_HSLB04	3840	2038	0,4693	525	5760	3056	0,46944	786	200
HSWB06_HSLB05	3840	2045	0,4674	527	5760	3060	0,46875	789	200
HSWB07_HSLB03	3840	2144	0,4417	840	5760	3212	0,44236	1257	336
HSWB07_HSLB04	3840	2047	0,4669	821	5760	3070	0,46701	1228	336
HSWB07_HSLB05	3840	2061	0,4633	821	5760	3084	0,46458	1228	336
HSWB07_HSLB06	3840	2112	0,4500	830	5760	3160	0,45139	1243	336
HSWB08_HSLB03	3840	2175	0,4336	1400	5760	3268	0,43264	2103	592
HSWB08_HSLB04	3840	2064	0,4625	1383	5760	3102	0,46146	2074	592
HSWB08_HSLB05	3840	2087	0,4565	1392	5760	3127	0,45712	2087	592
HSWB08_HSLB06	3840	2140	0,4427	1396	5760	3206	0,44340	2092	592
HSWB08_HSLB07	3840	2223	0,4211	1502	5760	3328	0,42222	2247	592
HSWB09_HSLB03	3840	2207	0,4253	2532	5760	3320	0,42361	3799	1104
HSWB09_HSLB04	3840	2086	0,4568	2517	5760	3142	0,45451	3773	1104
HSWB09_HSLB05	3840	2115	0,4492	2547	5760	3173	0,44913	3811	1104
HSWB09_HSLB06	3840	2205	0,4258	2764	5760	3309	0,42552	4149	1104
HSWB09_HSLB07	3840	2243	0,4159	2767	5760	3365	0,41580	4153	1104
HSWB09_HSLB08	3840	2281	0,4060	2770	5760	3422	0,40590	4158	1104
HSWB10_HSLB03	3840	2244	0,4156	4712	5760	3380	0,41319	7030	2128
HSWB10_HSLB04	3840	2116	0,4490	4716	5760	3194	0,44549	7008	2128
HSWB10_HSLB05	3840	2178	0,4328	5254	5760	3290	0,42882	7931	2128
HSWB10_HSLB06	3840	2218	0,4224	5254	5760	3350	0,41840	7933	2128
HSWB10_HSLB07	3840	2258	0,4120	5257	5760	3409	0,40816	7936	2128
HSWB10_HSLB08	3840	2298	0,4016	5256	5760	3469	0,39774	7936	2128
HSWB10_HSLB09	3840	2338	0,3911	5265	5760	3528	0,38750	7948	2128
Média		2179	0,4325	2318		3276	0,43119	3483	957

Fonte: Próprio autor

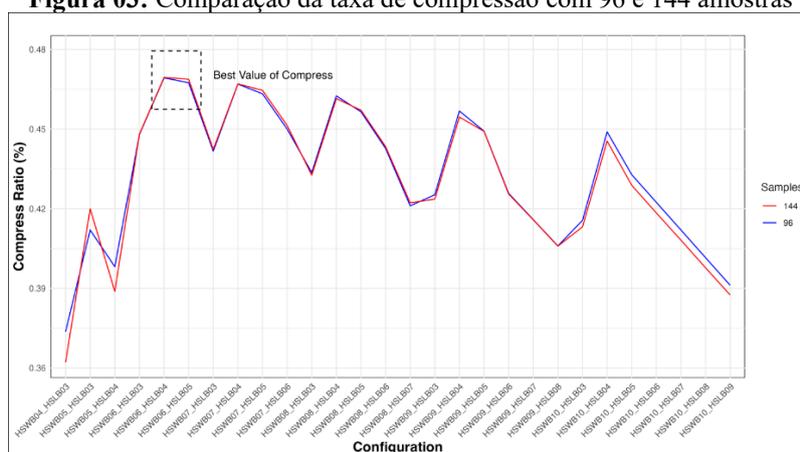
Os testes realizados com a biblioteca HEATSHRINK demonstraram claramente como as configurações de seus parâmetros influenciam a eficiência da compressão e o desempenho temporal do sistema. Durante as avaliações, a variação dos parâmetros `window_size` e `lookahead_sz2` revelou uma relação direta: as configurações que adotaram valores mais altos para esses parâmetros resultaram em maiores taxas de compressão, refletindo maior eficiência na compressão dos dados. No entanto, esse ganho foi acompanhado por um aumento significativo no tempo necessário para o processamento.

Esse comportamento destaca o trade-off característico entre eficiência de compactação e desempenho, uma questão central no contexto de sistemas embarcados. Esses dispositivos operam sob severas limitações de recursos, como memória e capacidade de processamento, o que requer um equilíbrio cuidadoso na escolha das configurações. Portanto, a otimização dos parâmetros da biblioteca HEATSHRINK deve considerar não apenas a maximização da compressão, mas também a necessidade de preservar o desempenho geral do sistema, garantindo que ele permaneça funcional e eficiente dentro das restrições impostas pelo hardware.

5 DISCUSSÃO DOS RESULTADOS

Dentre os resultados obtidos, destacou-se o desempenho de configurações intermediárias, como HSWB0_HSLB04 e HSWB06_HSLB05, que demonstraram uma taxa de compressão próxima a 46,93% ao processar 96 amostras, com tempos de execução gerenciáveis de aproximadamente 525 ms. Esses achados mostram que é possível alcançar um equilíbrio adequado entre eficiência de compressão e desempenho temporal, tornando essas configurações viáveis para sistemas embarcados que precisam otimizar o uso de recursos sem comprometer significativamente o tempo de resposta. Veja a **figura 03**.

Figura 03: Comparação da taxa de compressão com 96 e 144 amostras

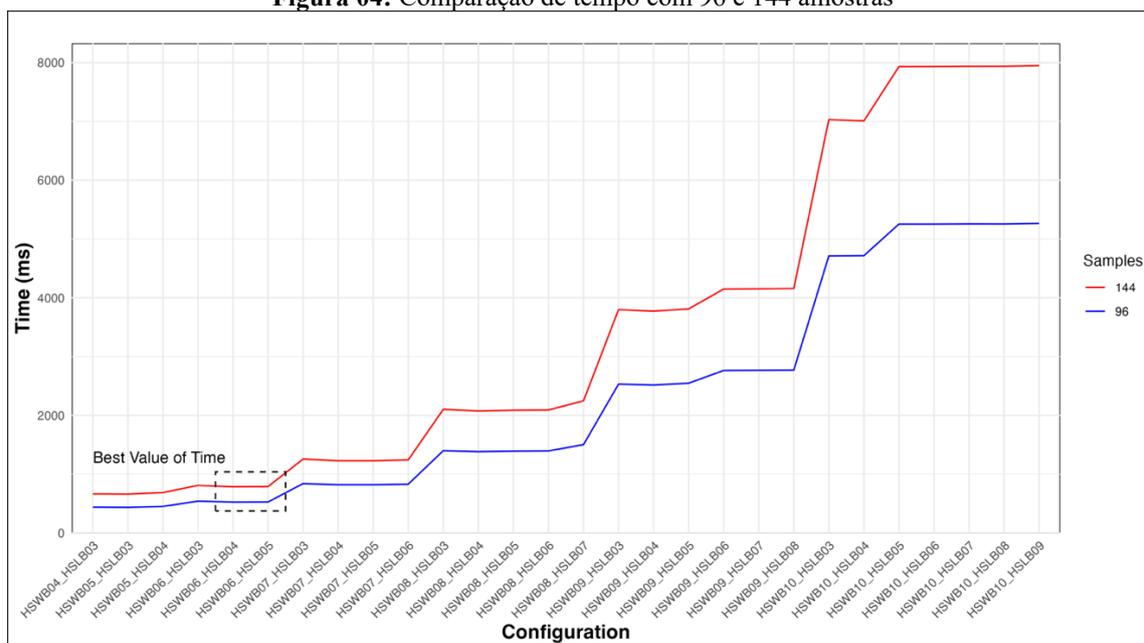


Fonte: Próprio autor

Em contrapartida, configurações mais extremas, como HSWB10_HSLB08 e HSWB10_HSLB09, apresentaram desempenho inferior, com taxas de compressão abaixo de 40%. Além disso, os tempos de processamento para essas configurações ultrapassaram 7.900 ms ao lidar com 144 amostras, valor que inviabiliza o uso em aplicações que exigem baixa latência ou respostas em tempo real. Esses resultados enfatizam a importância de balancear cuidadosamente os parâmetros para atender às demandas específicas de cada aplicação, especialmente em dispositivos embarcados onde os limites de hardware impõem restrições severas à escolha de configurações extremas.

Os testes também revelaram que o número de amostras processadas afeta diretamente o desempenho do tempo, enquanto as taxas de compressão permanecem praticamente inalteradas. Ao comparar cenários com 96 e 144 amostras, observou-se um aumento proporcional no tempo de execução em função do volume de dados processados, ver **figura 04**. Esses resultados indicam que a biblioteca HEATSHRINK é escalável, permitindo que ela seja usada com diferentes volumes de dados sem comprometer a eficiência da compactação. No entanto, em nosso aplicativo e caso de uso, o tempo não é um fator limitante, pois pode levar até minutos, pois estaremos compactando apenas uma vez por dia.

Figura 04: Comparação de tempo com 96 e 144 amostras

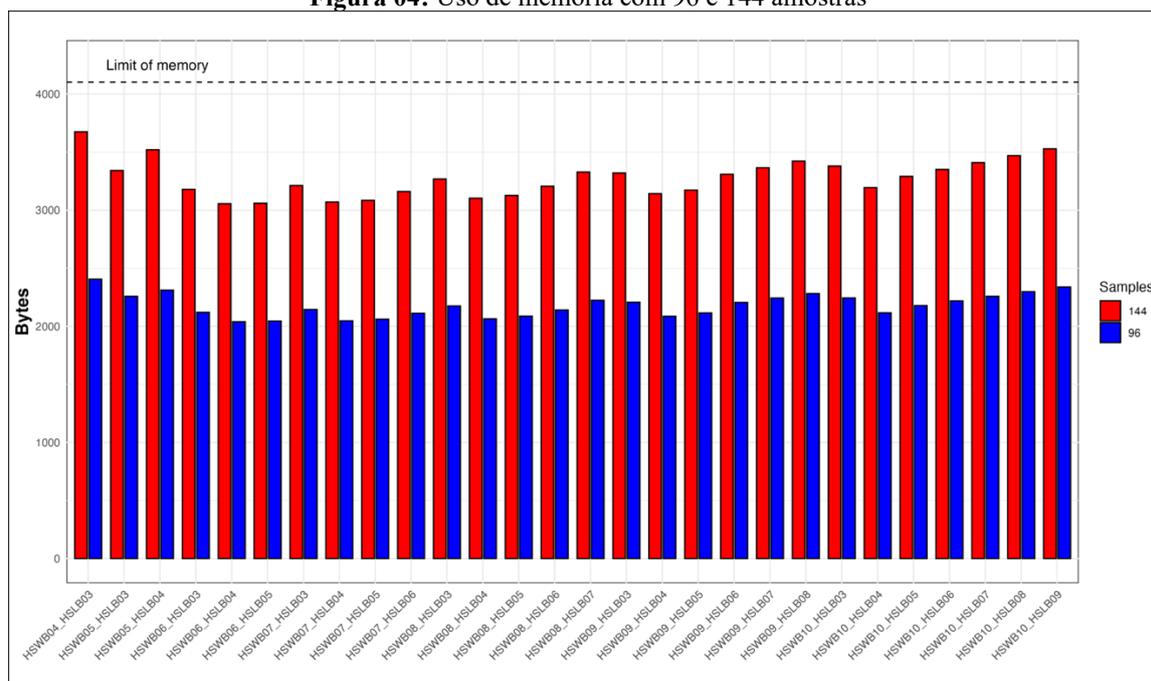


Fonte: Próprio autor

No entanto, o impacto no tempo de execução destaca a necessidade de ajustes precisos nas configurações, especialmente em aplicações que demandam tempos de resposta reduzidos. Essa análise reforça que, embora a escalabilidade do HEATSHRINK seja uma vantagem, sua aplicação em sistemas embarcados deve ser cuidadosamente projetada para garantir que o desempenho atenda aos requisitos operacionais específicos, mantendo a estabilidade e a eficiência do sistema.

A análise do impacto no uso de memória revelou resultados significativos, especialmente em configurações mais robustas como HSWB10_HSLB09, que demandavam até 2.128 bytes de memória, veja a **figura 05**. Esse consumo representa um desafio considerável para dispositivos embarcados, que geralmente são caracterizados por recursos limitados. Esses achados enfatizam a importância de fazer ajustes cuidadosos nos parâmetros, considerando não apenas a eficiência de compressão e o tempo de execução, mas também o consumo de memória, fator crítico para garantir a estabilidade e a funcionalidade dos sistemas embarcados em cenários operacionais restritos.

Figura 04: Uso de memória com 96 e 144 amostras



Fonte: Próprio autor

Com base nos resultados obtidos, as recomendações de configuração devem ser adaptadas ao contexto específico da aplicação. Para dispositivos que enfrentam restrições estritas de tempo e recursos, configurações intermediárias, como HSWB06_HSLB04 são mais apropriadas. Essas configurações oferecem um equilíbrio vantajoso, com taxas de compactação eficientes e tempos de execução gerenciáveis. Neste caso, a compressão provou ser altamente eficaz: reduzindo inicialmente o cabeçalho em 21,15% e aplicando uma taxa de compressão adicional de 46,93%, foi alcançada uma redução final de 68,08%, deixando o arquivo original com apenas 31,92% de seu tamanho inicial.

Em aplicações em que a eficiência máxima de compactação é uma prioridade, configurações com valores mais altos de `window_size` e `lookahead` podem ser exploradas, desde que o consumo de memória permaneça dentro dos limites aceitáveis do hardware. Dessa forma, é possível otimizar o desempenho da biblioteca HEATSHRINK para atender tanto às demandas de compressão quanto às restrições de cada ambiente operacional, garantindo a viabilidade e eficiência da aplicação em sistemas embarcados.

Os resultados obtidos enfatizam a importância de uma análise detalhada e criteriosa na definição dos parâmetros da biblioteca HEATSHRINK para sistemas embarcados. O ajuste adequado desses parâmetros é fundamental para alcançar o equilíbrio ideal entre eficiência de compressão, tempo de processamento e consumo de memória. Essa abordagem garante que os aplicativos atendam às suas demandas específicas sem comprometer a viabilidade técnica ou a eficiência operacional, especialmente em dispositivos com severas restrições de recursos.

Os resultados também destacam a necessidade de uma parametrização equilibrada da biblioteca. Em cenários onde a prioridade é a eficiência de compressão, valores mais altos para

window_size e lookahead_sz2 se mostraram vantajosos, proporcionando maiores taxas de compressão, mesmo com o aumento do consumo de memória e do tempo de processamento. Por outro lado, em aplicações que exigem baixa latência, como sistemas de tempo real, configurações intermediárias desses parâmetros foram mais adequadas. Essas configurações alcançaram taxas de compactação satisfatórias, com tempos de execução mais curtos e menor impacto no uso de RAM, tornando-as ideais para contextos onde o desempenho rápido é essencial.

O experimento mostrou que a aplicação do HEATSHRINK em sistemas embarcados requer uma compreensão completa das necessidades específicas da aplicação, bem como das limitações impostas pelo hardware. A estratégia de alocar os dados originais na memória FLASH e os dados compactados na RAM, aliada ao controle rigoroso sobre os parâmetros da biblioteca, possibilitou explorar sua eficiência em diferentes cenários. Essa abordagem não apenas otimizou o armazenamento e a transmissão de dados, mas também garantiu a estabilidade e o desempenho do sistema, mesmo em ambientes com recursos limitados.

Esses resultados fornecem uma base para a implementação de soluções de compactação em dispositivos embarcados. Ao equilibrar a configuração de parâmetros com as demandas específicas de cada aplicação, é possível maximizar os benefícios da compressão, garantindo a funcionalidade e eficiência do sistema sem comprometer sua estabilidade. Essas descobertas podem servir como uma referência valiosa para o desenvolvimento e otimização de sistemas embarcados que exigem soluções de compactação de dados robustas e escaláveis.

6 CONCLUSÃO

Este estudo investigou a aplicação de técnicas de compressão de dados em sistemas embarcados, com foco no uso da biblioteca HEATSHRINK para otimizar o desempenho de dispositivos baseados no microcontrolador STM32WBA. O principal objetivo era desenvolver uma solução eficiente que atendesse às restrições de recursos desse hardware, reduzindo o volume de dados transmitidos sem comprometer a integridade das informações ou a estabilidade do sistema.

Os testes realizados exploraram diferentes configurações de parâmetros HEATSHRINK, com ajustes nos valores de window_size e lookahead_sz2 para equilibrar a taxa de compressão, o tempo de processamento e o consumo de memória. As configurações intermediárias, como HSWB06_HSLB04 e HSWB06_HSLB05, alcançaram uma taxa de compactação de aproximadamente 46,93% ao processar 96 amostras, com tempos de execução em torno de 525 ms e um impacto controlado no uso da memória, limitado a 200 bytes de RAM. Considerando a redução adicional proporcionada pela remoção do cabeçalho, a eficiência total atingiu 68,08%. Esses resultados demonstram que tais configurações são adequadas para dispositivos embarcados que requerem uma combinação de boa eficiência de compactação e tempos de execução gerenciáveis.



Em contraste, configurações mais extremas, como HSWB10_HSLB08 e HSWB10_HSLB09, apresentaram taxas de compressão inferiores a 40%. Além disso, os tempos de processamento excederam 7.900 ms para 144 amostras, enquanto o consumo de memória atingiu 2.128 bytes. Esses resultados enfatizam a importância de uma análise cuidadosa na escolha de parâmetros, particularmente em sistemas com restrições severas de memória e latência.

A análise também revelou que o volume de amostras processadas afeta diretamente o desempenho temporal, mesmo que as taxas de compressão permaneçam estáveis. Esse comportamento indica que a biblioteca HEATSHRINK é escalável, mas requer ajustes precisos para atender às demandas específicas de cada aplicativo. Para cenários que exigem baixa latência, as configurações intermediárias provaram ser a escolha ideal, oferecendo um equilíbrio eficiente entre compactação e desempenho.

A viabilidade técnica do experimento foi garantida pela alocação estratégica dos recursos do STM32WBA. Com memória RAM limitada a 4.096 bytes, os dados originais foram armazenados na memória FLASH, enquanto os dados compactados foram alocados na RAM. O buffer necessário para a operação da biblioteca foi gerenciado localmente no STACK, garantindo que o sistema permanecesse estável e livre de estouros de memória durante os testes. A análise realizada com o STM32CubeIDE Build Analyzer confirmou que essas estratégias de alocação possibilitaram integrar a biblioteca HEATSHRINK de forma eficaz, sem comprometer as operações gerais do dispositivo. Esses resultados fornecem uma base para a aplicação da compressão de dados em sistemas embarcados, equilibrando eficiência e estabilidade em diferentes cenários operacionais.

Os resultados deste estudo confirmam que a compressão de dados, além de reduzir significativamente o volume transmitido, representa uma solução eficaz para prolongar a vida útil de dispositivos alimentados por bateria, reduzir o consumo de energia e otimizar a largura de banda em redes IoT. Considerando as limitações específicas do STM32WBA, este trabalho propôs configurações otimizadas que maximizam os recursos do microcontrolador sem comprometer sua capacidade de armazenamento ou RAM durante os processos de compressão e transmissão de dados.

Como sua principal contribuição, o estudo posiciona a biblioteca HEATSHRINK, combinada com algoritmos de compressão, como uma solução viável para compressão em dispositivos embarcados, desde que configurada de forma precisa e adequada para atender às restrições de hardware e demandas específicas de cada aplicação. Os resultados apresentados fornecem uma base para o desenvolvimento de implementações futuras, ao mesmo tempo em que sugerem a incorporação de técnicas híbridas e o uso de aprendizado de máquina para melhorar ainda mais o desempenho e a eficiência de dispositivos com recursos limitados.

Os avanços obtidos neste estudo oferecem uma contribuição significativa para a Wasion, fortalecendo sua posição como líder em soluções de medição inteligente e eficiência energética. A



análise detalhada do impacto da compactação de dados em medidores embarcados, especialmente os STM32WBA, fornece diretrizes valiosas para otimizar o desempenho do dispositivo sem comprometer a integridade das informações ou a estabilidade operacional. Ao demonstrar a viabilidade do HEATSHRINK como uma solução eficiente para a redução do volume de dados transmitidos, este estudo possibilita a implementação de sistemas mais ágeis e energeticamente eficientes, prolongando a vida útil dos medidores e otimizando o consumo de recursos computacionais. Além disso, as configurações propostas permitem que os dispositivos da Wasion sejam melhor adaptados a redes de comunicação de baixa largura de banda, como LoRaWAN e NB-IoT, ampliando o alcance e a competitividade da empresa no mercado global de medição inteligente. Com base nos resultados apresentados, futuras integrações com técnicas avançadas, como compressão híbrida e aprendizado de máquina, poderão melhorar ainda mais a eficiência dos sistemas embarcados da empresa, consolidando sua capacidade de inovação e atendendo às crescentes demandas por soluções sustentáveis e escaláveis no setor de energia.

As recomendações para trabalhos futuros incluem a exploração de técnicas de compressão híbrida que combinam algoritmos clássicos, como os usados neste estudo, com métodos modernos baseados em aprendizado de máquina. Abordagens como redes neurais compactadas e modelos preditivos adaptativos podem ser investigadas para otimizar ainda mais a eficiência da compactação, especialmente em dispositivos com severas restrições de recursos. Essas soluções têm o potencial de identificar padrões em tempo real e adaptar o processo de compressão às características dinâmicas dos dados coletados, proporcionando maior flexibilidade e eficiência. Também podemos explorar uma comparação prática com equipamentos em campo, analisando a eficiência antes e depois da implementação do algoritmo.

Além disso, recomendamos investigar estratégias avançadas de gerenciamento de memória para dispositivos incorporados com recursos ainda mais limitados do que o STM32WBA, bem como novas configurações de amostra e tempos de coleta. Métodos como alocação dinâmica de memória e buffers segmentados podem ser testados para oferecer maior flexibilidade no gerenciamento de recursos durante a compactação e o envio de dados. Essa linha de pesquisa pode incluir o desenvolvimento de métricas detalhadas para avaliar o impacto da compressão no desempenho geral do sistema, fornecendo uma visão mais abrangente dos benefícios e limitações das soluções propostas.

Outra área de interesse seria a integração de técnicas de compressão com sistemas de monitoramento e análise preditiva, usados em aplicações como manutenção preditiva e detecção de anomalias. A combinação de compactação eficiente com modelos preditivos não apenas reduziria o volume de dados transmitidos, mas também permitiria antecipar falhas e identificar comportamentos anômalos com maior precisão. Essa abordagem seria particularmente útil em redes IoT industriais, onde a confiabilidade e a eficiência são cruciais.



Por fim, a validação em campo das configurações otimizadas propostas neste estudo é uma etapa fundamental na consolidação dos resultados teóricos e experimentais. Estudos futuros poderão realizar testes em larga escala, simulando diferentes condições de operação e avaliando a robustez, eficiência e confiabilidade das soluções propostas em cenários reais. Esse tipo de validação prática tem o potencial de ampliar as aplicações das técnicas de compressão em setores como energia, saúde e transporte, ampliando o impacto e a relevância das contribuições deste trabalho.

AGRADECIMENTOS

Agradecemos ao Grupo Wasion, que acreditou no Evolução Instituto e em seus parceiros, NEPEN e IFRO, para realizar essa pesquisa e prototipar a solução encontrada. Esta pesquisa foi financiada pela Lei nº 8.387/1991 (Zona Franca de Manaus) enquadrada nos artigos 21 e 22 do Decreto nº 10.521, de 15/10/2020



REFERÊNCIAS

- Burrows, M., & Wheeler, D. J. (1994). *A block-sorting lossless data compression algorithm* (Report No. 124). Digital Equipment Corporation. <https://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf>
- Cleary, J. G., & Witten, I. H. (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications, 32*(4), 396–402.
- Costanzi, J. E., & Guembarovski, R. H. (2023). Aplicação de técnicas de mineração de dados para aprimoramento da gestão do sistema elétrico de baixa tensão. In *Seminário Nacional de Distribuição de Energia Elétrica (SENDI)*.
- Júnior, J. A. de O., Camargo, E. T. de, & Oyamada, M. S. (2023). Data compression in LoRa networks: A compromise between performance and energy consumption. *Journal of Internet Services and Applications, 14*(1), 1–15.
- Júnior, J. A. de O., & Oyamada, M. S. (2023). Avaliando o impacto da compressão de dados no desempenho e energia em redes LoRa. In *Seminário Nacional de Distribuição de Energia Elétrica (SENDI)*.
- Ketshabetswe, K. L., Zungeru, A. M., Mtengi, B., Lebekwe, C. K., & Prabakaran, S. R. S. (2021). Data compression algorithms for wireless sensor networks: A review and comparison. *IEEE Access, 9*, 136872–136882. <https://doi.org/10.1109/ACCESS.2021.3114358>
- Knuth, D. E. (1998). *The art of computer programming: Sorting and searching* (Vol. 3). Addison-Wesley.
- Kodituwakku, S. R., & Amarasinghe, U. S. (2010). Comparison of lossless data compression algorithms for text data. *Indian Journal of Computer Science and Engineering, 1*(4), 416–426.
- Lee, J., Gong, Q., Choi, J., Banerjee, T., Klasky, S., Ranka, S., & Rangarajan, A. (2022). Error-bounded learned scientific data compression with preservation of derived quantities. *Applied Sciences, 12*(13), Article 6718. <https://doi.org/10.3390/app12136718>
- Lee, W.-H., Chen, M.-H., Lee, C.-Y., & Li, Y.-L. (2016). *Lossless compression of data tables in mobile devices using co-clustering*.
- Malandrino, F., Di Giacomo, G., Karamzade, A., Levorato, M., & Chiasserini, C. F. (2024). Tuning DNN model compression to resource and data availability in cooperative training. *IEEE/ACM Transactions on Networking, 32*(2), 1600–1615. <https://doi.org/10.1109/TNET.2023.3325299>
- Meffe, A., Prieto, M. A. P., Garcez Neto, A. de F., & Teodoro Júnior, J. R. (2023). Solução de baixo custo para leitura e gerenciamento remoto de unidades consumidoras dispersas com tecnologia LoRaWAN. In *Seminário Nacional de Distribuição de Energia Elétrica (SENDI)*.
- Moon, A., Kim, J., Zang, J., & Son, S. W. (2018). Evaluating fidelity of lossy compression on spatiotemporal data from an IoT. *Computers and Electronics in Agriculture, 154*, 304–313. <https://doi.org/10.1016/j.compag.2018.09.015>
- Python Software Foundation. (2024). *bz2 compression compatible with bzip2*. <https://docs.python.org/3/library/bz2.html>



Qin, J., Lu, Y., & Zhong, Y. (2020). Block-split array coding algorithm for long-stream data compression. **Journal of Sensors, 2020**, Article 1–22. <https://doi.org/10.1155/2020/8839888>

Ribera Neto, D., & Leite, M. A. R. (2023). Aplicação de inteligência artificial no processo de análise dos dados de medição com foco na recuperação de energia. In **Seminário Nacional de Distribuição de Energia Elétrica (SENDI)**.

Sandoval, S., Vargas, A., Ortega, C., & Vidal, Y. (2020). Three-way unsupervised data mining for power system applications based on tensor decomposition. **International Journal of Electrical Power & Energy Systems, 123**, Article 106241. <https://doi.org/10.1016/j.ijepes.2020.106241>

Santos, W. G. V. dos, Queiroz Costa, J. V. de, Salustiano de Oliveira, R., & Cabral, M. de O. (2023). Medidores inteligentes com tecnologia NB-IoT: Análise em ambientes urbanos e regiões rurais com o uso de blindagem. In **Seminário Nacional de Distribuição de Energia Elétrica (SENDI)**.

Seward, J. (2024). **bzip2 and libbzip2: A program and library for data compression**. <https://www.sourceware.org/bzip2/>

Zhang, F., Liu, M., Zhang, Z., He, J., & Gao, W. (2023). Real-time synchrophasor data compression technique with phasor interpolation and extrapolation. **Journal of Modern Power Systems and Clean Energy, 11*(3)*, 803–815. <https://doi.org/10.35833/MPCE.2021.000567>

Zhang, Z. (2023). The improvement of the discrete wavelet transforms. **Mathematics, 11*(1770)*. <https://doi.org/10.3390/math11081770>