Desempenho Energético no FreeRTOS com Paralelismo em Relação a Sistema Sequencial em um Protótipo de Hidrômetro LoRa

Robert A. Cabral[1], **Paulo A. C.**[2], **Michel S. Bonfim**[2], **Emanuel F. Coutinho**[2]

[1]Instituto de Pesquisas Eldorado
Av. Alan Turing, 275 – Cidade Universitaria – Barão Geraldo
Campinas – SP – Brasil – CEP 13083-898

[2]Programa de Pós Graduação em Computação – Universidade Federal do Ceará
Caixa Postal 63902.580 – Quixadá – CE – Brasil

robertcabral.qx@gmail.com, {pauloaguilar, michelsb, emanuel.coutinho}@ufc.br

Abstract. *This article analyzes the energy performance of an IoT system for water monitoring. The study compared two implementations: one based on the FreeRTOS real-time operating system and the other using the Arduino framework. The tests were conducted in a controlled environment, ensuring uniform conditions for both approaches and minimizing variations arising from the testing process. The results showed that FreeRTOS offers significant advantages in terms of energy efficiency due to its multitasking management, while Arduino stood out for its simplicity of implementation. The study highlights the impact of software choices on the autonomy of IoT-embedded systems, focusing on the importance of efficient energy management to ensure the sustainability of IoT systems in remote monitoring applications.*

Resumo. *Este artigo apresenta uma análise do desempenho energético de um sistema IoT para monitoramento hídrico. O estudo comparou duas implementações distintas: uma baseada no sistema operacional de tempo real FreeRTOS e outra utilizando o framework Arduino. Os testes foram realizados em um ambiente controlado, assegurando condições uniformes para ambas as abordagens, minimizando as variações decorrentes do processo de teste. Os resultados demonstraram que o FreeRTOS oferece vantagens significativas em termos de eficiência energética devido ao seu gerenciamento multitarefa, enquanto o Arduino se destacou pela simplicidade de implementação. O estudo ressalta o impacto das escolhas de software na autonomia de sistemas embarcados IoT, com foco na importância de uma gestão energética eficiente para garantir a sustentabilidade de sistemas IoT em aplicações de monitoramento remoto.*

1. Introdução

A gestão sustentável dos recursos hídricos é um dos maiores desafios globais, especialmente em países como o Brasil, onde o desperdício e as perdas na distribuição de água atingem níveis alarmantes. De acordo com o Estudo de Perdas de Água (2024), cerca de 37,78% da água tratada é perdida durante a distribuição, enquanto 32,62% não é faturada,

revelando ineficiências significativas no sistema de abastecimento. Além disso, os efeitos das mudanças climáticas e a variabilidade hidrológica tornam o monitoramento contínuo uma necessidade estratégica para o planejamento e a segurança hídrica [ANA 2024]. O crescimento populacional e a urbanização também aumentam a pressão sobre os recursos hídricos, agravando ainda mais os desafios enfrentados pelos gestores hídricos.

Nesse contexto, tecnologias baseadas em Internet das Coisas (IoT) têm se destacado como soluções eficazes para a automação e o controle das redes hídricas. O protocolo LoRa (Long Range), amplamente utilizado em aplicações de monitoramento remoto, se destaca pela sua capacidade de comunicação de longo alcance com baixo consumo de energia, tornando-se especialmente útil em cenários onde a conectividade precisa ser mantida em áreas de difícil acesso [Felinto Filho 2023]. A eficiência energética do LoRa permite que sistemas IoT, como hidrômetros inteligentes, operem de forma contínua e autossustentável, mesmo em locais remotos e sem acesso constante a fontes de energia.

Estudos recentes reforçam a importância dessas tecnologias. O trabalho de [Kurdija et al. 2024] explorou a implementação da tecnologia LoRaWAN para leitura remota de hidrômetros em áreas urbanas, demonstrando sua viabilidade em ambientes desafiadores, embora sem detalhar a plataforma embarcada utilizada. Já [Gorawski et al. 2024] analisaram o consumo energético de medidores inteligentes equipados com LoRa, evidenciando que a frequência de transmissão de dados influencia significativamente a vida útil das baterias. Por sua vez, [Migabo et al. 2021] propuseram um hidrômetro baseado em LoRaWAN com ênfase na eficiência energética, utilizando a plataforma EFM32, mas sem considerar o impacto do *software* no consumo energético. Da mesma forma, [Ayatullah et al. 2023] desenvolveram um sistema de hidrômetro inteligente com comunicação LoRa, mas sem apresentar uma análise detalhada do consumo energético ou do *software* utilizado.

Apesar da eficiência do LoRa na comunicação, esses sistemas, por estarem distribuídos em diferentes pontos e frequentemente sem fontes de energia disponíveis, demandam uma gestão rigorosa de energia. A capacidade de operar por longos períodos com baterias limitadas depende da implementação de estratégias de gestão eficiente de energia [González et al. 2020]. Segundo [Mór et al. 2010], o aumento da eficiência energética é importante em todo contexto computacional, independente do *hardware* adotado. Ou seja, para todas arquiteturas e implementações físicas de máquinas paralelas, as quais podem apresentar diferentes consumos de potência, a eficiência energética é um fator crucial a ser analisado. Portanto, além de considerar a comunicação eficiente via LoRa, o desenvolvimento de sistemas IoT para monitoramento de água, como o hidrômetro inteligente, deve também focar em estratégias eficazes para o gerenciamento do consumo energético.

A escolha do *software* utilizado em sistemas embarcados desempenha um papel fundamental no consumo energético. O Arduino, um *framework* amplamente utilizado, é conhecido por sua simplicidade e facilidade de desenvolvimento [PlatformIO 2025]. No entanto, apesar de ser uma solução prática, o Arduino não oferece o mesmo controle detalhado sobre o gerenciamento de energia que sistemas mais avançados, como o FreeRTOS, que permite um gerenciamento eficiente de tarefas e modos de operação [FreeRTOS 2024].

Este trabalho apresenta uma análise do desempenho energético de um protótipo de

hidrômetro inteligente com comunicação LoRa, utilizando duas abordagens de *software*: o Arduino e o FreeRTOS. O objetivo do estudo é avaliar o impacto dessas arquiteturas no consumo de energia, destacando como a gestão energética pode afetar a eficiência de aplicações IoT. Diferentemente dos trabalhos anteriores, esta pesquisa foca tanto no consumo energético quanto na influência do *software* embarcado no desempenho dos sistemas de monitoramento hídrico.

2. Fundamentação teórica

Esta seção apresenta os conceitos fundamentais relacionados ao trabalho proposto, abordando de forma concisa a Internet das Coisas (IoT), Sistemas Operacionais em Tempo Real (RTOS) e Eficiência Energética.

2.1. Internet das Coisas

A Internet das Coisas pode ser definida como a interconexão de objetos físicos com a Internet, permitindo a coleta, o processamento e o compartilhamento de dados [Gubbi et al. 2013]. Essa tecnologia integra sensores, dispositivos e comunicação para criar redes inteligentes, sendo amplamente aplicada em setores como saúde, transporte e cidades inteligentes [Ashton 2009]. No contexto da gestão hídrica, sistemas IoT permitem detectar vazamentos e desperdícios com mais eficiência, contribuindo para o uso sustentável da água.

2.2. Sistemas Operacionais em Tempo Real (RTOS)

Sistemas operacionais são fundamentais para gerenciar os recursos de *hardware* e *software* dos dispositivos [Tanenbaum and Bos 2022]. Em sistemas embarcados, os RTOS garantem a execução determinística de tarefas, priorizando a previsibilidade. O FreeRTOS, por exemplo, destaca-se por sua leveza, suporte a múltiplas plataformas e integração facilitada com dispositivos IoT [Cooling 2020].

O FreeRTOS também possibilita a criação de tarefas que executam de forma independente e paralela, desempenhando um papel fundamental na melhoria da eficiência ao utilizar melhor os recursos computacionais e aumentar o desempenho das aplicações. Além disso, permite um aproveitamento máximo da CPU, reduzindo sua ociosidade [Mór et al. 2010].

2.3. Eficiência Energética

Eficiência energética refere-se à redução do consumo de energia sem comprometer a funcionalidade [MME and EPE 2007]. Em dispositivos IoT, onde a maioria dos sistemas opera com alimentação por bateria, a gestão eficiente da energia é um fator crítico para aumentar a autonomia e reduzir a necessidade de manutenção. Para isso, diversas estratégias são empregadas, como o uso de microcontroladores com modos de baixo consumo, sensores de alta eficiência e protocolos de comunicação otimizados.

Microcontroladores modernos, como os da família STM32F0, oferecem modos de economia de energia, como *sleep* e *stop*, que permitem reduzir significativamente o consumo durante períodos de inatividade [STMicroelectronics 2020]. Além disso, a implementação de técnicas como ajuste dinâmico da frequência de operação, uso de reguladores de tensão de alta eficiência e otimização do código para reduzir ciclos de processamento também contribuem para minimizar o gasto energético.

3. Metodologia

O sistema foi projetado para realizar a coleta e transmissão de dados de consumo hídrico de um hidrômetro com suporte a sensor pulsado, utilizando um protocolo de comunicação de baixa potência, o LoRaWAN. Esse sistema será utilizado como base para a análise de desempenho energético, sendo implementado em duas abordagens diferentes: uma utilizando o sistema operacional em tempo real FreeRTOS e outra com o *framework* do Arduino. A análise comparará o impacto de cada abordagem no consumo energético do dispositivo, visando identificar qual solução oferece maior eficiência.

O diagrama na Figura 1 apresenta o funcionamento geral do sistema, destacando os fluxos principais de operação e o gerenciamento por interrupções priorizadas.

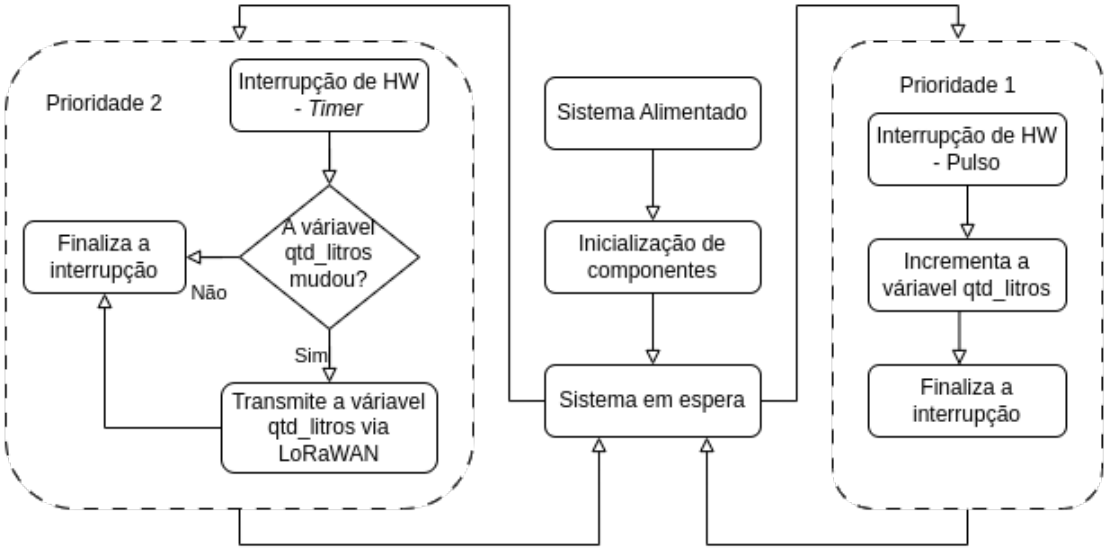

Figure 1. Diagrama do prótotipo do sistema.

As subseções a seguir apresentam os componentes essenciais para compreender a metodologia empregada neste trabalho.

3.1. Protótipo do sistema

O protótipo do sistema foi desenvolvido utilizando como microcontrolador principal o STM32F030C8T6. A placa foi projetada para ser compacta, eficiente e adequada para aplicações IoT de baixo consumo energético. Os circuitos embutidos na placa incluem:

- Circuito de carregamento de bateria;
- Circuito regulador de tensão da bateria;
- Circuito regulador de tensão para o sensor ultrassônico pulsado;
- Circuito de alimentação do microcontrolador;
- Circuito de conexão com o módulo RFM95W.

A Figura 2 apresenta o protótipo do sistema, mostrando o *hardware* desenvolvido, além da bateria e do painel solar instalado em um hidrômetro ultrassônico com sensor pulsado.

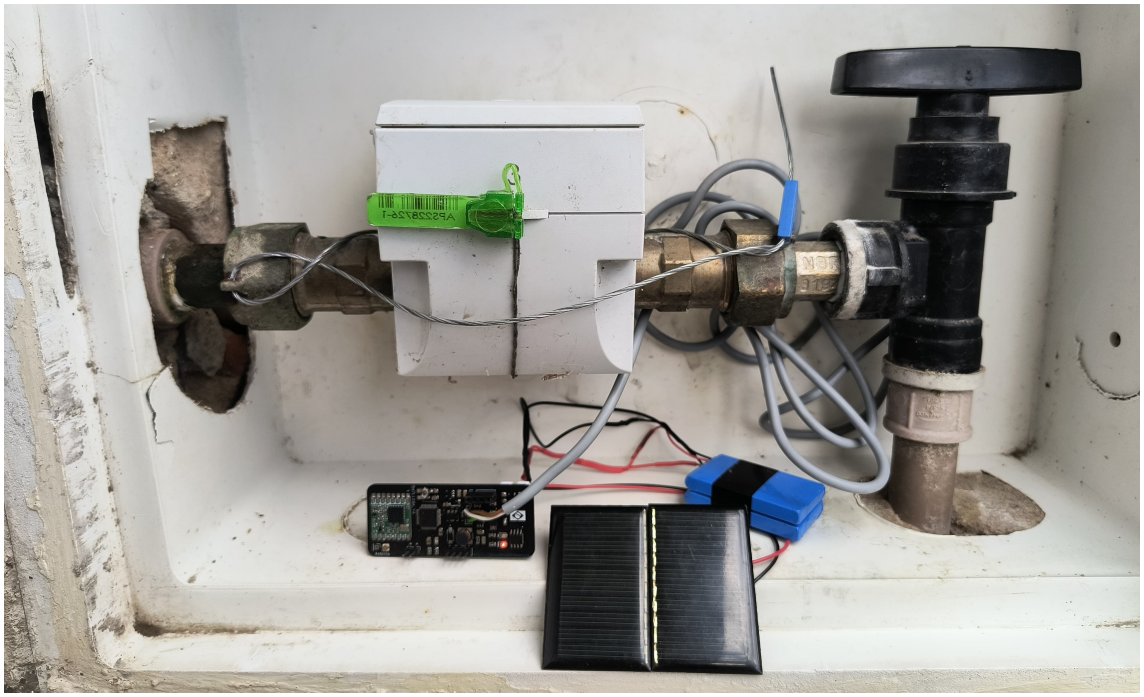

Figure 2. Protótipo do sistema.

3.1.1. Estados do Sistema

Conforme ilustrado na Figura 1, após ser alimentado, o sistema realiza a inicialização dos componentes de *hardware* e *software* necessários para seu funcionamento, incluindo o microcontrolador, os módulos de comunicação LoRaWAN, as interrupções e as variáveis associadas ao controle do fluxo de dados. Após a inicialização, o sistema entra no estado de espera.

Nas implementações, foram configuradas duas interrupções de *hardware*, cada uma com uma prioridade distinta. A interrupção de maior prioridade é responsável por monitorar o pulso enviado pelo hidrômetro, utilizando um pino GPIO do microcontrolador. Quando essa interrupção é acionada, a variável que armazena a quantidade de litros consumidos é incrementada, indicando que o hidrômetro registrou o consumo de mais um litro de água.

A segunda interrupção, de menor prioridade, é baseada em um temporizador (*timer*). Ela é acionada periodicamente a cada um minuto e verifica se houve alteração na quantidade de litros consumidos desde a última verificação. Caso a variável tenha sido alterada, o sistema transmite a nova quantidade de litros para o servidor utilizando a comunicação LoRaWAN. Vale ressaltar que, para fins de teste e coleta de dados em um período mais curto, o envio de dados foi configurado para ocorrer a cada um minuto. No entanto, em um ambiente real, esse envio normalmente seria configurado para uma frequência menor, de acordo com as necessidades da aplicação.

Nas duas implementações, utilizando tanto o FreeRTOS quanto o *framework* do Arduino, as interrupções foram configuradas para realizar os mesmos procedimentos, garantindo consistência no comportamento do sistema para a análise comparativa.

3.2. Implementação nos Ambientes FreeRTOS e Arduino

A análise de desempenho foi realizada utilizando o mesmo *hardware*, mas com diferentes implementações de *software*: uma baseada no FreeRTOS e outra no *framework* do Arduino.

3.2.1. FreeRTOS

O FreeRTOS, como um sistema operacional de tempo real, possibilita o gerenciamento multitarefa, oferecendo maior eficiência e paralelismo. Recursos nativos, como filas e semáforos, permitem um controle eficiente das tarefas e tornam o sistema mais confiável e organizado. Essas funcionalidades são particularmente úteis em aplicações críticas, como sistemas IoT, onde a previsibilidade e a estabilidade são fundamentais. [FreeRTOS 2024]

A implementação do FreeRTOS, embora poderosa, não é trivial, pois exige acesso e ajustes nas camadas mais baixas do sistema. No entanto, a ampla comunidade *open source* e a vasta documentação disponível oferecem suporte significativo, facilitando o desenvolvimento mesmo para projetos complexos.

3.2.2. Framework do Arduino

O *framework* do Arduino é uma abstração amplamente utilizada no desenvolvimento de sistemas embarcados, oferecendo uma interface amigável e diversas bibliotecas que simplificam a criação de aplicações. Baseado em C++, ele traz conceitos de orientação a objetos, permitindo maior modularidade e organização no desenvolvimento. [PlatformIO 2025]

Embora o uso desse *framework* facilite a implementação, ao fornecer um esqueleto pré-configurado para o código do desenvolvedor, ele também abstrai o acesso direto ao *hardware*. Essa característica pode dificultar ajustes em camadas mais baixas do sistema, o que pode ser uma limitação em projetos que demandam maior controle sobre o *hardware*. [Cavalcanti 2025]

3.3. Comunicação LoRaWAN

Os dados de consumo hídrico são transmitidos para um gateway LoRaWAN, que os encaminha a um servidor na nuvem para análise e monitoramento. Para a transmissão, foi utilizado o módulo RFM95W[1], configurado para operar na frequência de 915 MHz.

É importante destacar que, neste trabalho, não foram realizadas medições sobre a quantidade de pacotes enviados ou perdidos, uma vez que o escopo está focado na análise do consumo energético em diferentes implementações de *software*.

3.4. Ambiente de Testes

O ambiente de testes foi projetado para avaliar o consumo energético do sistema de forma precisa e controlada, garantindo a comparabilidade entre as abordagens implementadas (FreeRTOS e *framework* do Arduino).

[1] *Datasheet* transceptor RFM95W

O INA219[2] é um sensor de alta precisão utilizado para monitorar corrente, tensão e potência em tempo real. Ele foi integrado ao circuito para registrar os dados de consumo do sistema durante os testes. Para fornecer uma alimentação estável e eliminar variações inerentes ao uso de baterias, foi utilizada uma fonte de bancada configurada para fornecer uma tensão constante.

Os dados obtidos pelo INA219 foram lidos e processados por uma ESP32, que registrou os valores medidos. Para simular o funcionamento do hidrômetro e garantir a consistência dos testes entre as duas abordagens de *software*, foi utilizado um pino GPIO da ESP32, configurado para gerar pulsos eletrônicos correspondentes ao fluxo de água. Cada pulso gerado no pino GPIO equivalia a um litro de água registrado pelo hidrômetro.

Durante os experimentos, o consumo elétrico do sistema foi monitorado ao longo da simulação de 100 litros de água, distribuídos em 20 minutos, resultando em uma vazão média de 5 litros por minuto. A escolha desse volume permitiu a obtenção de uma quantidade significativa de dados em um intervalo de tempo relativamente curto, facilitando a análise comparativa entre as abordagens de *software*. A frequência dos pulsos foi mantida constante, assegurando que as variações no consumo de energia estivessem associadas principalmente ao gerenciamento do sistema pelo *software*.

A Figura 3 apresenta o esquema de ligação utilizado para a coleta dos dados, incluindo o sensor INA219, o pino GPIO da ESP32 e a configuração geral do sistema para os testes.

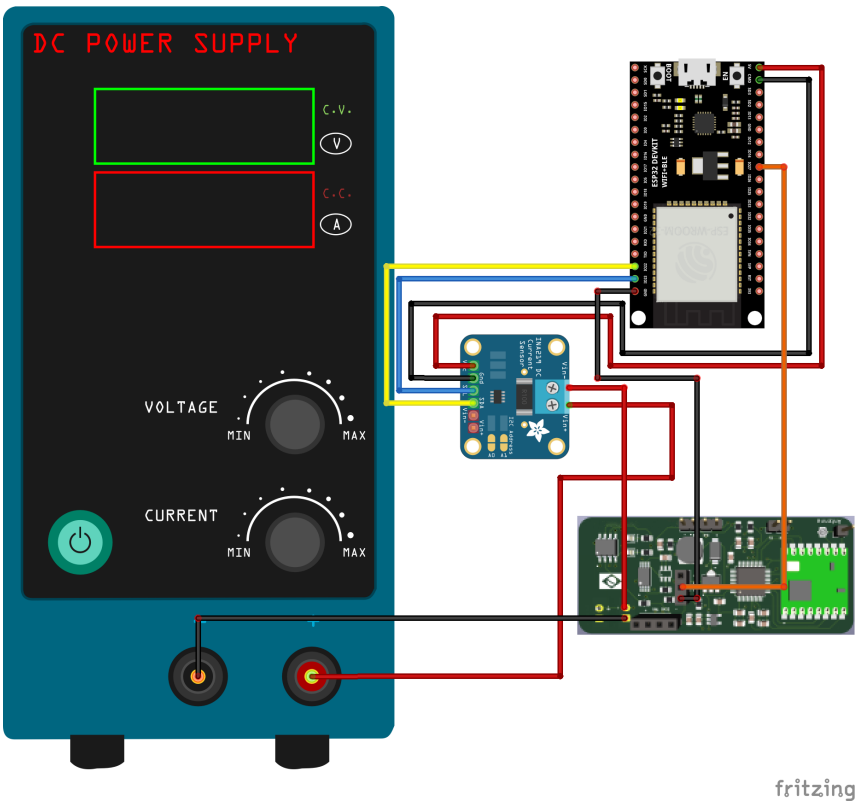

Figure 3. Esquema de testes.

[2]Guia de utilização do módulo INA219

Como ilustrado na Figura 3, a fonte de bancada alimenta o sistema, passando pelo sensor INA219, que, por sua vez, está conectado à ESP32 via I2C. A ESP32 também está conectada ao protótipo através do GPIO, que simulará o pulso gerado pelo hidrômetro. Vale ressaltar, ainda, que todos os componentes devem estar no mesmo potencial, ou seja, compartilhando o mesmo GND, para evitar instabilidades na captura dos dados pelo sensor INA219.

3.5. Métrica de Avaliação e Análise de Desempenho

A eficiência energética do sistema foi avaliada com base no consumo de energia durante diferentes estados de operação. Para isso, utilizou-se o sensor INA219 como amperímetro digital, responsável por capturar os dados de corrente consumida. A tensão constante de 3,7 V foi aplicada para calcular a potência consumida em cada estado do sistema.

Para a análise comparativa, foram considerados parâmetros como média, mediana e desvio padrão do consumo de energia durante os testes para cada sistema (Arduino e FreeRTOS). A média fornece uma visão geral do consumo de energia, enquanto a mediana ajuda a entender o valor central do consumo, especialmente em casos de distribuição assimétrica. Já o desvio padrão foi utilizado para avaliar a variação no consumo de energia, auxiliando na medição da estabilidade do sistema em cada implementação.

O consumo foi monitorado durante os estados de operação do sistema, incluindo os períodos de transmissão de dados, interrupções e inatividade, o que permitiu identificar os picos de consumo de energia, que foram analisados separadamente. A eficiência energética foi medida considerando a quantidade de dados transmitidos em relação ao consumo de energia, permitindo avaliar qual implementação se mostrou mais eficiente em termos de energia utilizada para realizar o trabalho útil, ou seja, a transmissão de dados.

4. Resultados

A análise de desempenho energético do sistema foi realizada para duas implementações distintas: uma utilizando o *framework* Arduino e a outra utilizando o sistema operacional em tempo real FreeRTOS. A seguir, são apresentados os resultados quantitativos sobre o consumo de energia durante o período de 20 minutos de coleta, seguidos de uma análise detalhada de um intervalo de 1 minuto para observar as flutuações de consumo.

4.1. Consumo Energético Total

Os resultados do consumo energético para ambos os sistemas ao longo de 20 minutos são apresentados na Tabela 1. O consumo de energia foi monitorado utilizando o sensor INA219, e os valores de média, mediana e desvio padrão foram calculados.

Sistema	Média (mW)	Mediana (mW)	Desvio Padrão (mW)
Arduino	156,19	150,4	28,36
FreeRTOS	41,45	40,8	27,22

Table 1. Estatísticas do Consumo Energético para Arduino e FreeRTOS

Como pode ser observado na Tabela 1, o Arduino apresentou um consumo médio quase quatro vezes maior que o FreeRTOS. A mediana também manteve essa relação,

enquanto o desvio padrão foi semelhante em ambos os casos. Isso ocorre porque, em ambas as situações, há picos de 1000 mW regularmente a cada 1 minuto, além de pequenas variações a cada 12 segundos, causadas pelo processamento dos dados lidos do hidrômetro. O gerenciamento multitarefa do FreeRTOS e o acesso direto ao *hardware* permitiram, portanto, uma distribuição mais eficiente do consumo de energia.

4.2. Desenvolvimento no FreeRTOS versus Arduino

O desenvolvimento utilizando FreeRTOS, como esperado, foi mais complexo, porém proporcionou um controle muito mais refinado sobre o *hardware*, permitindo, por exemplo, o acesso direto aos componentes responsáveis pela gestão de energia do microcontrolador. Na plataforma STM32F030C8T6, foi possível utilizar o modo *SLEEP*, que reduz significativamente o consumo de energia sempre que o dispositivo não está executando tarefas. Esse modo desativa parte dos periféricos internos, mantendo operacionais apenas os essenciais, como o Relógio de Tempo Real (RTC) e a memória, o que resulta em uma expressiva economia energética. Essa abordagem demandou um desenvolvimento consideravelmente mais extenso, totalizando 1.789 linhas de código em arquivos `.c` e 764 linhas em arquivos `.h`, desconsiderando os arquivos de *drivers* gerados automaticamente pelo STM32CubeIDE.

Por outro lado, a implementação na plataforma Arduino foi substancialmente mais simples, com um desenvolvimento ágil e de menor complexidade. O código resultante possui apenas 56 linhas, utilizando duas bibliotecas principais: `SPI.h` e `LoRa.h`. Contudo, essa simplicidade tem como contrapartida limitações significativas no controle do consumo energético, uma vez que a abstração oferecida pela plataforma restringe o acesso a funcionalidades avançadas de gestão de energia. Como resultado, os componentes do microcontrolador permaneceram em funcionamento contínuo, mesmo nos períodos sem atividade, elevando o consumo energético.

4.3. Consumo de Energia entre os Modos

O consumo de energia do sistema em diferentes modos foi comparado entre os sistemas Arduino e FreeRTOS na Tabela 2.

Modo de Operação	Arduino (mW)	FreeRTOS (mW)
Sistema em espera	150	40
Interrupção	200	70
Envio de Dados	1300	1000

Table 2. Comparação do Consumo de Energia entre Arduino e FreeRTOS

Embora o Arduino apresente um pico de consumo de 1300 mW durante o envio de dados, enquanto no FreeRTOS esse valor seja de 1000 mW, essa diferença se deve à forma como a implementação foi realizada. O módulo LoRa, responsável pela transmissão, demanda aproximadamente 1000 mW em ambos os casos. No entanto, o FreeRTOS se destaca por sua maior eficiência energética durante os períodos de inatividade. Enquanto o FreeRTOS consome apenas 40 mW em modo de espera (*sleep*), o Arduino consome 150 mW nesse mesmo estado. Essa diferença é significativa, considerando que, na maior parte do tempo, o sistema permanece em modo de espera.

4.4. Gráfico de Consumo Energético

A Figura 4 apresenta o gráfico dos testes, onde o eixo vertical representa a potência em milliwatts, e o eixo horizontal, o tempo total de 20 minutos para ambos os casos. A linha azul corresponde ao teste utilizando o *framework* do Arduino, enquanto a linha laranja representa o teste com o FreeRTOS.

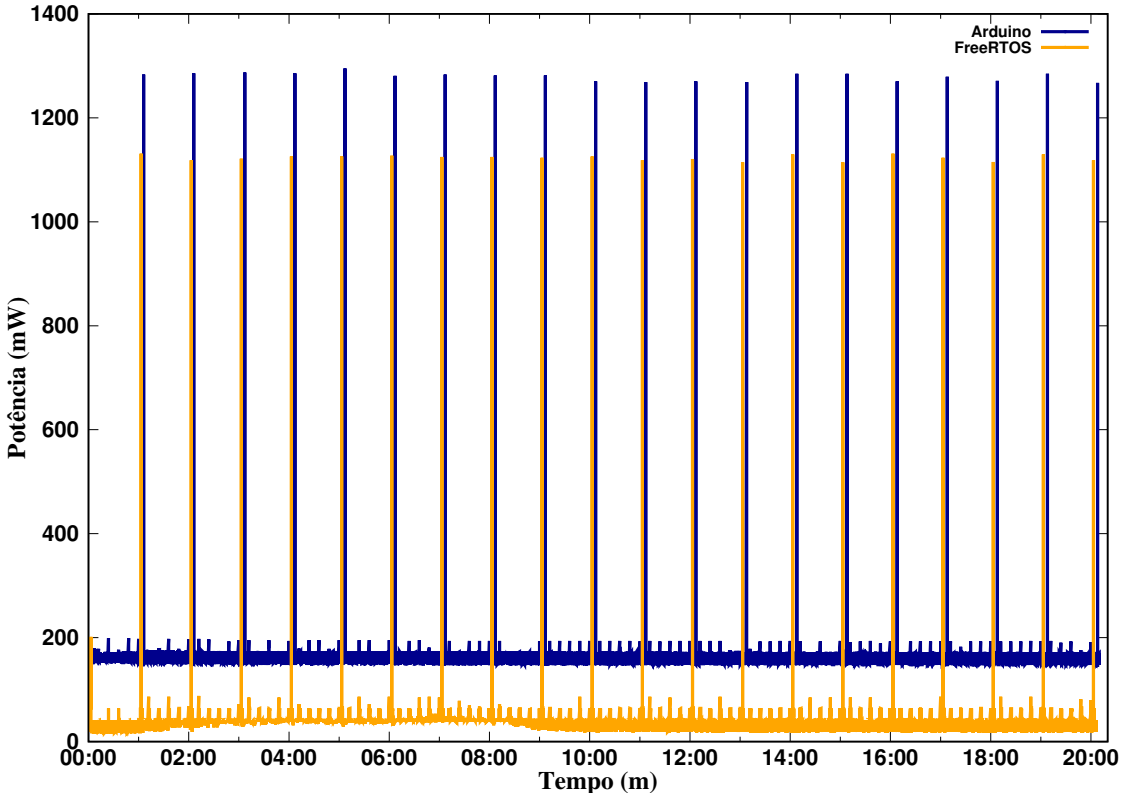

Figure 4. Consumo Energético total do sistema

Em ambos os casos, observam-se picos regulares de consumo aproximadamente a cada 1 minuto, resultantes do envio de dados via LoRa, totalizando 20 picos principais ao longo do período analisado. Além disso, nota-se a presença de flutuações associadas às interrupções provocadas pelo sensor de fluxo de água, que ocorrem a cada 12 segundos. Algumas dessas pequenas flutuações podem não parecer totalmente coerentes com esse intervalo, possivelmente devido à sensibilidade do sensor INA219, que pode ter influenciado a leitura em períodos muito curtos de tempo, como durante a execução da rotina de interrupção da leitura do sensor, resultando em pequenos picos não completamente representados no gráfico. Também é possível perceber que os gráficos apresentam comportamentos similares, porém deslocados verticalmente.

Com base no gráfico e nas métricas estatísticas, conclui-se que o FreeRTOS demonstrou maior eficiência energética em comparação ao Arduino. Ambos realizaram a mesma tarefa de leitura e envio de dados, porém com níveis de consumo distintos, sendo o FreeRTOS mais econômico.

5. Conclusão

O sistema utilizando o FreeRTOS apresenta uma eficiência energética superior quando comparado ao sistema baseado no Arduino. O FreeRTOS é capaz de operar com um consumo mais baixo, principalmente devido ao seu gerenciamento eficiente de energia durante os períodos em que não há necessidade de processamento. Isso resulta em um consumo mais equilibrado ao longo do tempo, permitindo que o sistema aproveite melhor os períodos de inatividade.

O Arduino, por outro lado, consome mais energia quando não está processando nada e durante as interrupções. Esse comportamento torna o FreeRTOS mais adequado para sistemas que exigem alta eficiência energética, como dispositivos de monitoramento remoto, onde a duração da bateria é crucial. Em contrapartida, o Arduino é mais indicado para projetos onde a simplicidade e a rapidez no desenvolvimento são mais importantes do que a eficiência energética extrema.

Os resultados obtidos neste estudo reforçam a importância da escolha do *software* embarcado no desenvolvimento de sistemas IoT, especialmente em dispositivos que dependem de alimentação por bateria. Em aplicações como monitoramento ambiental, telemetria rural e gestão de infraestrutura urbana, a eficiência energética impacta diretamente a viabilidade e os custos de manutenção desses sistemas. Dispositivos que operam por longos períodos sem intervenção humana podem se beneficiar significativamente de estratégias avançadas de economia de energia, como as implementadas pelo FreeRTOS.

A partir dos dados obtidos neste estudo, futuros trabalhos poderão expandir a análise comparativa, incluindo outros sistemas operacionais e abordagens de programação, como o *Zephyr* e a programação *baremetal* no STM32. Esta comparação permitirá uma compreensão mais profunda das diferenças no consumo energético e no desempenho entre essas plataformas em sistemas embarcados, fornecendo informações valiosas para a escolha da abordagem mais adequada com base nas exigências de eficiência energética para diferentes tipos de aplicações IoT.

Além disso, será interessante explorar o impacto de estratégias de *energy harvesting* (captação de energia) em conjunto com o gerenciamento de energia. A utilização de fontes alternativas de energia, como painéis solares ou termelétricos, pode permitir que dispositivos IoT operem por períodos prolongados sem a necessidade de troca constante de baterias, especialmente em aplicações de monitoramento remoto e em ambientes de difícil acesso. A integração dessas soluções com sistemas de gerenciamento energético como o FreeRTOS pode proporcionar um aumento significativo na autonomia de dispositivos IoT, com o mínimo impacto no consumo de energia.

Por fim, os achados deste estudo reforçam que a escolha da arquitetura de *software* pode ser tão crítica quanto a escolha do *hardware* na otimização de sistemas embarcados. O FreeRTOS, por oferecer um controle mais refinado sobre o consumo energético, se mostrou uma solução mais eficiente para aplicações que exigem alta autonomia, enquanto o Arduino permanece uma opção viável para projetos que priorizam simplicidade e tempo de desenvolvimento reduzido.

6. Referências

[ANA 2024] ANA (2024). *Conjuntura dos Recursos Hídricos no Brasil 2024*. ANA.

[Ashton 2009] Ashton, K. (2009). That 'internet of things' thing. *RFID Journal*, 22(7):97–114.

[Ayatullah et al. 2023] Ayatullah, M. D., Wibowo, G. H., Febrita, R. E., Prasetyo, J. A., Sarosa, M., and Hapsari, R. I. (2023). Smart water meter based on lora communication technology. *International Conference on Advanced Mechatronics, Intelligent Manufacture and Industrial Automation (ICAMIMIA)*.

[Cavalcanti 2025] Cavalcanti, M. (2025). Microcontrolador: Framework - o que é e quais as suas vantagens? Accessed: January 11, 2025.

[Cooling 2020] Cooling, J. (2020). *Real-Time Operating Systems: The Engineering of Real-Time Embedded Systems*. CreateSpace Independent Publishing Platform, 3rd edition.

[Felinto Filho 2023] Felinto Filho, G. G. (2023). Aplicações do lora para monitoramento hídrico. *Dissertação de Mestrado*.

[FreeRTOS 2024] FreeRTOS (2024). What is freertos? Accessed: January 9, 2025.

[González et al. 2020] González, E., Casanova-Chafer, J., Romero, A., Vilanova, X., Mitrovics, J., and Llobet, E. (2020). Lora sensor network development for air quality monitoring or detecting gas leakage events. *Sensors*, 20(21).

[Gorawski et al. 2024] Gorawski, M., Marjasz, R., Grochla, K., and Frankiewicz, A. (2024). Comparative analysis of energy consumption in simulated lora water meter reconfiguration vs. real-world readings. *IFIP Networking Conference*.

[Gubbi et al. 2013] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660.

[Kurdija et al. 2024] Kurdija, A., Mostarac, P., Željko Ilić, Bruvo, S., Šišul, G., and Marić, I. (2024). Implementation and analysis of lorawan technology for remote water meter reading in urban areas. *International Conference on Smart Systems and Technologies*.

[Migabo et al. 2021] Migabo, E., Djouani, K., and Kurien, A. (2021). Design of an energy efficient lorawan-based smart iot water meter for african municipalities. *International Conference on Electrical, Computer and Energy Technologies (ICECET)*.

[MME and EPE 2007] MME, M. d. M. e. E. and EPE, E. d. P. E. (2007). PNE 2030 - Plano Nacional de Energia 2006-2007. *Ministerio de Minas e Energia*, 11 Eficiên.

[Mór et al. 2010] Mór, S., Alves, M., Lima, J., Maillard, N., and Navaux, P. (2010). Eficiência energética em computação de alto desempenho: Uma abordagem em arquitetura e programação para green computing. In *Anais do XXXVII Seminário Integrado de Software e Hardware*, pages 346–360, Porto Alegre, RS, Brasil. SBC.

[PlatformIO 2025] PlatformIO (2025). Arduino framework — platformio documentation. Accessed: January 11, 2025.

[STMicroelectronics 2020] STMicroelectronics (2020). *STM32F0 Series Reference Manual*. Accessed: January 10, 2025.

[Tanenbaum and Bos 2022] Tanenbaum, A. S. and Bos, H. (2022). *Modern Operating Systems*. Pearson Education, 4th edition.