Desenvolvimento de Sistema de Processamento de Sinal em Tempo Real com FreeRTOS: Aplicações em Filtro IIR e Tarefas Periódicas.

Davi Ferreira
Eduardo Schvinn
Matheus Nunes Franco
Joinville, Brasil

***Abstract*—This work presents the development and implementation of a real-time signal processing system using FreeRTOS, an open-source real-time operating system. The system is applied to the implementation of IIR (Infinite Impulse Response) filters, whose characteristics were initially modeled and validated in the MATLAB environment before being integrated into the embedded system.**

The implementation of the IIR filter aims to attenuate specific signal characteristics, contributing to efficient data analysis and processing. Additionally, strategies for creating and executing periodic tasks are addressed, demonstrating how FreeRTOS manages the synchronization and prioritization of these tasks in real-time environments.

The case study uses a practical application to exemplify the application of the proposed system, integrating theoretical concepts with concrete implementations. The obtained results highlight the efficiency of FreeRTOS in executing real-time tasks and its applicability in signal processing applications.

This work contributes to understanding the capabilities of FreeRTOS in the context of real-time systems, emphasizing the integration between the MATLAB environment and the practical implementation of IIR filters. It provides valuable insights for researchers, developers, and engineers interested in embedded systems and signal processing.

Keywords: Keywords: FreeRTOS, Signal Processing, IIR Filters, Real-Time Systems.

I. Introdução

O desenvolvimento de sistemas de processamento de sinal em tempo real representa uma área crucial no âmbito da engenharia eletrônica e computacional. O uso de sistemas embarcados para manipulação de sinais em tempo real é essencial em diversas aplicações, desde comunicações sem fio até controle de dispositivos industriais. Este trabalho concentra-se na implementação prática desses sistemas, utilizando o FreeRTOS, um sistema operacional de tempo real de código aberto.

O FreeRTOS oferece um ambiente eficiente e robusto para o desenvolvimento de sistemas embarcados, sendo especialmente adequado para plataformas como as da família STM32. Neste contexto, exploramos aplicações avançadas, focalizando a implementação de filtros IIR (Infinite Impulse Response) e tarefas periódicas para processamento de sinais em tempo real.

A realização prática dessas aplicações é ilustrada através da criação de uma senoide com adição de ruído e sua subsequente filtragem por meio de um filtro IIR. O uso de tarefas periódicas, gerenciadas pelo FreeRTOS, proporciona uma abordagem eficiente e precisa para a manipulação de sinais em tempo real.

II. Proposta de projeto

Este projeto propõe a implementação de um sistema de processamento de sinal em tempo real com base no FreeRTOS, um sistema operacional de tempo real de código aberto. O foco central do sistema é a aplicação de filtros de Resposta ao Impulso Infinito (IIR) e tarefas periódicas. A investigação abordará a integração do FreeRTOS em sistemas embarcados, desenvolvendo tarefas específicas para geração de sinal, filtragem IIR e execução periódica. Utilizando o MATLAB, a pesquisa se concentrará no design e análise de filtros IIR, facilitando a transição fluida da simulação para a implementação em tempo real. O projeto visa fornecer percepções significativas sobre a viabilidade, eficiência e aplicabilidade do FreeRTOS em cenários práticos de processamento de sinal em tempo real.

III. Fundamentação Teórica

A. FreeRTOS: Sistema Operacional de Tempo Real

O FreeRTOS é um sistema operacional de tempo real de código aberto, projetado para aplicações embarcadas. Ele oferece um ambiente multitarefa preemptivo, que permite a execução concorrente de várias tarefas com prioridades diferentes [1]. Sua estrutura modular e leve o torna adequado para sistemas com recursos limitados, como microcontroladores e microprocessadores embarcados [2]. O FreeRTOS fornece mecanismos eficientes para a criação, suspensão, retomada e sincronização de tarefas, sendo uma escolha popular em projetos que demandam tempo real e eficiência.

B. Filtros de Resposta ao Impulso Infinito (IIR)

Os filtros IIR são uma categoria de filtros digitais que utilizam realimentação para obter uma resposta ao impulso infinita. Esses filtros são amplamente utilizados em processamento de sinal devido à sua eficiência computacional e flexibilidade de projeto [5]. Eles apresentam características que os tornam ideais para muitas aplicações, como a capacidade de implementar funções de transferência complexas com ordens

menores em comparação com filtros FIR (Resposta ao Impulso Finito) [4].

C. Tarefas Periódicas.

As tarefas periódicas referem-se à execução repetitiva de operações em intervalos regulares de tempo. Em sistemas embarcados de tempo real, as tarefas periódicas desempenham um papel crucial na sincronização e na garantia de que determinadas operações ocorram em momentos específicos [3]. Ao utilizar tarefas periódicas, é possível controlar o tempo de execução de diferentes componentes do sistema, como aquisição de dados, processamento e comunicação. Isso é essencial para aplicações que demandam precisão temporal, como sistemas de controle e processamento de sinais em tempo real [3].

IV. Metodologia

A. Implementação do Sistema com FreeRTOS

A implementação do sistema utilizando o FreeRTOS seguirá uma abordagem modular e orientada a tarefas, garantindo eficiência e concorrência para aplicações de processamento de sinais em tempo real.

1. Estrutura do Sistema:

O sistema será estruturado em torno de tarefas dedicadas a funções específicas, permitindo execução concorrente e otimizada. Para isto serão criados dois arquivos que implementaram o FreeRTOS, estes são:

2. Filtros

Esta implementação sera responsavel pelas funções **filtrosiir()** e **CriarSinalcomRuido()** que correspondem ao filtro IIR e a onda com ruido simulada.

Estas funções realizam sua operações simultaneamente com um periodo especifico, sendo este o parametro mais importante para sua funcionalidade, ja que sera corresponde a frequencia de amostragem do sinal.

O compartilhametos dos dados que devem ser amostrados e filtrados sera compartilhado a partir de uma fila global, onde a função que cria o sinal envia este dado para a fila e função que aplica o filtro, ler e exclui esse dado. Esta fila sera gerada utilizando a biblioteca queue, biblioteca padrão para criação de filas e que garante em sua configuração padrão boas condições de concorrência, evitando assim que as funções sobrescrevam seus dados ou recebam valores vazios.

3. Escalonador RM

Para a implementação do escalonador Rate Monotonic (RM) analisamos os períodos e os tempos de execução das tarefas. Para isso utilizamos um vetor para armazenar as informações.

Esses dados são utilizados pelo escalonador criado juntamente com o escalonador do FreeRTOS para ordenar as tarefas de acordo com o algoritmo RM. O FreeRTOS fornece recursos para definir prioridades de tarefas, e a implementação do RM utiliza esses recursos para atribuir prioridades dinamicamente, com base nos períodos das tarefas.

Essa integração entre os dados das tarefas e o escalonador do FreeRTOS permite um gerenciamento eficiente e ordenado das tarefas, garantindo que as mais críticas sejam executadas dentro dos prazos exigidos pelo sistema de tempo real.

B. Projeto e Análise de Filtros IIR no MATLAB

Nesta seção, descreveremos o projeto e a análise do filtro IIR utilizando o MATLAB. O foco é projetar um filtro Chebyshev Tipo I, aplica-lo a uma onda com ruído que simulará um sinal real, e analisar seu desempenho no domínio do tempo e da frequência.

Para criação do filtro Chebyshev I, utilizaremos a função cheby1, esta recebe como parametros 3 variaveis n, wp e wn, que são detalhados abaixo:

n (ordem do filtro): A ordem do filtro é um parâmetro que determina o número de polos e zeros em um filtro. Em filtros digitais, a ordem está diretamente relacionada à complexidade do filtro.

wp (Frequência de Passagem): A frequência de passagem é a frequência até a qual um filtro permite a passagem sem atenuação significativa. Em filtros passa-baixa, por exemplo, a frequência de passagem é a frequência até a qual o sinal é permitido passar sem ser significativamente atenuado.

wn (Frequência de Corte Normalizada): A frequência de corte normalizada é uma forma de expressar a frequência de corte em termos normalizados em relação à metade da taxa de amostragem ($Fs/2$), onde Fs é a taxa de amostragem. Em filtros digitais, a frequência de corte é muitas vezes especificada como uma fração da frequência de Nyquist.

Para definição desses parâmetros é necessario caracteristicas do sinal que esta sendo amostrado. Para simular este sinal geraremos uma senoide de 5Hz e aplicaremos a este ruidos gerados de maneira aleatoria, buscando simular sinais sensoriados em regiões com vibrações mecânicas como motores.

A senoide gerada tera uma frequência de 5Hz se baseando no método proposto de nyquist para amostragem de sinais, devemos no minimo amostrar uma frequência duas vezes maior. No entanto para melhor vizualização do sinal assumiremos uma frequência de amostragem de 100Hz, portanto o periodo de amostragem sera 10ms.

A onda com ruido gerada, pode ser vista na figura 1.

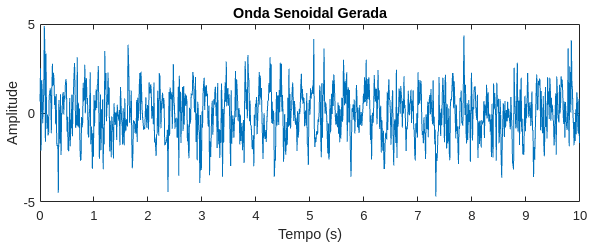


Fig. 1: Onda senoidal de 5Hz com ruido.

Serão gerados dois filtros, um passa baixa e um passa alta, para o filtro passa baixa adotaremos uma frequência de corte de 5hz, buscando preserva ao maximo o sinal original. Para o filtro passa alta adotaremos 30hz para frequência de corte.

A ordem do filtro foi determinada através de testes, a ordem de um filtro é um parâmetro importante, pois influência diretamente o comportamento do filtro em termos de atenuação e largura de banda. Filtros de ordem mais alta geralmente

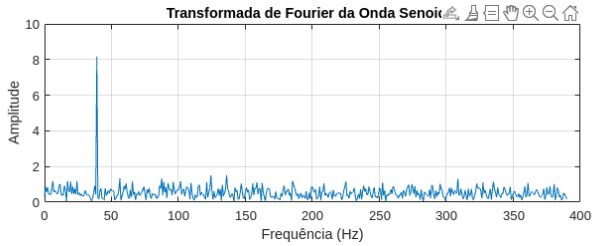


Fig. 2: Espectro da frequência do sinal gerado.

proporcionam uma resposta em frequência mais seletiva, mas também podem introduzir mais distorção de fase. Desta forma, optamos por aderir como valor ideal 2. Os sinais filtrados, passa baixa e passa podem ser vistos nas figuras 3 e 4.

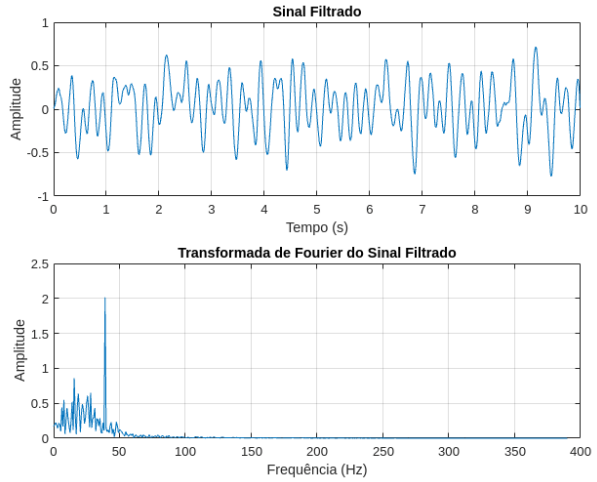


Fig. 3: Sinal filtrado - Filtro passa baixas.

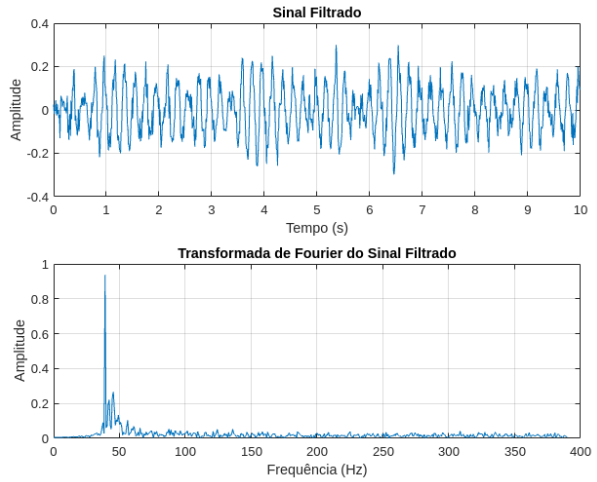


Fig. 4: Sinal filtrado - Filtro passa altas.

C. Configuração do Hardware STM32

O microcontrolador STM32F103C8T6, foi escolhido pela sua versatilidade e eficiência, desempenha um papel crucial em aplicações de processamento de sinais em tempo real. Sua configuração específica para essas finalidades envolve uma compreensão detalhada de suas características e periféricos, destacando seu potencial para manipulação de sinais estas são:

Fig. 5: STM32F103C8T6 (BluePill).

Núcleo ARM Cortex-M3: O STM32F103C8T6 incorpora o núcleo ARM Cortex-M3, proporcionando um desempenho eficiente e baixo consumo de energia, essenciais para aplicações de processamento de sinais. E em aplicações que envolvem comunicação em tempo real, é necessário garantir que as operações sejam concluídas dentro de prazos específicos. O desempenho eficiente é crucial para atender a esses requisitos temporais.

Periféricos Avançados: Além dos periféricos padrão, como GPIO, UART, SPI e I2C, o microcontrolador possui recursos específicos, como timers avançados, ADC (Conversor Analógico para Digital) de alta precisão e capacidade de controle PWM (Modulação por Largura de Pulso), tornando-o ideal para captura e processamento de sinais analógicos.

ADC de Alta Resolução: Configurar o ADC para garantir uma aquisição precisa de sinais analógicos, permitindo uma digitalização eficiente e de alta qualidade.

Configuração do Ambiente de Desenvolvimento:

A escolha do STM32CubeIDE como ambiente de desenvolvimento é estratégica para explorar todo o potencial do microcontrolador em aplicações de processamento de sinais. Os passos específicos para configuração incluem:

D. Integração e Configuração dos Componentes

NA integração entre o hardware STM32 e o software desenvolvido no MATLAB para a aplicação do filtro IIR é uma etapa crucial para garantir a operação eficiente do sistema de processamento de sinais em tempo real. Esta seção descreve o processo de incorporação da função de transferência projetada no MATLAB no ambiente de programação STM32, destacando os passos necessários para garantir uma implementação coesa.

Com o filtro desenvolvido e testado no MatLab, podemos exibir a função de transferencia do filtro IIR. A função de transferência de um filtro é uma representação matemática que descreve a relação entre a entrada e a saída do filtro em termos

de frequência. A função de transferência de um filtro digital IIR pode ser expressa como:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}} \quad (1)$$

Para implementação no hardware, faz necessario encontrar a função inversa de z, para analisa-la no dominio do tempo, esta pode ser expressa como:

$$\frac{y(n)}{x(n)} = \mathcal{Z}^{-1}\left\{\frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_n z^{-n}}\right\} \quad (2)$$

Com a formula acima podemos isolar y(n) e encontrar as equações de transferência dos filtros em função do tempo.

Função de transferencia no dominio do tempo para o filtro passa baixa:

$$Y(n) = 1.699 \cdot y(n-1) - 0.7601 \cdot y(n-2)$$

$$+ 0.01356 \cdot x(n) + 0.02712 \cdot x(n-1) + 0.01356 \cdot x(n-2) \quad (3)$$

Função de transferencia no dominio do tempo para o filtro passa alta:

$$Y(n) = 1.845 \cdot y(n-1) - 0.966 \cdot y(n-2)$$

$$+ 0.2912 \cdot x(n) - 0.5824 \cdot x(n-1) + 0.2912 \cdot x(n-2) \quad (4)$$

Com as funções calculadas podemos implementas na função filtrosIRR() na IDECube.

E. Avaliação do Desempenho

A avaliação de desempenho de sistemas de tempo real é uma área ampla e pode depender do contexto específico do sistema em questão. No entanto pode-se dizer que o desepenho de um STR, esta vinculado ao escalonamento adequado de tarefas e se todas as tarefas executaram sem perde seu deadline. Portanto para avaliação do desempenho deste projeto serão realizados duas etapas, a primeira consiste em teste de escalonamento e avaliação do tempo de resposta de cada tarefa com métodos teoricos e a vizualização deste escalonamento com a utilização de ferramentes de Trace. Para realização do escalonamento teorico, utilizaremos a equação de viabilidade de escalonamento de tarefas para um rate monotonic (RM). O escalonamento por Rate Monotonic (RM) é um algoritmo de escalonamento utilizado em sistemas de tempo real, onde as tarefas são atribuídas prioridades com base nos seus períodos. Esta pode ser expressa como:

$$\sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) \leq N(2^{1/N} - 1) \quad (5)$$

Ci = Tempo de computação das tarefas. Pi = Periodo das tarefas. N = O numero de tarefas.

O período e o número de tarefas são parâmetros que podem ser facilmente definidos. No entanto, obter o tempo de computação é uma tarefa mais complexa. Com o objetivo de obter um valor que se aproxime ao máximo do que será realizado no projeto, decidimos calcular o tempo de computação das funções de filtro e geração de onda em sua forma mais simples, sem a utilização do FreeRTOS e variáveis compartilhadas. Para isso, reescrevemos o código em C com o propósito de medir exclusivamente o tempo necessário para os cálculos em cada função.

Como as funções foram simplificadas em sua estrutura, o tempo de execução foi obtido em milisegundos utilizando a biblioteca time.h. Na tabela 1 e possivel observar o tempo de computação e periodo encontrados.

TABLE I: Parêmetros de

Ti	Ci(ms)	Pi(ms)
T1	0,004	10
T2	0,007	10

Com os valores podemos aplicar o teste de escalonamento RM:

$$\left(\frac{0.023 + 0.006}{10 + 10}\right) \leq 2(2^{1/2} - 1) \quad (6)$$

$$0.00145 \leq 0.828 \quad (7)$$

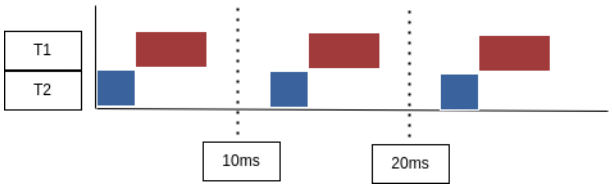


Fig. 6: Escalonamento teórico.

Com base no resultado da inequação podemos presumir que o sistema e escalonavel, na figura 6 e possivel ver o escalonamento teorico desde sistema.

Para uma avaliação mais aprofundada do desempenho, é fundamental realizar a visualização escalonada do sistema no hardware. Para essa finalidade, empregaremos ferramentas de Trace, que se referem a recursos utilizados para capturar, visualizar e analisar dados de rastreamento em sistemas. Essas ferramentas são comumente aplicadas em atividades de desenvolvimento de software, depuração, otimização de desempenho e análise de sistemas em tempo real.

Nesta pesquisa sera utilizada a ferramenta SystemView, é uma ferramenta de análise de sistema em tempo real projetada para fornecer informações detalhadas sobre a execução do código em sistemas embarcados em tempo real (RTOS - Real-Time Operating Systems). Desenvolvido pela empresa SEGGER Microcontroller, o SystemView é amplamente utilizado em ambientes de desenvolvimento de software embarcado para otimização de desempenho e diagnóstico de sistemas em tempo real.

Os resultados encontrados pelo SystemView seram demonstrados e analisados na etapa de resultados.

V. Modelo de projeto proposto

Com base no que foi discutido nos topicos acima, é proposto uma estrutura do sistema, levando em contas as seguintes necessidades do usuario:

- Capacidade de escolher o filtro.
- Capacidade de vizualização do sinal filtrado.

A comunicação do usuario com o hardware deve ser realizada atraves de um intermediario, neste caso um computador, esta sera responsavel por enviar requisições ao hardware e disponivilizar a exibição dos sinais para o usuario. Com base nistas necessidades temos o modelo proposto na figura 7.

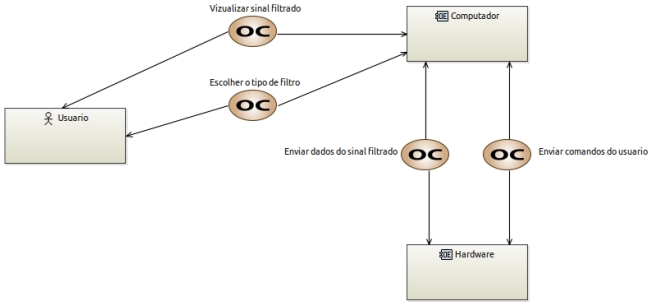


Fig. 7: Modelagem do sistema.

Para estabelecer a comunicação com o computador, será utilizado o protocolo UART para enviar dados para uma porta serial. Para transmitir dados via USB para o computador, é necessário converter a saída do STM32. Essa conversão será realizada por meio de um conversor FTDI232 chip conversor USB – UART, permitindo o envio de dados em tempo real para serem monitorados pelo usuário.

As conexões necessarias entre os dois dispositivos pode ser vista na figura 8.

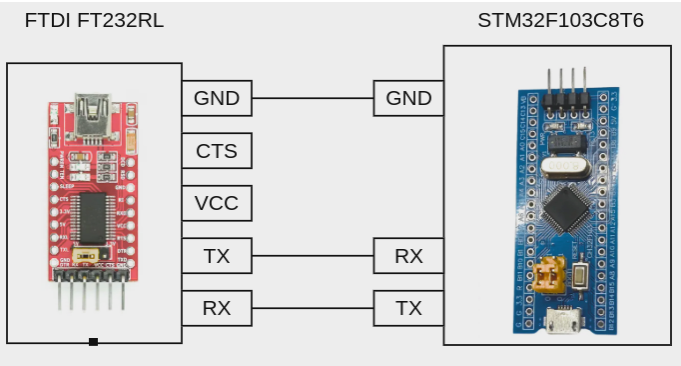


Fig. 8: conexão entro o conversor FTDI e o STM32.

VI. Resultados

A. Avaliação do escalonamento de tarefas

Para avaliação do escalonamento das tarefas periódicas, foi comparado o escalonamento realizado de maneira teórica com o escalonamento real encontrado pelo software SystemView, conforme ilustrado na Figura 8.

A primeira diferença notada foi a presença de outras tarefas no escalonamento. Essas tarefasq são essenciais para garantir o funcionamento adequado do sistema STM32, tornando impraticável desativá-las. No entanto, é perceptível que as tarefas T1 e T2 continuam a ser executadas no período correto de 10ms. Contudo, devido à presença de tarefas com maior prioridade (período de execução menor), as tarefas T1 e T2 estão sendo interrompidas, sem ocasionar problemas no escalonamento.

Vale ressaltar que a interrupção ocorrida no início de cada execução de T1 é esperada devido às características do compartilhamento entre as tarefas. Isso ocorre porque elas utilizam uma variável para compartilhamento de recursos, onde, em nossa lógica, T2 escreve um dado e T1 o lê e remove. Esse processo é protegido por um mutex, que só permite a execução quando há dados a serem lidos por T1. Portanto, tendo T1 e T2 o mesmo período, as regras estabelecidas pelo compartilhamento de recursos impedem bloqueios ou sobreposição de dados.

Foi possivel tambem constatar, o tempo real de computação das tarefas, que foram:

TABLE II: Parêmetros de

Ti	C real	C teorico
T1	0,349	0,004
T2	0,480	0,007

Embora haja discrepância entre os tempos de computação teórico e real, é fundamental ressaltar que os tempos teóricos foram derivados de uma simulação que não considerou o compartilhamento de recursos. Adicionalmente, a execução da simulação ocorreu em um computador, fator que pode ter contribuído para a disparidade observada nos tempos.

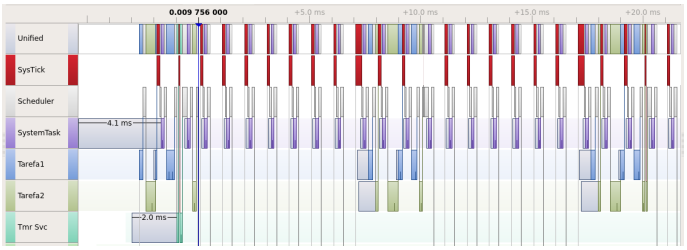


Fig. 9: Escalonamento de tarefas gerado pelo SysView.

Apesar das variações nos valores reais em comparação aos teóricos, o escalonamento real demonstrou-se em conformidade com as expectativas teóricas. As tarefas estão sendo executadas nos períodos planejados, e este é o parâmetro crucial para assegurar a adequada execução do projeto.

B. Avaliação dos sinais filtrados

Alcançamos uma redução significativa de ruídos nos sinais filtrados ao utilizar os filtros de Chebyshev, tanto para a faixa de passagem-baixa quanto para a de passagem-alta de frequência. Estes filtros foram implementados de maneira específica para atenuar os ruídos associados às vibrações mecânicas presentes em sinais com uma frequência de 5 Hz.

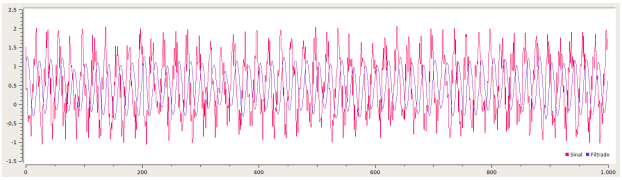


Fig. 10: Sinal filtrado - Filtro passa baixa.

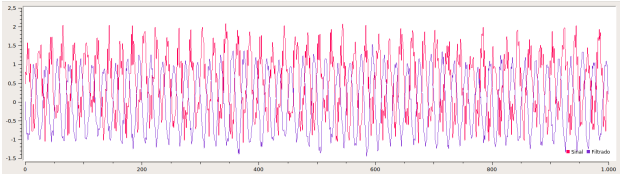
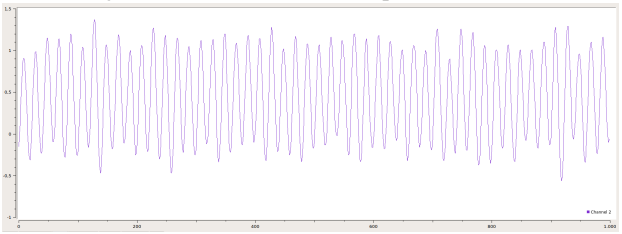


Fig. 11: Sinal filtrado - Filtro passa alta.



captionoffigureSomente sinal filtrado - passa baixa.

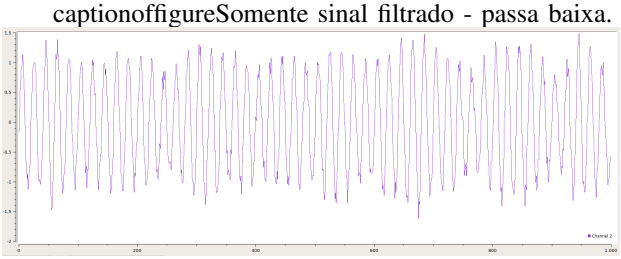


Fig. 12: Somente sinal filtrado - passa alta.

VII. Considerações finais

Demonstramos com êxito a viabilidade de interações em tempo real de uma aplicação de filtros IIR, atendendo requisitos específicos de tempo em um sistema embarcado. Utilizamos o SystemView para monitorar em tempo real a chegada de eventos e seu tempo de execução. Adicionalmente, validamos os resultados por meio do MATLAB, apresentando os sinais originais e suas respectivas Transformadas Rápidas de Fourier (FFT) após o processo de filtragem. Confirmamos, por meio de testes manuais de escalonamento utilizando o algoritmo Rate-Monotonic (RM) e validação com a ferramenta de trace data, que nenhuma tarefa ultrapassou seu prazo limite.

É importante destacar que a condução de testes utilizando sinais provenientes de sensores reais, juntamente com uma metodologia mais refinada para determinar os tempos de execução das tarefas, levando em consideração as limitações do hardware utilizado é crucial para aprimorar a precisão do sistema.

References

1 - FreeRTOS - Real Time Operating System (RTOS) for embedded microcontrollers. Disponível em: https://www.freertos.org/

2 - Barry, P., Wilmshurst, T. (2006). Embedded systems: Introduction to ARM Cortex-M Microcontrollers. Newnes.

3 - OLIVEIRA, Rômulo Silva de; FARINES, Jean-Marie; FRAGA, Joni da Silva. Sistemas de Tempo Real. Florianópolis: ., 2000. 208 p.

4 - MATTAR, Karina Mahon. IMPLEMENTAÇÕES EM PROCESSAMENTO DIGITAL DE SINAIS UTILIZANDO O TMS320C6711. 2006. 163 f. Tese (Doutorado) - Curso de Engenharia Elétrica, Universidade de Brasília, Brasília, 2006.