UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MURILO MARQUES DOS SANTOS

PROTÓTIPO DE MEDIDOR DE COMPONENTES DE FREQUÊNCIA EM SISTEMAS ELÉTRICOS BASEADO NO ALGORITMO *ANTI-LEAKAGE MATCHING PURSUIT*

PATO BRANCO

2023

MURILO MARQUES DOS SANTOS

PROTÓTIPO DE MEDIDOR DE COMPONENTES DE FREQUÊNCIA EM SISTEMAS ELÉTRICOS BASEADO NO ALGORITMO *ANTI-LEAKAGE MATCHING PURSUIT*

Frequency Component Meter Prototype in Electrical Systems Based on the Anti-Leakage Matching Pursuit Algorithm

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do Curso de Bacharelado em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Giovanni Alfredo Guarneri

Coorientador: Prof. Dr. Gustavo Weber Denardin

PATO BRANCO

2023

MURILO MARQUES DOS SANTOS

PROTÓTIPO DE MEDIDOR DE COMPONENTES DE FREQUÊNCIA EM SISTEMAS ELÉTRICOS BASEADO NO ALGORITMO *ANTI-LEAKAGE MATCHING PURSUIT*

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do Curso de Bacharelado em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 14/junho/2023

Giovanni Alfredo Guarneri
Doutorado
Universidade Tecnológica Federal do Paraná

Juliano de Pelegrini Lopes
Doutorado
Universidade Tecnológica Federal do Paraná

Marcelo Gonçalves Trentin
Doutorado
Universidade Federal de Santa Maria

PATO BRANCO

2023

Este trabalho é dedicado a todos aqueles que
se dedicam a utilizar e aprimorar a tecnologia
para fazer do mundo um lugar melhor.

AGRADECIMENTOS

Agradeço a todos os meus familiares em especial aos meus pais Zouraide Marques Pereira dos Santos e Dorival dos Santos por todo apoio, carinho e paciência durante todo este meu tempo distante.

Aos meus orientadores Prof. Dr. Giovanni Alfredo Guarneri e Prof. Dr. Gustavo Weber Denardin por todo conhecimento repassado, auxilio e disposição durante o desenvolvimento deste trabalho.

Aos laborataristas Célio Antônio Degaraes e os demais profissionais do Departamento de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná do campus Pato Branco por toda estrutura e suporte oferecido.

Aos meus amigos e Bruno Bohn dos Santos, Joel Ferrari Carvalho, Eduardo André Kosinski Fonseca, Jeverton Nardi, Wagner Pinto de Lima, entre muitos outros, que não contribuíram diretamente com este trabalho, mas fortemente com toda a minha formação acadêmica.

"Estou alcançando o que antes eu dizia pra mim: ’Se você acha que consegue então vai ser assim Tudo depende da vontade e a determinação Então vou ver se isso é um plano ou só uma ilusão’."(PENSE, 2011)

RESUMO

Motivado pelos estudos sobre qualidade de energia e distúrbios na rede elétrica, este trabalho consiste no desenvolvimento de um protótipo de medição de componentes de frequência harmônicas e inter-harmônicas utilizando um algoritmo recente chamado *Anti-Leakage Matching Pursuit* (ALMP) em um *kit* de desenvolvimento com um microcontrolador dedicado para processamento de sinais. Com base em fontes de distorções harmônicas e inter-harmônicas reais, sinais foram criados e reproduzidos em um gerador de funções e adquiridos pelo protótipo que iterativamente estima a amplitude, frequência e fase de cada uma das componentes de frequência presentes nesses sinais utilizando algoritmo ALMP. Para o funcionamento deste projeto foram utilizados *softwares* e bibliotecas no microcontrolador, como o sistema operacional de tempo real `FreeRTOS` que gerencia o tempo de execução de cada atividade do microcontrolador, a biblioteca `levmar` para resolução de mínimos quadrados não lineares, necessário para a execução do algoritmo ALMP e a biblioteca `LVGL` para o desenvolvimento e exibição de interfaces gráficas em *displays* LCD. Durante e após a aquisição e estimação dos sinais, cada uma das componentes de frequência e seus parâmetros de interesse são apresentados em uma interface gráfica no *display* do *kit* de desenvolvimento, essas componentes de frequência são comparadas com as dos sinais originais e suas formas de onda verificadas com as das reproduzidas em um osciloscópio.

Palavras-chave: estimação de harmônicos e inter-harmônicos; processamento de sinais; protótipo de medição.

ABSTRACT

Motivated by studies on power quality and disturbances in the electrical grid, this work consists of developing a prototype for measuring harmonic and inter-harmonic frequency components using a recent algorithm called Anti-Leakage Matching Pursuit (ALMP) on a discovery kit with a dedicated microcontroller for signal processing. Based on real sources of harmonic and inter-harmonic distortions, signals were generated and reproduced in a function generator, and acquired by the prototype, which iteratively estimates the amplitude, frequency, and phase of each frequency component present in these signals using the ALMP algorithm. For the functioning of this project, softwares and libraries were utilized on the microcontroller, such as the real-time operating system `FreeRTOS`, which manages the execution time of each microcontroller activity, the `levmar` library for solving non-linear least squares, necessary for the execution of the ALMP algorithm, and the `LVGL` library for the development and exhibition of graphical interfaces on LCD displays. During and after the acquisition and estimation of the signals, each frequency component and its parameters of interest are presented in a graphical interface on the discovery kit's display, these frequency components are compared with those of the original signals, and their waveforms are verified with those ones reproduced on an oscilloscope.

Keywords: harmonics and inter-harmonics estimation; signal processing; measurement prototype.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

LVGL	*Light and Versatile Graphics Library*
ADC	*Analog-to-digital converter*
ALMP	*Anti-Leakage Matching Pursuit*
CPU	*Central Processing Unit*
DFT	*Discrete Fourier Transform*
DMA	*Direct Memory Access*
DSP	*Digital Signal Processor*
FFT	*Fast Fourier Transform*
FMP	*Fast Matching Pursuit*
FPU	*Float Point Unit*
GUI	*Graphical User Interface*
HIESSD	*Harmonics and Interharmonics components Estimation based on Signal Sparse Decomposition*
LM	*Levenberg-Marquardt*
MP	*Matching Pursuit*
MPM	*Matrix Pencil Method*
NM	*Nelder-Mead*
POO	Programação Orientada a Objetos
pu	*por unidade*
QN-BFGS	*Quasi-Newton Broyden-Fletcher-Goldfarb-Shannon*
RAM	*Random Access Memory*
RSI	Rotinas de Serviço De Interrupção
RTOS	Sistemas Operacionais de Tempo Real

LISTA DE SÍMBOLOS

LETRAS LATINAS

$\boldsymbol{\eta}$	Ruído gaussiano
$\boldsymbol{\phi}$	Equação do átomo
$\boldsymbol{r}$	Resíduo do sinal
$\boldsymbol{x}$	Sinal reconstruído
$\boldsymbol{y}$	Representação do sinal
ϵ	Tolerância
γ	Conjunto de parâmetros do átomo
ω	Frequência angular do átomo
π	Constante circular
θ	Fase do átomo
A	Amplitude do átomo
Au	Amplitude ajustada do átomo
f	Frequência fundamental
h	Harmônico
K	Número total de átomos
k	Número da iteração atual
l	Índice da frequência de entrada
n	Número de amostras do sinal
W_N	Fator angular

SUMÁRIO

1 INTRODUÇÃO

Com a projeção de crescimento do consumo de energia elétrica nas próximas décadas, equipamentos de eletrônica de potência serão mais utilizados, tanto na geração de energia elétrica quanto pelo consumidor final (BRASIL, MME/EPE, 2020). Embora esses dispositivos sejam eficientes no uso da energia, acabam gerando distúrbios na rede elétrica, devido a esse problema, mais estudos estão surgindo nas áreas de engenharia elétrica e qualidade de energia (TEIXEIRA, 2009).

Entre os distúrbios associados a qualidade de energia podem ser destacados a variação, flutuação e desequilíbrio de tensão, cintilação (*flicker*), variações de frequência e distorções harmônicas (BALTAZAR, 2007). No caso das distorções harmônicas, esses distúrbios podem ser classificados como harmônicas (componentes de frequência múltiplas inteiras da frequência fundamental, no Brasil, de 60 Hz) e inter-harmônicas (componentes múltiplas não inteiras da componente fundamental) (HANZELKA; BIEŃ, 2004).

As principais fontes de harmônicas e inter-harmônicas são as cargas do tipo não lineares conectadas à rede elétrica, isso é, todo conjunto de equipamentos e elementos elétricos que não obedecem à lei de Ohm, como reguladores de tensão, fontes de alimentação chaveadas, conversores de frequência, motores e transformadores em saturação (OLIVEIRA, 2015).

A presença de harmônicos no sistema elétrico podem causar perdas, sobrecargas e aquecimento, em capacitores, transformadores e motores, além de gerar interferência em sistemas de comunicação (SANTOSO *et al.*, 2004). Enquanto as inter-harmônicas geram efeitos térmicos em condutores, flutuação de tensão, oscilações de baixa frequência em sistemas mecânicos, cintilação (*flicker*) em lâmpadas fluorescentes e equipamentos eletrônicos, interferência em sinais de controle e proteção de linhas de transmissão e sobrecarga de filtros passivos (HANZELKA; BIEŃ, 2004).

Devido aos problemas citados, tornou-se necessário desenvolver métodos para medir corretamente componentes de frequência harmônicas e inter-harmônicas, possibilitando maior velocidade e confiabilidade no controle do sistema elétrico, reduzindo efeitos adversos na energia entregue ao consumidor.

Alguns dos métodos encontrados na literatura são a Transformada Discreta de Fourier (*Discrete Fourier Transform* (DFT)), o *Matrix Pencil Method* (MPM) (HUA; SARKAR, 1988), o *Fast Matching Pursuit* (FMP) (CHEN *et al.*, 2017), o *Harmonics and Interharmonics components Estimation based on Signal Sparse Decomposition* (HIESSD) (PRADO; BARROS; GUARNERI, 2019), entre outros. Atualmente, a medição de harmônicos e inter-harmônicos no sistema elétrico é definida pela a norma internacional IEC 61000-4-7 (IEC, 2006) que determina uma resolução de 5 Hz (200 ms) para a medição de componentes de frequência, podendo trazer algumas restrições para a estimação dessas ondas.

No método da DFT, ao estimar componentes inter-harmônicas em uma janela de 200 ms quando o sinal apresenta *desvio de frequência*, isso é, a frequência do sistema elétrico não

esta perfeitamente ajustada em seu valor padrão (no Brasil, 60 Hz) pode ocorrer na medição um efeito chamado *vazamento espectral* (TESTA *et al.*, 2007). No vazamento espectral, a energia associada às componentes não múltiplas de 5 Hz do sinal original podem ser adicionadas nas componentes múltiplas da frequência fundamental do sistema medido, resultando em um erro de medição pela DFT (ROCHA, 2016).

O algoritmo MPM supera o vazamento espectral e consegue estimar harmônicos e inter-harmônicos, com robustez ao ruído de medição. Apesar dessas qualidades, o MPM é método que exige alta carga computacional (PRADO; BARROS; GUARNERI, 2019). No caso do método FMP, um sinal qualquer pode ter componentes harmônicas e inter-harmônicas estimadas pela decomposição em componentes senoidais e atualização das frequências e fases durante o processamento, inclusive sendo robusto em relação ao desvio de frequência. Porém, com leituras na presença de ruído, componentes não existentes podem ser estimadas (PRADO; BARROS; GUARNERI, 2019). O algoritmo HIESSD possui algumas das técnicas utilizadas no FMP, conseguindo estimar, além da frequência e fase, a amplitude das harmônicas e inter-harmônicas e sendo ainda robusto a ruído. Porém, não estima corretamente as componentes de frequência quando o sinal apresenta desvio de frequência (BARROS; GUARNERI; LAZZARETTI, 2021).

Devido às desvantagens dos métodos citados, foi proposto recentemente um novo algoritmo chamado *Anti-Leakage Matching Pursuit* (ALMP) obtendo melhores resultados de estimação e tempo de execução. Esse método possibilita estimar amplitude, frequência e fase de componentes harmônicas e inter-harmônicas, por meio de um método de resolução de *mínimos quadrados não lineares* em cada interação, sendo esse o principal estagio do ALMP (BARROS; GUARNERI; LAZZARETTI, 2021).

Entre os algoritmos de resolução de *mínimos quadrados não lineares* são utilizados comumente os métodos de *Levenberg-Marquardt* (LM) (LOURAKIS *et al.*, 2005), *Nelder-Mead* (NM) segundo Singer e Nelder (2009) e *Quasi-Newton Broyden-Fletcher-Goldfarb-Shannon* (QN-BFGS) (NOCEDAL; WRIGHT, 2006). Anteriormente, esses três algoritmos foram implementados e testados no laço principal do ALMP, embora os erros percentuais para estimação de amplitude, frequência e fase de harmônicas e inter-harmônicas não foram significativamente diferente entre eles, o método de *Levenberg-Marquardt* apresentou menor tempo de processamento sob as mesmas condições (BARROS; GUARNERI; LAZZARETTI, 2021).

Em geral, o ALMP possui baixo erro de estimação, na presença de ruído, robustez ao desvio de frequência e apresenta baixo tempo de processamento para componentes harmônicas e inter-harmônicas comparado aos demais algoritmos de estimação (BARROS; GUARNERI; LAZZARETTI, 2021).

Visto que o algoritmo *Anti-Leakage Matching Pursuit* foi implementado e testado previamente apenas em ambientes computacionais, utilizando MATLAB e Python, com sinais oriundos de fontes específicas causadoras de harmônicos, como fonte de alimentação contínua monofásica, conversor de seis pulsos, máquina síncrona e sinais extraídos de uma micro-rede composta por fontes solares (BARROS; GUARNERI; LAZZARETTI, 2021). O projeto desenvol-

vido neste trabalho buscou desenvolver um protótipo de medidor de componentes harmônicas e inter-harmônicas em tempo real baseado no algoritmo *Anti-Leakage Matching Pursuit*, utilizando um *kit* de desenvolvimento para processamento de sinais e sistemas embarcados, no caso o modelo STM32F746G-DISCO (STMICROELECTRONICS, 2019) com a apresentação dos resultados no *display* gráfico do próprio *kit*.

1.1 Objetivos

Nesta seção são apresentados o objetivo geral e os objetivos específicos desenvolvidos desse projeto.

1.1.1 Objetivo geral

O objetivo geral desse trabalho é projetar um protótipo de medição para componentes harmônicas e inter-harmônicas baseado no algoritmo *Anti-Leakage Matching Pursuit*.

1.1.2 Objetivos específicos

- Revisar a literatura acerca do algoritmo *Anti-Leakage Matching Pursuit*.
- Pesquisar e instalar bibliotecas em linguagem C com o algoritmo de *Levenberg-Marquardt*, para resolução de problemas de mínimos quadrados não lineares.
- Implementar e validar o funcionamento do algoritmo ALMP no *kit* de desenvolvimento.
- Adequar a entrada do conversor analógico-digital (*Analog-to-digital converter* (ADC)) do *kit* de desenvolvimento para recepção de sinais externos e integração ao laço de estimação do ALMP.
- Elaborar uma interface gráfica para a apresentação dos resultados no *display* matricial do *kit* de desenvolvimento.
- Testar o protótipo, analisar os resultados e validar o sistema com sinais compostos de componentes harmônicas e inter-harmônicas previamente conhecidos.

1.2 Estrutura do trabalho

Este trabalho esta organizado em 5 capítulos, incluindo esta introdução. O Capítulo 2 expõe o referencial teórico sobre o funcionamento dos algoritmos *Anti-Leakage Matching Pursuit* e de *Levenberg-Marquardt*, e conceitos sobre sistemas operacionais de tempo real utilizado em microcontroladores. No Capítulo 3 são descritos os materiais e a metodologia utilizada para

construção do protótipo de medidor. O Capítulo 4 demonstra os resultados e discussões obtidas com os testes e validações do protótipo. Finalmente o Capítulo 5 apresenta as conclusões obtidas a partir do desenvolvimento deste trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a fundamentação teórica necessária para o desenvolvimento deste trabalho. Nesta seção serão apresentados conceitos referentes aos modelos de ondas harmônicas e inter-harmônicas, aos algoritmos de medição aplicados nesse projeto e aos sistemas operacionais de tempo real utilizados em microcontroladores.

2.1 Modelo de Sinal Harmônico

A modelagem de sinais formados por ondas harmônicas e inter-harmônicas, a partir do método *Matching Pursuit* (MP) (MALLAT; ZHANG, 1993), considera que esses sinais são composto pelo somatório de ondas mais simples, também chamados de *átomos*, podendo ser representados por senoides ou cossenoides, como na equação

$$\boldsymbol{\phi}_{\gamma}(n) = A\cos\left(\omega n + \theta\right), \tag{1}$$

em que $\boldsymbol{\phi}_{\gamma}(\cdot)$ representa a equação do átomo, γ o conjunto de parâmetros de interesse do átomo [A, ω, θ], sendo A a amplitude, ω a frequência angular, θ a fase e n a variável de amostragem (1, 2, ..., N) sendo N o número total de amostras. Assim, um sinal genérico formado por componentes harmônicas e inter-harmônicas pode ser expresso pelo somatório de um número limitado de átomos, como apresentado por

$$\boldsymbol{y}(n) = \sum_{k=1}^{K} \boldsymbol{\phi}_{\gamma_k}(n) + \boldsymbol{\eta}(n), \tag{2}$$

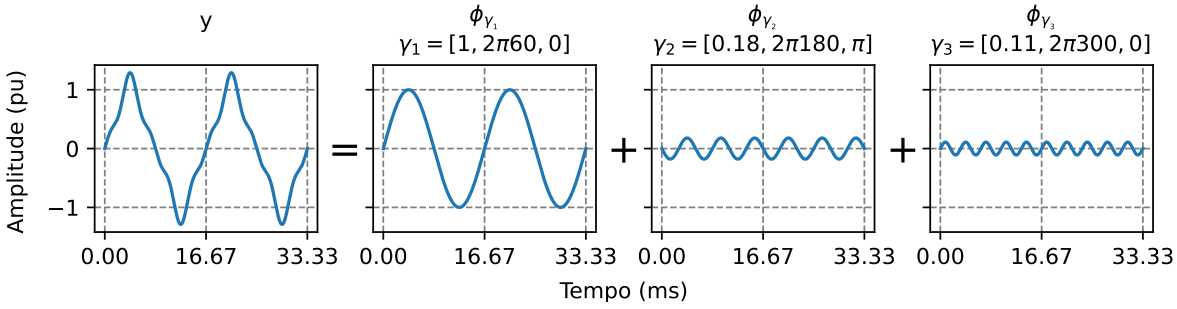
em que $\boldsymbol{y}(n)$ corresponde a um sinal genérico com N amostras, $\boldsymbol{\phi}_{\gamma}(n)$ ao átomo, como na Equação (1), $\boldsymbol{\eta}(n)$ ao ruído gaussiano e k ao número de átomos, que varia de 1 a K átomos que constituem o sinal. Visto que o sinal está limitado a um total de K átomos, não é necessário conhecer previamente a quantidade de componentes harmônicas. Dessa forma, para processar um sinal harmônico será necessário K iterações para estimar cada um dos átomos harmônicos ou inter-harmônicos por um algoritmo iterativo (BARROS; GUARNERI; LAZZARETTI, 2021).

Para ilustrar o modelo acima, a Figura 1 apresenta o sinal harmônico (y) da corrente de um transformador sobre-excitado, composto de três átomos, sendo eles $\boldsymbol{\phi}_{\gamma_1}$ com parâmetros $\gamma_1 = [1,\, 2\pi 60,\, 0]$, $\boldsymbol{\phi}_{\gamma_2}$ com $\gamma_2 = [0{,}18,\, 2\pi 180,\, \pi]$ e $\boldsymbol{\phi}_{\gamma_3}$ com $\gamma_3 = [0{,}11,\, 2\pi 300,\, 0]$.

2.2 Anti-Leakage Matching Pursuit

O ALMP é um algoritmo iterativo baseado no método MP que consegue estimar componentes harmônicas e inter-harmônicas de sinais elétricos do sistema de potência com baixo erro (BARROS; GUARNERI; LAZZARETTI, 2021). O diferencial desse algoritmo se deve ao

Figura 1 – Representação de sinal composto de átomos



Fonte: Modificado de (BARROS; GUARNERI; LAZZARETTI, 2021).

uso do método de mínimos quadrados não lineares para encontrar corretamente os parâmetros γ de cada átomo presente em um sinal de entrada. Esse trecho do algoritmo é caracterizado como etapa de otimização, onde os parâmetros γ são estimados pela resolução do seguinte problema:

$$\hat{\boldsymbol{\gamma}}_k = \underset{\gamma}{\operatorname{argmin}} \left\| \boldsymbol{r}_k - \boldsymbol{\phi}_\gamma \right\|^2. \tag{3}$$

O termo $\hat{\boldsymbol{\gamma}}_k$ equivale ao conjunto de parâmetros ideais $[\hat{A}_k, \hat{\omega}_k, \hat{\theta}_k]$, sendo eles a amplitude, a frequência angular e a fase do átomo da iteração k, o $\boldsymbol{\phi}_\gamma$ é o átomo, como apresentado na Equação (1) e $\boldsymbol{r}_k$ representa o *resíduo*, que na inicialização do ALMP é uma cópia do sinal de entrada e ao final de cada iteração é removido o último átomo encontrado.

O problema da Equação (3) representa a busca pelos parâmetros $\hat{\boldsymbol{\gamma}}_k$ que minimizam a energia do resíduo da iteração k, isso é, os parâmetros γ que tornam mínima a diferença entre o resíduo $\boldsymbol{r}_k$ e o átomo $\boldsymbol{\phi}_\gamma$, ou simplesmente, aqueles que mais aproximem a forma de onda do átomo a do resíduo da atual iteração k. A etapa de otimização, realizada pelo método de mínimos quadrados não lineares, representa a principal etapa do ALMP e garante a robustez ao desvio de frequência e ao ruído, extraídos diretamente do resíduo atual, até que o resíduo seja mínimo, sobrando apenas ruído (BARROS; GUARNERI; LAZZARETTI, 2021).

Neste trabalho foi utilizado o algoritmo de Levenberg-Marquardt (LM) amplamente usado para resolução de mínimos quadrados não lineares. Comparado a outros algoritmos semelhantes como Nelder-Mead (NM) e Quasi-Newton Broyden-Fletcher-Goldfarb-Shannon (QN-BFGS), o LM apresentou menor tempo de execução para estimação de ondas harmônicas e interharmônicas apesar de todos os algoritmos alcançaram resultados equivalentes sob as mesmas condições para os mesmos sinais de entrada (BARROS; GUARNERI; LAZZARETTI, 2021).

No início de cada iteração, o ALMP fornece parâmetros iniciais para o LM com o objetivo de melhorar a eficiência de estimação dos parâmetros $\hat{\boldsymbol{\gamma}}_k$ durante a etapa de otimização. Esses parâmetros iniciais são calculados por meio da *norma infinito* da Transformada Rápida de Fourier (ou, do inglês, *Fast Fourier Transform* (FFT)) do sinal do resíduo da iteração k, como apresentado pela equação

$$\boldsymbol{\phi}_{\gamma_{0_k}} = \left\| \frac{1}{N} \sum_{n=0}^{N-1} \boldsymbol{r}_k W_N^{ln} \right\|_{\infty}, \tag{4}$$

em que $\boldsymbol{\phi}_{\gamma_{0_k}}$ corresponde ao átomo inicial da iteração k, $\boldsymbol{\gamma_{0_k}}$ ao conjunto de parâmetros [A_{0_k}, ω_{0_k}, θ_{0_k}] sendo eles, a amplitude, a frequência angular e a fase que serão usadas como ponto de partida para o algoritmo de otimização, N ao número de amostras, $W_N = e^{j2\pi/N}$ e $l = 1, 2, ..., N-1$.

A Equação (4) estima as componentes de frequência presentes no sinal do resíduo com a FFT e realiza a *norma infinita* ($\|\cdot\|_{\infty}$) dessa estimação a fim de encontrar a componente de maior amplitude, sua frequência angular e a fase associada a essa amplitude, sendo esses os parâmetros que $\boldsymbol{\gamma_{0_k}}$ recebe. Desse modo, além da Equação (4) agilizar o processamento durante a etapa de otimização, visto que a onda de maior amplitude é mais característica no sinal do resíduo, também garante que a ordem de mensuração dos átomos tenha amplitude decrescente a cada iteração, isso é, que na primeira iteração seja estimada a componente fundamental e na última iteração uma onda próximo ao ruído.

Logo depois da etapa de otimização, quando os parâmetros $\hat{\boldsymbol{\gamma}}_k$ estão definidos, ocorre a atualização do resíduo com a remoção do átomo atual, como apresentado em

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{\phi}_{\hat{\gamma}_k}. \tag{5}$$

Assim como a atualização do *sinal reconstruído* $\boldsymbol{x}_k$, sendo esse, um sinal que na inicialização do algoritmo é nulo e conforme ocorrem as iterações recebe os átomos da iteração atual, como representado por

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{\phi}_{\hat{\gamma}_k}. \tag{6}$$

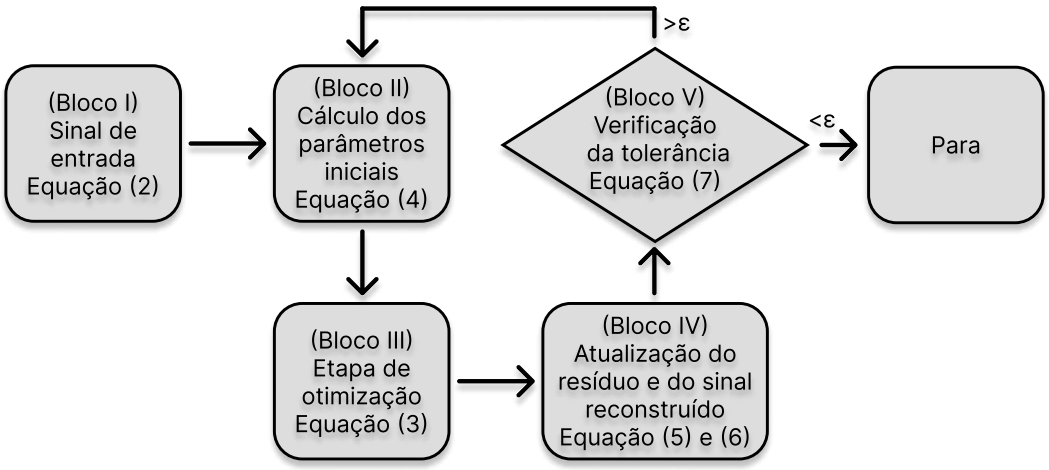
A medida que ocorrem as iterações, o resíduo $\boldsymbol{r}_k$ decresce se aproximando do ruído $\boldsymbol{\eta}$, enquanto o sinal reconstruído $\boldsymbol{x}_k$ cresce em amplitude se aproximando do sinal de entrada $\boldsymbol{y}$. No final das iterações, quando $k = K$, $\boldsymbol{r} \approx \boldsymbol{\eta}$ e $\boldsymbol{x} \approx \boldsymbol{y}$, o ALMP considera que o processo chegou ao fim assim que a energia do sinal do resíduo para de diminuir, em outros termo, a diferença entre o resíduo atual e o resíduo atualizado seja menor que uma tolerância predefinida ϵ, conforme a condição

$$\sqrt{\sum_{n=0}^{N-1} |\boldsymbol{r}_k|^2} - \sqrt{\sum_{n=0}^{N-1} |\boldsymbol{r}_{k+1}|^2} \leq \epsilon. \tag{7}$$

A ordem de execução do algoritmo ALMP, conforme apresentado no fluxograma da Figura 2, inicia com a aquisição do sinal de entrada (2), a partir dele é feito o cálculo dos parâmetros iniciais do resíduo pela FFT (4), na sequência, com os parâmetros iniciais e o resíduo, são estimados os parâmetros ótimos pelo algoritmo LM na etapa de otimização (3). O quarto passo é atualização do resíduo (5) e do sinal reconstruído (6), em seguida, verificado se a to-

lerância (7) foi alcançada, se sim, o algoritmo é encerrado, caso não, novos parâmetros iniciais são calculados com o sinal do resíduo pela a FFT (4).

Figura 2 – Sequência de execução do algoritmo ALMP



Fonte: Modificado de (BARROS; GUARNERI; LAZZARETTI, 2021).

2.3 Sistemas Operacionais de Tempo Real

Os Sistemas Operacionais de Tempo Real (RTOS) são uma subclasse de sistemas operacionais, geralmente voltados para sistemas embarcados que utilizam microcontroladores. A principal característica desses sistemas operacionais é que o tempo de resposta a eventos é definido e projetado para que os prazos sejam sempre cumpridos, em casos em que esses prazo são extrapolados, falhas no sistema podem ocorrer (DENARDIN; BARRIQUELLO, 2019).

Geralmente sistemas embarcados eram programados usando os conceitos de sistemas *foreground*/*background*, nos quais o projeto era executado dentro de um laço infinito (*background*) e quando necessário tratar eventos síncronos ou assíncronos eram geradas interrupções, tratando esses eventos fora do laço infinito, dentro de Rotinas de Serviço De Interrupção (RSI) (*foreground*). Nos sistemas *foreground*/*background* quando há muitas funções sendo realizadas e um ou mais eventos levem muito tempo para ser completamente executados, as demais operações do sistema são atrasadas podendo fazer o projeto não funcionar corretamente.

Diferente dos sistemas *foreground*/*background*, os sistemas operacionais de tempo real são projetados para executar dentro de tarefas (ou *threads*) durante *marcas de tempo*. As tarefas são sequências de instruções projetadas para realizar uma determinada função ou atividade, funcionando como sistemas *backgrounds* individuais, em que cada tarefa é executada concorrentemente, disputando acesso ao processador (*Central Processing Unit* (CPU)). O termo *marca de tempo* refere-se a menor unidade de tempo reconhecida pelo sistema operacional, que normalmente varia de 10 a 100 ms e é gerada por uma interrupção de estouro de tempo

por *hardware*. Nos RTOS, é ideal que um conjunto de tarefas sempre completem suas execuções dentro dessas marcas de tempo, mesmo que nem sempre isso seja possível (DENARDIN; BARRIQUELLO, 2019).

Cada tarefa possui pilha e contexto salvos na memória *Random Access Memory* (RAM). A pilha contém as variáveis locais e informações de execução da tarefa, e o contexto é formado pelo valor dos registradores do processador, indicando o estado de execução da tarefa. Quando a tarefa perde acesso a CPU, seu contexto é salvo e ao receber de volta, a execução continua de onde parou. Durante a instalação das tarefas é definido seu endereço inicial na memória RAM, nome, prioridade de execução e tamanho da pilha.

Nos sistemas operacionais, o gerenciamento do processador, memória, tempo, recursos do sistema, bem como o escalonamento das tarefas que ganham acesso ao processador, é feito pelo núcleo (*kernel*). Os núcleos dos RTOS podem ser do tipo não preemptivo ou preemptivo. Nos núcleos não preemptivos, as tarefas precisam desistir do controle do processador para que outra tarefa possa ganhar acesso. Já nos RTOS de núcleo preemptivo, uma tarefa pode perder o acesso à CPU involuntariamente a qualquer momento, baseado na prioridade das tarefas prontas para receber acesso ao processador.

Para que as tarefas sejam executadas dentro dos prazos, o núcleo possui um mecanismo de seleção e ordem de execução de quais tarefas receberão acesso à CPU. Esse mecanismo é chamado de escalonador, que utiliza um algoritmo de seleção de escalonamento, podendo ser dirigido por tempo ou por prioridades. Além do escalonamento de tarefas, o *kernel* do RTOS também fornece serviços e objetos para coordenar a sincronização e comunicação entre tarefas, como os semáforos, *mutex*, caixas e filas de mensagens, entre outros.

Os semáforos são mecanismos de sincronização de atividades de tarefas que cooperam entre si. Quando uma das tarefas *toma* um semáforo, as demais tarefas que precisam do mesmo semáforo são suspensas até que o semáforo seja *liberado*. O *mutex* é um objeto do núcleo para sincronização de recursos disputados entre tarefas. O *mutex* é derivado dos semáforos, porém o *mutex* só é liberado pela tarefa que o tomou, diferente dos semáforos, em que qualquer tarefa pode liberá-los.

As caixas e filas de mensagens são objetos de comunicação, permitindo a sinalização e transferência de dados entre tarefas. As caixas de mensagens são filas para transferência de um único dado entre tarefas, podendo ser utilizado para sinalização e sincronização de tarefas. Já as filas de mensagens são filas que permitem reter mensagens temporariamente em um *buffer* até o destinatário recebê-las. Não sendo necessário a tarefa destinatária receber a mensagem para transmitir a próxima (DENARDIN; BARRIQUELLO, 2019).

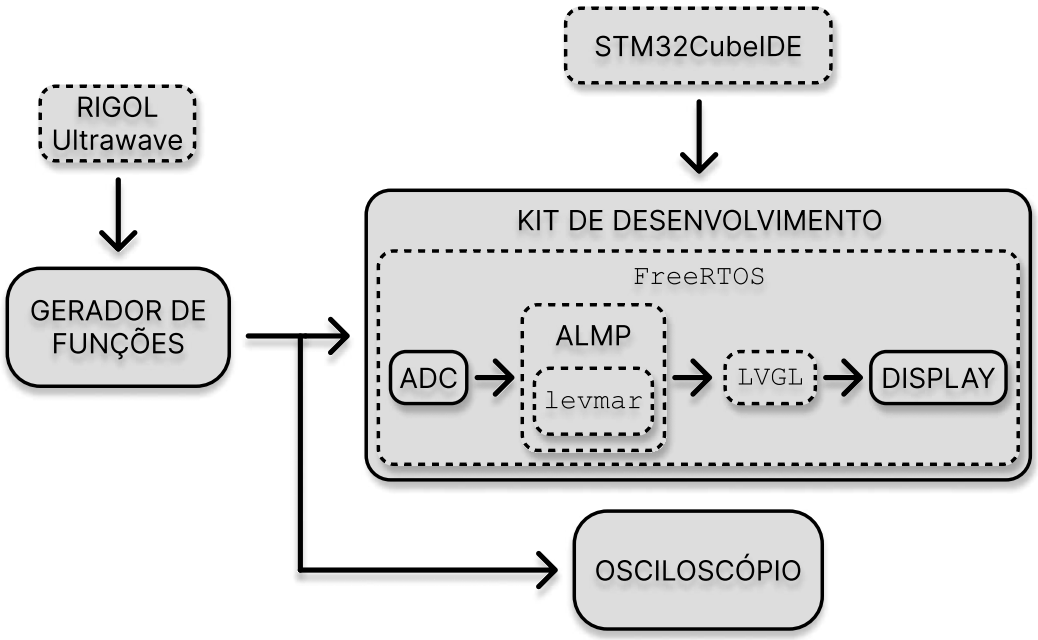
Esses objetos são utilizados quando uma ou mais tarefas precisam fazer um acesso seguro a um mesmo recurso, variável ou periférico do microcontrolador que são compartilhados entre elas. Isso evita que os dados sejam embaralhados, conflitos de acesso de dispositivos ocorram ou que falhas aconteçam impedindo o funcionamento do sistema, uma vez que uma tarefa pode ser interrompida durante o uso desses recursos.

Em resumo, os sistemas operacionais de tempo real dão a impressão de que todas as atividades estão acontecendo ao mesmo tempo, diminuem a complexidade de desenvolvimento de um projeto em tarefas individuais para cada problema, executam primeiro as tarefas mais importantes em detrimento das de menor prioridade e possibilitam a sincronização ou comunicação das tarefas quando recursos são compartilhados.

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os materiais e os métodos necessários para realização deste trabalho, a Figura 3 ilustra os itens usados e como se interagem no sistema, destacando que os itens com bordas pontilhadas representam *softwares* ou bibliotecas de código e os itens com bordas contínuas se referem a materiais físicos.

Figura 3 – Materiais utilizados para o desenvolvimento do protótipo



Fonte: Autoria própria.

3.1 Materiais

Essa seção descreve os materiais utilizados para o desenvolvimento desse projeto, o proposito de cada item usado e suas principais características.

3.1.1 Kit de desenvolvimento

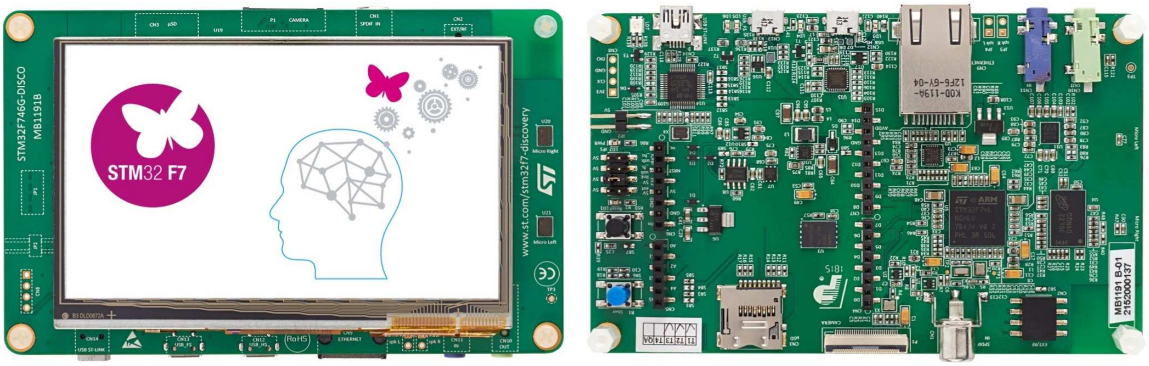
A aquisição dos sinais harmônicos e ou inter-harmônicos, a estimação de cada um dos átomos presente nesses sinais de entrada e a apresentação dos resultados, sendo esses as amplitudes, frequências e fase de cada átomo são realizados pelo *kit* de desenvolvimento.

O *kit* de desenvolvimento utilizado foi o STM32F746G-Disco, equipado com o microcontrolador STM32F746NGH6 da STMicroelectronics que possui um processador (CPU) ARM Cortex-M7 com barramento de 32 *bits*, *clock* máximo de 216 MHz, unidade de ponto flutuante (*Float Point Unit* (FPU)) para cálculo envolvendo representações de números reais, além de con-

junto de instruções dedicadas para processamento de sinais (*Digital Signal Processor* (DSP)), o que torna ideal para esse projeto (STMICROELECTRONICS, 2016).

O microcontrolador possui internamente memória RAM de 320 KB e *flash* de 1024 KB, três conversores analógicos digitais (ADC), cada um com 15 canais, de resolução de 12 bits (4096 níveis de tensão) e tensão de referência de 3,3 V, 14 *timers* de proposito geral, um controlador de acesso direto à memória (*Direct Memory Access* (DMA)) de 16 canais, além de outros periféricos. Na placa do *kit* há também um *display* do tipo LCD-TFT de tamanho 4,3" (polegadas), com resolução de 480 por 272 *pixels* e *touch screen* de tipo capacitivo (STMICROELECTRONICS, 2019), como mostrado na Figura 4.

Figura 4 – *Kit* de desenvolvimento STM32F746G-Disco



(a) Frente **(b) Verso**

Fonte: st.com.

3.1.2 Gerador de funções

A avaliação do funcionamento desse projeto requer sinais harmônicos e inter-harmônicos compostos por átomos de parâmetros (A, ω, θ) conhecidos, e que após a estimação, os parâmetros encontrados correspondam aos desses sinais de entrada.

Os sinais de entrada podem ser reproduzidos por um gerador de funções. Nesse trabalho, foi usado o modelo RIGOL DG1022, como mostrado na Figura 5, que possui dois canais de saída e é capaz de gerar diversas formas de onda, como as funções seno, quadrada, rampa, pulso e ruído, além de outras 48 funções arbitrárias e funções editáveis por *software* para computador.

Esse gerador de funções pode operar com frequências de operação de 1 μHz até 20 MHz dependendo da forma de onda gerada, amplitude de tensão entre 2 mVpp e 10 Vpp no o canal 1 e de 2 mVpp a 3 Vpp no o canal 2 e também operar com um *offset* para o canal 1 de até 5 V para as cargas de baixa impedância e 10 V para as de alta impedância e no canal 2 de até 1,5 V para as cargas de baixa impedância e 3 V para as de alta (RIGOL TECHNOLOGIES, INC, 2015).

Figura 5 – RIGOL DG1022

Fonte: rigolna.com.

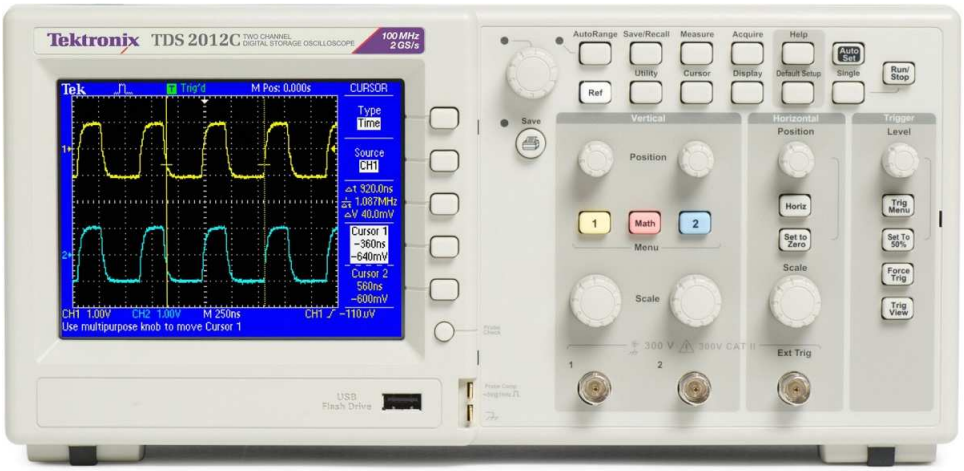
3.1.3 Osciloscópio

Para verificar se os sinais criados pelo gerador de funções estão corretos, assim como os átomos que compõem esses sinais, é necessário o uso de um osciloscópio. O osciloscópio utilizado foi o Tektronix TDS2012C apresentado na Figura 6.

O modelo em questão possui dois canais de entrada, uma largura de banda de 100 MHz, sendo essa a frequência máxima em que não haverá uma atenuação considerável na amplitude, uma taxa de amostragem de 2 GS/s e cursores para aferição manual nos eixos da amplitude e tempo.

Inclui uma tela LCD-TFT de resolução 320 por 240 pixels, com 5,7" de tamanho. O osciloscópio possui também uma porta USB para salvar e exportar rapidamente as leituras ou os dados apresentados na tela (TEKTRONIX, INC, 2019).

Figura 6 – Tektronix TDS2012C



Fonte: tek.com.

3.1.4 STM32CubeIDE

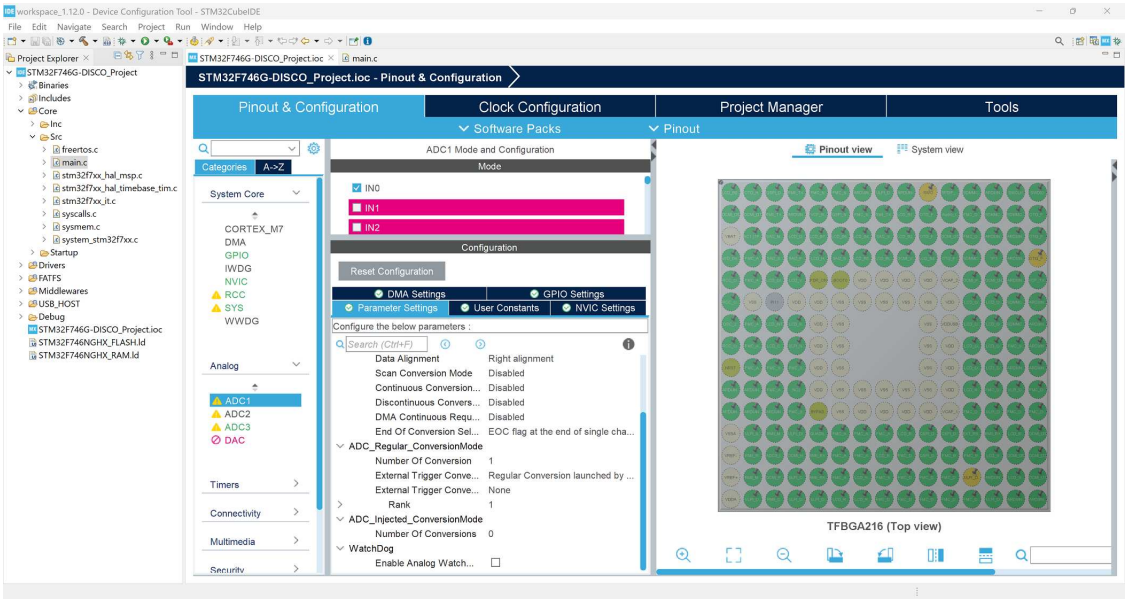
O desenvolvimento do *firmware* do microcontrolador, a configuração dos seus periféricos e inclusão de bibliotecas externas é feita pelo *software* de computador STM32CubeIDE. Essa ferramenta é uma plataforma de desenvolvimento C/C++ dedicada para a geração de código, compilação e depuração de microcontroladores STM32 (STMICROELECTRONICS, 2021).

O *software* fornece uma interface gráfica para configuração dos periféricos, do sistema de *clock*, e outras funções, antes e durante o desenvolvimento do *firmware* (STMICROELECTRONICS, 2023), como mostrado na configuração do ADC1 da Figura 7.

Disponibiliza também o conjunto de bibliotecas CMSIS otimizadas para microcontroladores de arquitetura ARM, incluindo funções para processamento de sinais, redes neurais, sistemas operacionais de tempo real, entre outras.

O STM32CubeIDE inclui também analisadores de compilação e pilha para avaliação do *status* e dos requisitos de memória do projeto, além de recursos de depuração para acompanhar a sequência de execução do código e monitoramento de variáveis.

Figura 7 – STM32CubeIDE



Fonte: Autoria própria.

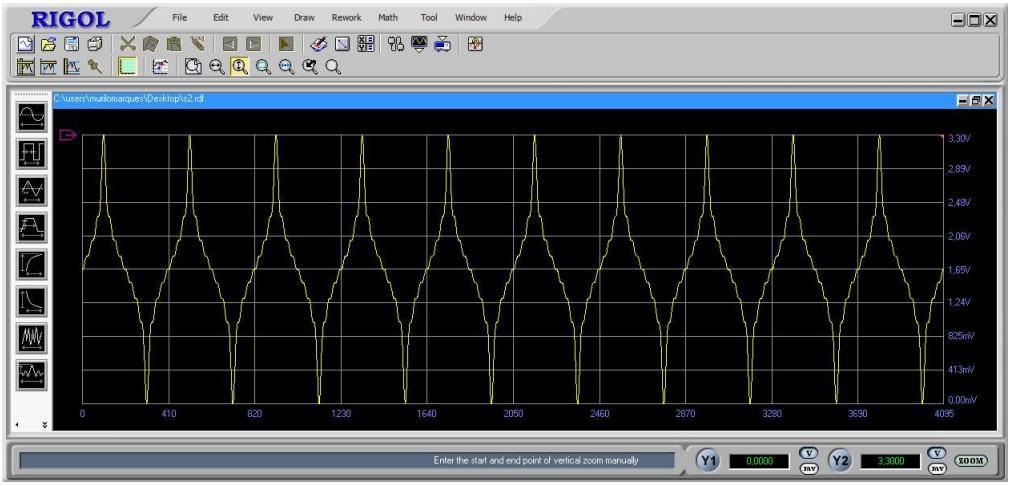
3.1.5 RIGOL Ultrawave

A criação de sinais harmônicos e inter-harmônicos, desde os átomos até a onda final para reprodução na saída do gerador de funções é realizada pelo *software* RIGOL Ultrawave. Essa ferramenta foi desenvolvida pela RIGOL para criação e modificação de sinais arbitrários para reprodução nos geradores de função da família DG1000 (RIGOL TECHNOLOGIES, INC, 2015).

O programa permite a criação de ondas numa janela de até 20 V de pico a pico no eixo da amplitude e 4095 pontos (12 *bits*) no eixo do tempo, o que equivale a um ciclo de repetição do sinal. As funções que compõem o Ultrawave para geração de sinais arbitrários são: seno, quadrada, rampa, pulso, exponencial de subida ou descida, ruído, sinc (seno cardinal), e degrau contínuo.

O *software* possui diversas funcionalidades, destacando as funções matemáticas de adição, subtração e multiplicação entre ondas, o que possibilita a soma de senoides diversas na criação de sinais harmônicos e inter-harmônicos, como o sinal de uma fonte de corrente contínua monofásica da Figura 8.

Figura 8 – RIGOL Ultrawave



Fonte: Autoria própria.

3.1.6 Sistema Operacional `FreeRTOS`

A ordem de execução, o gerenciamento de recursos e de tempo das atividades realizadas pelo microcontrolador deste projeto são feitos por um sistema operacional de tempo real (RTOS). Neste trabalho foi utilizado o `FreeRTOS`, um RTOS de licença MIT, escrita no padrão MISRA C, compatível com mais de 40 arquiteturas de microcontroladores, sendo um dos mais utilizados do mercado (AMAZON WEB SERVICES, INC., 2023).

O núcleo do `FreeRTOS` ocupa cerca de 12 kB, e pode alocar mais memória dependendo do número de tarefas e objetos utilizados em um projeto. Esse sistema operacional fornece diversas funções, objetos e serviços, entre elas, funções para gerenciamento de tarefas (*threads*), semáforos, filas, temporizados, interrupções, alocação de memória, entre outras.

3.1.7 Biblioteca `levmar`

A etapa de otimização do ALMP consiste na resolução de mínimos quadrados não lineares, neste trabalho, realizada pelo algoritmo de Levenberg-Marquardt (LM). A implantação do algoritmo LM no microcontrolador foi feita por meio da biblioteca `levmar`, de licença GPL e escrita em ANSI C/C++ (LOURAKIS, 2011).

A biblioteca `levmar` fornece diversas funções para resolução problemas de otimização utilizando o método de LM, podendo ser divididas em duas categorias, funções sem restrição (*unconstrained*) e com restrição (*constrained*). As sub-rotinas com restrição podem limitar os parâmetros da equação a ser otimizada por meio de equações lineares, inequações e limites de caixa, sendo essa última a restrição das variáveis em intervalos de valores definidos.

As funções podem ser de precisão dupla (64 *bits*) ou simples (32 *bits*) e executadas para otimização por meio de jacobiano analítico ou por diferenças finitas. No caso das sub-rotinas que usam o jacobiano analítico é preciso fornecer as derivadas da equação de otimização em relação a cada um dos parâmetros a serem otimizados. Já no caso das funções que utilizam diferenças finitas, as derivadas são aproximadas numericamente durante a execução da função.

Para exemplificar uma função presente na biblioteca, a função `dlevmar_bc_der()` é uma função de precisão dupla (prefixo `d`) que possui restrição de caixa (componente `_bc`) e utiliza jacobiano analítico (sufixo `_der`). Independente da função, a biblioteca `levmar` precisa receber o sinal medido, o número de amostras do sinal, a equação de otimização, neste trabalho a equação do átomo, a quantidade de parâmetros a serem otimizados, o valor dos parâmetros iniciais e o número máximo de iterações.

3.1.8 Biblioteca `LVGL`

Os parâmetros dos átomos harmônicos e inter-harmônicos são apresentados, durante e após o processo de estimação, no *display* do *kit* de desenvolvimento. A organização das informações apresentadas na tela é realizada pela biblioteca *Light and Versatile Graphics Library* (`LVGL`), sob a licença MIT e escrita em C99 compatível com C++ e MicroPython (LVGL KFT., 2023b).

A biblioteca `LVGL` é dedicada para o desenvolvimento de interfaces gráficas de usuário (*Graphical User Interface* (GUI)). Em aplicações feitas para microcontroladores é recomendado pelo menos 180 kB de memória *flash* e 24 kB de memória RAM, variando dependendo da resolução e da quantidade de elementos gráficos usados.

A biblioteca foi desenvolvida utilizando os conceitos de Programação Orientada a Objetos (POO) e disponibiliza diversos objetos ou elementos gráficos, também chamados de *widgets*, como botões, *labels*, *charts*, listas, menus, entre outros. Visto que se trata de POO, todos os *widgets* possuem atributos, como coordenadas de posição, tamanho, hierarquia de camada (objeto pai e filho), estilos, entre outras propriedades.

A biblioteca fornece também vários outros recursos, como rolagem de objetos, resposta a eventos e *timers*, *layouts* de tela, animações, entre outros recursos que ajudam a melhorar o *design* de GUIs. Neste trabalho foram utilizados os *widgets tabview*, *flexbox labels* e *charts* e a função *timer handler*.

A *tabview* é um *widgets* usado para organizar informações em guias e é composto de outros objetos como a área de conteúdo e botões para troca de abas. A *flexbox* é um objeto de organização, que cria *layouts* flexíveis, isso é, ajusta o espaçamento interno dinamicamente, aumenta ou diminui de tamanho conforme a quantidade de elementos que estão dentro de si.

As *labels* são objetos básicos da biblioteca usados para exibir textos, numéricos ou caracteres especiais no *display*. As *charts* são *widgets* usadas para criar e apresentar gráficos a partir de dados numéricos. Já a função *timer handler* é usada para atualizar *widgets* ou recursos do *display* em períodos ajustáveis.

No *site* oficial da biblioteca é disponibilizada uma vasta documentação, incluindo exemplos de cada *widgets* e recurso, além do *blog* e do fórum onde frequentemente são divulgados projetos com aplicações reais que utilizam essa biblioteca (LVGL KFT., 2023a).

3.2 Métodos

Essa seção apresenta os métodos utilizados no desenvolvimento deste trabalho e suas subseções, apesar de apresentar uma ordem lógica de desenvolvimento, não foram executadas necessariamente na ordem apresentada.

3.2.1 Configuração de periféricos

O ponto de partida para o desenvolvimento desse projeto é a configuração dos periféricos do microcontrolador. Nesse projeto foram configurados dois ADCs, um *timer* e uma unidade de acesso direto a memória (DMA), além do sistema de *clock* do microcontrolador com o cristal externo do *kit* de desenvolvimento. A configuração pode ser feita no *software* STM32CubeIDE, tanto pela interface gráfica, própria para ajuste dos periféricos, quanto por linhas de texto no código fonte.

O *clock* do sistema foi ajustado para a frequência máxima de 216 MHz, mas dependendo do barramento, o *clock* do sistema passa por divisões, fazendo com que os periféricos trabalhem em frequências menores. Já os ADCs foram configurados para receber os sinais em apenas um dos canais de entrada por ADC, com resolução de 12 *bits* e programados para fazer apenas uma conversão por vez a cada requisição do DMA.

Considerando que a frequência da rede de elétrica é 60 Hz e que foi definida uma taxa de 256 amostras por ciclo do sinal, a frequência de amostragem resultante é 15360 Hz (256 amostras * 60 Hz = 15360 Hz). A norma IEC 61000-4-7 (IEC, 2006) determina uma janela de

medição de harmônicos de 5 Hz. Assim, a quantidade total de amostras em uma janela de medição é 3072 (15360 Hz / 5 Hz = 3072 amostras).

Dessa maneira, o *timer* foi utilizado para ser gatilho do início de conversão dos ADCs, foi configurado para fazer contagens no modo de subida, como a frequência do *timer* é a metade do *clock* do sistema, o período definido foi de 7031 contagens ((216 MHz / 2) / 15360 Hz = 7031 contagens). Já o DMA ficou responsável por transferir os valores das conversões dos canais do ADC direto para a memória, poupando carga computacional do CPU. Sendo configurado no modo circular, assim, quando o *buffer* de dados estiver completamente preenchido com 3072 amostras por ADC, ele é reinicializado, sem pausa, na primeira posição do *buffer*.

O DMA foi programado também para executar em multi-modo triplo, em que são coletadas simultaneamente as conversões dos três canais dos ADCs. Dessa forma não há atraso de conversões para o mesmo ADC e defasagem entre os diferentes ADCs. Apesar de serem utilizados dois ADCs, foram ativados três, uma vez que não é possível utilizar o multi-modo duplo pois o *kit* de desenvolvimento disponibiliza apenas um pino para um canal do ADC1 e outro do ADC3, sendo necessário habilitar o ADC2 para fazer as leituras dos canais ADC1 e ADC3 juntos.

3.2.2 Utilização do sistema operacional de tempo real

O próximo passo foi a implementação do RTOS ao projeto do *kit* de desenvolvimento. Apesar do STM32CubeIDE disponibilizar o `FreeRTOS` modificado da biblioteca CMSIS, foi importado e compilado o `FreeRTOS` direto do *site* oficial para a arquitetura do microcontrolador utilizado no projeto do STM32CubeIDE.

Para o protótipo deste trabalho foi necessário apenas duas *threads*, uma fila e quatro semáforos para sincronização entre as trocas de informações entre as tarefas. A primeira tarefa, `ALMP_Task()`, ficou responsável pela aquisição dos sinais dos ADCs e pelo processamento do algoritmo ALMP. Já a segunda tarefa, `LVGL_Task()`, ficou responsável pela ativação do *display* com a biblioteca `LVGL` e pela atualização da GUI.

A fila foi utilizada para informa a `ALMP_Task()` que o *buffer* do DMA foi completamente preenchido pela função `HAL_ADC_ConvCpltCallback()`, ou seja, o ADC1 e ADC3 terminaram as conversões do sinal em uma janela de 200 ms, como recomendado na norma IEC 61000-4-7, permitindo o inicio de execução do ALMP.

Já os semáforos foram utilizados para compartilhar e sincronizar os parâmetros dos átomos entre as duas *threads*, durante a execução do ALMP e a atualização da interface gráfica. A cada iteração, semáforos são tomados pela tarefa `ALMP_Task()` garantindo que o átomo atual seja estimado e que o resíduo e o sinal reconstruído sejam atualizados, antes que a tarefa `LVGL_Task()` atualize com os valores do átomo estimado no *display*.

3.2.3 Desenvolvimento da interface gráfica

Outra etapa foi a instalação da biblioteca `LVGL` e o desenvolvimento de uma interface gráfica para a apresentação das ondas harmônicas e inter-harmônicas estimadas. A biblioteca `LVGL` foi importada do *site* oficial e configurada para funcionar adequadamente com o `FreeRTOS` e com o *kit* de desenvolvimento, incluindo a ativação de outros periféricos do microcontrolador para ativação do *display*.

Como mencionado anteriormente, para o desenvolvimento da interface gráfica foram utilizados os *widgets tabview*, *flexbox*, *chart* e *label*, e a função *timer handler*. Foram desenvolvidas duas *tabviews* para exibição dos parâmetros dos átomos estimados e do sinal de entrada.

A primeira *tabview*, chamada de `Detalhado`, apresenta cada átomo estimado textualmente por meio de *labels*. Nesta guia, foram criadas duas *flexbox*, uma destinada a exibir as estimações de um canal de um dos ADCs, e a segunda *flexbox* para o outro canal. Dentro das *flexbox* são apresentadas as *labels*, numerando cada átomo encontrado e seus parâmetros (A, ω, θ). Para esta *tabview* também é utilizado um *timer handler*, onde um semáforo do RTOS é tomado, atualizando as *labels* com o ultimo átomo encontrado, trabalhando paralelamente com a tarefa `ALMP_Task()`.

A segunda guia, `Tempo`, apresenta o sinal original no domínio do tempo por meio de uma *chart* de linha. A maior parte do código do *layout* e da *chart* desta *tabview* foi feita utilizando exemplos do *site* oficial da biblioteca `LVGL`. As modificações no código foram feitas para receber e exibir as 3072 amostras dos sinais dos ADCs, e também utilizar o *timer handler* para atualizar o *display* a cada sinal de entrada recebido.

3.2.4 Implementação da ALMP

A principal parte do desenvolvimento deste trabalho foi a implementação do algoritmo ALMP no *kit* de desenvolvimento. Para isso, foi necessário instalar o algoritmo de Levenberg-Marquardt (LM) pela biblioteca `levmar` e a Transformada de Rápida de Fourier (FFT) pelas funções da biblioteca DSP da interface CMSIS para a arquitetura do microcontrolador usado.

A FFT foi usada para estimar os valores aproximados das ondas harmônicas e inter-harmônicas presente no sinal do resíduo e delas extraídos os parâmetros da componente de maior amplitude ($\boldsymbol{\gamma}_{\mathbf{0}_k}$) que serão usados como parâmetros iniciais no algoritmo LM. Enquanto o LM tem o objetivo de estimar os parâmetros ideais da onda de maior amplitude presente no resíduo.

Neste projeto foi utilizada a função `slevmar_der()`, da biblioteca `levmar`, para a otimização utilizando o método de LM, uma função de precisão simples, sem restrição, que utiliza o jacobiano analítico. Por utilizar jacobiano analítico, além da equação do átomo ($\phi_\gamma(\cdot)$), foi necessário fornecer também as derivadas da equação do átomo em relação a cada um dos parâmetros de interesse (A, ω, θ) como mostrado por

Listagem 1 – Aquisição do sinal de entrada

```
// Percorre o vetor de conversões do ADC de 3 em 3 posições
// em que o número de amostras N_MEASUREMENTS = 3072
for(uint16_t adc_idx=0; adc_idx<=3*(N_MEASUREMENTS - 1); adc_idx+=3) {

    // Utiliza o sinal do Canal  do ADC1
    if(input_Channel == 0)
        signal_Residue[signal_Indx] = ADC_Buffer[adc_idx];
    // Utiliza o sinal do Canal do ADC3
    else if (input_Channel == 1)
        signal_Residue[signal_Indx] = ADC_Buffer[adc_idx + 2];

    // Converte o sinal da base binária para a base por unidade (pu)
    // em que 0.000244200244 = 1/4095
    // Converte de 0b a 4095b para 0pu a 1pu
    signal_Residue[signal_Indx] *= 0.000244200244;
    // Desloca de 0pu a 1pu para -0.5pu a 0.5pu
    signal_Residue[signal_Indx] -= 0.5;
    // Amplia de -0.5pu a 0.5pu para -1pu a 1pu
    signal_Residue[signal_Indx] *= 2.0;

    // Calcula a energia do sinal de entrada
    sum_sq_Input += signal_Residue[signal_Indx]*signal_Residue[signal_Indx];
    // Anula o sinal reconstruído
    signal_Reconst[signal_Indx] = 0.0;
    // Preenche o vetor do sinal que será usado na FFT
    signal_FFT[signal_Indx] = signal_Residue[signal_Indx];

    // Anula todos os parametros dos átomos a serem estimados
    if(signal_Indx<N_ITER){
        almp_atom_A[signal_Indx] = 0.0;
        almp_atom_F[signal_Indx] = 0.0;
        almp_atom_P[signal_Indx] = 0.0;
    }

    // Incrementa em uma posição o indice do sinal
    signal_Indx++;
}
```

Fonte: Autoria própria.

$$\begin{aligned}
\frac{d\phi_\gamma(n)}{dA} &= \cos(\omega n + \theta), \\
\frac{d\phi_\gamma(n)}{d\omega} &= -An\sin(\omega n + \theta), \\
\frac{d\phi_\gamma(n)}{d\theta} &= -A\sin(\omega n + \theta).
\end{aligned} \tag{8}$$

Uma vez instalada a FFT e o `levmar`, foi implementado o restante da estrutura do ALMP, como será apresentado nas listagens a seguir. A Listagem 1 apresenta o trecho do código para aquisição do sinal de entrada, como apresentado no Bloco I da Figura 2.

Já a Listagem 2 exibe o trecho do projeto em que são calculados os parâmetros iniciais do sinal com a FFT antes de utilizá-las no algoritmo LM, equivalente ao Bloco II da Figura 2. Enquanto a Listagem 3 ilustra o trecho do código que busca os parâmetros otimizados do sinal pelo algoritmo de LM, como apresentado no Bloco III da Figura 2.

A Listagem 4 apresenta o trecho do algoritmo que atualiza o sinal do resíduo e do sinal reconstruído com o sinal da última componente de frequência estimada, e calcula a energia do átomo, do resíduo e do sinal reconstruído atual, conforme o Bloco IV da Figura 2. Já a

Listagem 2 – Cálculo dos parâmetros iniciais

```
// Processa a FFT com vetor signal_FFT
arm_rfft_fast_f32(&RFFT_Handle, signal_FFT, signal_FFT, 0);

// Limpa as variáveis auxiliares FFT_out_Indx e FFT_out_Max
FFT_out_Indx = 0;
FFT_out_Max = 0.0;

// Percorre o vetor signal_FFT processado pela FFT
// em que o tamanho da amostra da FFT é N_FFT = 2048
for (uint16_t FFT_idx=0; FFT_idx<(uint16_t)(N_FFT/2-1); FFT_idx++) {

    // Adquire a parte real e imaginaria da FFT
    FFT_out_Real[FFT_idx] = signal_FFT[FFT_idx*2];
    FFT_out_Imag[FFT_idx] = signal_FFT[(FFT_idx*2)+1];

    // Calcula a magnitude ao quadrado da FFT
    FFT_out_Mag = FFT_out_Real[FFT_idx]*FFT_out_Real[FFT_idx] +
                  FFT_out_Imag[FFT_idx]*FFT_out_Imag[FFT_idx];

    // Busca e salva a maior magnitude ao quadrado e seu indice
    if(FFT_out_Max < FFT_out_Mag){
        FFT_out_Max = FFT_out_Mag;
        FFT_out_Indx = FFT_idx;
    }
}

// Calcula o modulo da maior magnitude da FFT
arm_sqrt_f32(FFT_out_Max, &FFT_out_Max);

// Adquire os parametros do átomo inicial
// em que o inverso do tamanho da amostra da FFT é N_1_FFT = 1/2048
// e a taxa de amostragem do sinal é CONS_FS_AM = 15360
// Amplitude
lm_params[0] = 2.0*FFT_out_Max*(float32_t)N_1_FFT;
// Frequência
lm_params[1] = 2.0*(float32_t)M_PI*(float32_t)FFT_out_Indx*CONS_FS_AM*(float32_t)N_1_FFT;
// Fase
lm_params[2] = atan2f(FFT_out_Imag[FFT_out_Indx], FFT_out_Real[FFT_out_Indx]);
```

Fonte: Autoria própria.

Listagem 5 exibe o laço de execução do algoritmo ALMP destacando o trecho de verificação da tolerância do algoritmo, assim como mencionado no Bloco V da Figura 2, porém com algumas alterações em relação a Equação 7.

O critério da tolerância foi modificado para o algoritmo executar enquanto

$$\begin{gathered}\frac{\sum_{n=0}^{N-1}\left|\boldsymbol{r}_k\right|^2}{\sum_{n=0}^{N-1}\left|\boldsymbol{y}\right|^2}>\epsilon, \\ \frac{\sum_{n=0}^{N-1}\left|\boldsymbol{\phi}_{\hat{\gamma}_k}\right|^2}{\sum_{n=0}^{N-1}\left|\boldsymbol{y}\right|^2}>\epsilon, \\ k<M\end{gathered} \tag{9}$$

Listagem 3 – Etapa de otimização

```
// Realiza a otimização dos parametros pelo algoritmo de Levenberg-Marquardt
slevmar_der(lm_func_cos_dif, lm_jacb_cos_der, lm_params, signal_Residue, N_PARAMS,
            N_MEASUREMENTS, N_ITER, lm_opts, lm_info, lm_work_Buffer, NULL, NULL);

// Guarda os parametros otimizados do LM
// Amplitude em pu, em que S_MAGNITUDE = 1
// Frequência angular em Hertz, em que M_1_PI = 1/3.1415926535
// Fase em graus
almp_atom_A[almp_Iter] = S_MAGNITUDE*lm_params[0];
almp_atom_F[almp_Iter] = lm_params[1]*0.5*M_1_PI;
almp_atom_P[almp_Iter] = lm_params[2]*180.0*M_1_PI;

// Atualiza fase negativa em positiva adicionando 360 graus
while(almp_atom_P[almp_Iter] < 0) almp_atom_P[almp_Iter] += 360.0;
```

Fonte: Autoria própria.

Listagem 4 – Atualização dos sinais

```
// Limpa energia do átomo, do resíduo e do sinal reconstruído
sum_sq_Atom = 0.0;
sum_sq_Residue = 0.0;
sum_sq_Reconst = 0.0;

// Atualiza os sinais e calcula a energia
// em que a quantidade de amostras N_MEASUREMENTS = 3072
for(uint16_t almp_Update=0; almp_Update<(uint16_t)N_MEASUREMENTS;
    almp_Update++){

    // Constrói o sinal do átomo atual
    signal_Atom[almp_Update] = lm_params[0]*
        arm_cos_f32(lm_params[1]*float32_t)CONS_T_AM*almp_Update + lm_params[2]);
    // Atualiza o resíduo, removendo o átomo atual
    signal_Residue[almp_Update] -= signal_Atom[almp_Update];
    // Atualiza o sinal reconstruído, adicionando o átomo atual
    signal_Reconst[almp_Update] += signal_Atom[almp_Update];
    // Atualiza o sinal para a FFT com o resíduo
    signal_FFT[almp_Update] = signal_Residue[almp_Update];

    // Calcula a energia do átomo, do resíduo e do sinal reconstruído atual
    sum_sq_Atom += signal_Atom[almp_Update] * signal_Atom[almp_Update];
    sum_sq_Residue += signal_Residue[almp_Update] * signal_Residue[almp_Update];
    sum_sq_Reconst += signal_Reconst[almp_Update] * signal_Reconst[almp_Update];
}
```

Fonte: Autoria própria.

em que $\boldsymbol{r}_k$ é o resíduo da atual iteração k, $\boldsymbol{\phi}_{\hat{\gamma}_k}$ é o átomo com de parâmetros ótimos da iteração k, $\boldsymbol{y}$ é o sinal de entrada, ϵ é a tolerância ajustada para $\sqrt{2}/100$ e M é o número máximo de iterações do algoritmo ALMP, definido para um limite de 100 iterações. Essa alteração para encerrar a execução do algoritmo foi feita pois considerando a verificação da Equação 7, o algoritmo não chegava ao fim após estimar todos os átomos, e assim, encontrava componentes de frequência de amplitudes baixas que não faziam parte dos sinais gerados. O código completo deste projeto esta disponível para *download* no GitHub e pode ser baixado neste ***link***.

Listagem 5 – Verificação de tolerância

```
// Execução do algoritmo ALMP
do{
    // ... Etapas anteriores do ALMP

    // Adquire a energia do átomo e do resíduo atual
    energy_iter_Atom[almp_Iter] = sum_sq_Atom;
    energy_iter_Residue[almp_Iter] = sum_sq_Residue;

    // Compara a energia do resíduo a tolerancia definida
    if(energy_iter_Residue[almp_Iter]/sum_sq_Input < (float32_t)I_TOL){

        // Compara o átomo atual com o resíduo atual
        if(energy_iter_Atom[almp_Iter] < energy_iter_Residue[almp_Iter]){

            // Limpa os parametros da próxima iteração
            if(almp_Iter < N_ITER){
                almp_atom_A[almp_Iter+1] = 0.0;
                almp_atom_F[almp_Iter+1] = 0.0;
                almp_atom_P[almp_Iter+1] = 0.0;
            }

            // Limpa a energia do resíduo e do átomo atual
            energy_iter_Residue[almp_Iter] = 0.0;
            energy_iter_Atom[almp_Iter] = 0.0;
        }
    }

    // Incremento indice para próxima iteração do ALMP
    almp_Iter++;

// Condições de execução do ALMP
// em que a tolerancia I_TOL = SQRT(2)/100
// e o número máximo de iterações N_ITER = 100
} while(((energy_iter_Residue[almp_Iter-1]/sum_sq_Input)>(float32_t)I_TOL ||
        (energy_iter_Atom[almp_Iter-1]/sum_sq_Input)>(float32_t)I_TOL) &&
        (almp_Iter < (uint32_t)N_ITER));
```

Fonte: Autoria própria.

3.2.5 Geração de sinais

Após realizada todas as etapas anteriores foram gerados sinais harmônicos e interharmônicos para serem estimados pelo ALMP no *kit* de desenvolvimento. A saída dos sinais foi feita pelo gerador de funções RIGOL DG1022, porém só com as formas de ondas disponíveis no gerador, não era possível de gerar sinais harmônicos com mais de um único átomo.

Para criar sinais compostos por diversas componentes de frequência foi necessário utilizar o software RIGOL Ultrawave desenvolvido para o gerador de funções utilizado. Nesse *software* é necessário considerar a janela de 4096 pontos no eixo do tempo e fazer ajustes para obter a frequência ou o número de ciclos de uma onda na saída.

No *software* cada átomo de um sinal harmônico é criado individualmente e depois somados um de cada vez em um novo sinal. Neste trabalho foram recriados seis sinais de fontes harmônicas reais, os sinais de um transformador sobre-excitado, fonte de corrente contínua monofásica, retificador de seis pulsos, cicloconversor, de uma máquina síncrona e uma microrede com desvio de frequência, baseado nos dados utilizados de Barros, Guarneri e Lazzaretti (2021).

Após a criação do sinais gerados com o *software*, foram passados para o gerador de funções, reproduzidas e verificadas no osciloscópio Tektronix TDS2012C se o formato do sinal era semelhante ao esperado. Em seguida os sinais foram ligados aos pinos de entrada do ADC do microcontrolador, para estimação dos átomos que compõem cada um desses sinais.

4 RESULTADOS

Neste capitulo são apresentados os resultados obtidos durante o desenvolvimento deste projeto, a comparação entre os sinais harmônicos e inter-harmônicos gerados e estimados, discussões sobre o desempenho do algoritmo utilizado e a quantidade de memória usada do microcontrolador.

4.1 Estimação de sinais

Após configurar o microcontrolador, instalar o sistema operacional, as bibliotecas, implementar a interface gráfica e o algoritmo ALMP, foram gerados seis sinais de distorções harmônicas e inter-harmônicas para estimação no protótipo como mencionado na subseção 3.2.5. Antes de apresentar os sinais gerados, suas equações, formas de onda e os resultados das estimações será comentado alguns ajustes feitos nas ondas antes de serem recriadas no software Ultrawave adequando a amplitude desses sinais a escala de tensão do ADC do microcontrolador, facilitando a estimação e a visualização desses sinais.

As amplitudes dos sinais são representadas *por unidade* (pu), porém a maioria deles extrapolam a unidade, uma vez que a soma dos valores absolutos das amplitudes de cada átomo resulta em um valor maior que 1 pu. Como a tensão de entrada do ADC do microcontrolador é limitada entre 0 e 3,3 V e cada um dos sinais tem amplitudes máximas diferentes, para padronizar a escala de amplitude, aproveitar toda a faixa de tensão do ADC, aumentar a resolução do sinal e melhorar a visualização dos resultados no *display* todos os seis sinais tiveram suas amplitudes de pico ajustadas para 1 pu, tornando a escala de -1 a 1 pu equivalente a escala de 0 a 3,3 V, centralizando os sinais (0 pu) em 1,65 V.

Antes da criação de cada sinal no *software* Ultrawave, foi calculado sua amplitude máxima com o somatório dos módulos das amplitudes de cada átomo presente em um sinal, em seguida divido a expressão do sinal por esse somatório, limitando a amplitude do sinal em um valor unitário, como expresso na equação

$$\boldsymbol{y}_u(t) = \frac{A_1 \sin(h_1 \omega t + \phi_1) + A_2 \sin(h_2 \omega t + \phi_2) + ... + A_n \sin(h_n \omega t + \phi_n)}{|A_1| + |A_2| + ... + |A_n|}. \tag{10}$$

em que $\boldsymbol{y}_u$ equivale ao sinal de amplitude unitária, A corresponde a amplitude dos átomos, ω a frequência angular que é igual a $2\pi f$, em que f é a frequência fundamental da rede elétrica, h ao harmônico e ϕ a fase de cada átomo.

4.1.1 Transformador sobre-excitado

O primeiro sinal gerado foi o da corrente de um transformador sobre-excitado modelado por Arrillaga e Watson (2004), originalmente apresentado pela equação

$$\boldsymbol{i}_1(t) = \sin(\omega t) - 0{,}18\sin(3\omega t) + 0{,}11\sin(5\omega t), \tag{11}$$

em que ω é igual a $2\pi f$, em que f é a frequência fundamental da rede elétrica, nesse caso 60 Hz. Esse sinal é composto de três componentes de frequência e é classificado como um sinal de distorção exclusivamente harmônica, uma vez que todos os átomos são múltiplos inteiros da frequência fundamental, sendo eles a fundamental, a terceira e a quinta harmônica.

Como já mencionado, ao calcular a amplitude máxima desse sinal com o somatório dos valores absolutos das amplitudes dos átomos é encontrado uma amplitude máxima de 1,29 pu (1 + 0,18 + 0,11 pu). A Tabela 1 apresenta o ajuste na amplitude desse sinal para uma amplitude de pico de até 1 pu, além da conversão para a tensão de entrada do ADC do microcontrolador em *volts*, ainda sem o offset de 1,65 V.

Tabela 1 – Ajuste da amplitude do sinal de corrente de um transformador sobre-excitado

Harmônico h	A (pu)	|A| (pu)	Au=A/(Σ|A|) (pu)	|Au| (pu)	Au*(3,3/2) (V)	|Au*(3,3/2)| (V)
1	1	1	0,7752	0,7752	1,2791	1,2791
3	-0,18	0,18	-0,1395	0,1395	-0,2302	0,2302
5	0,11	0,11	0,0853	0,0853	0,1407	0,1407
Somatório (Σ)		1,29		1,0000		1,6500

Fonte: Autoria própria.

A Tabela 1 exibe as amplitudes (A) de cada átomo do sinal, o modulo dessa amplitude (|A|), a amplitude máxima do sinal original (Σ(|A|)), a amplitude ajustada (Au) para que o sinal tenha uma amplitude máxima de 1 pu (Σ(|Au|)), o modulo da amplitude ajustada (|Au|), a tensão equivalente de cada átomo após o ajuste (Au*(3,3/2)), o modulo da tensão equivalente (|Au*(3,3/2)|) e a amplitude de pico do sinal em *volts* (Σ(|Au*3,3/2|)), expondo que antes do ajuste a amplitude do sinal poderia chegar a 1,29 pu e depois do ajuste a 1 pu e um pico de 1,65 V, totalizando 3,3 V de pico a pico durante a reprodução no gerador de funções.

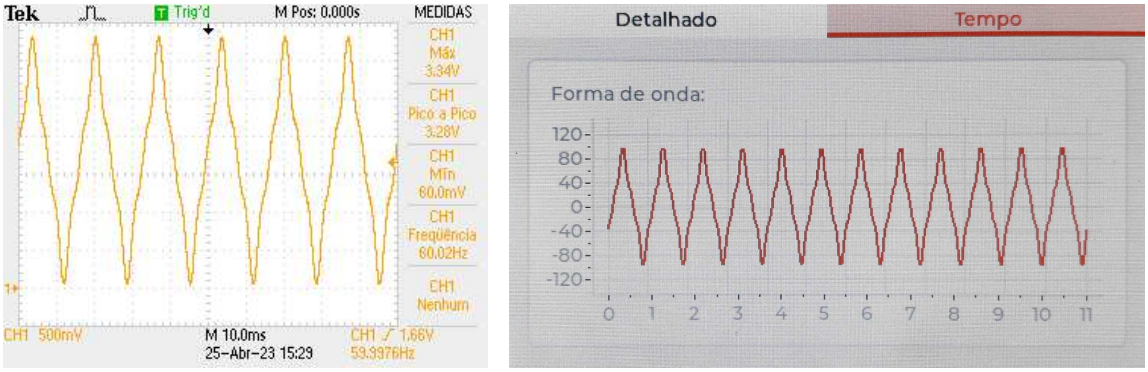
Dessa forma, o sinal de corrente unitário do transformador sobre-excitado também pode ser representado pela equação

$$\boldsymbol{i}_{1u}(t) \approx 0{,}7752\sin(\omega t) - 0{,}1395\sin(3\omega t) + 0{,}0853\sin(5\omega t). \tag{12}$$

A partir dos dados da Tabela 1 foi criado e gerado o sinal de corrente do transformador sobre-excitado, como apresentado nas formas de onda da Figura 9, tanto pelo osciloscópio (9a) quanto pelo *display* do *kit* de desenvolvimento (9b). A Figura 10 ilustra os resultados das esti-

mações do sinal do transformador sobre-excitado, exibindo na interface do *display* a amplitude, frequência e fase dos primeiros átomos desse sinal.

Figura 9 – Forma de onda do sinal de corrente do transformador sobre-excitado

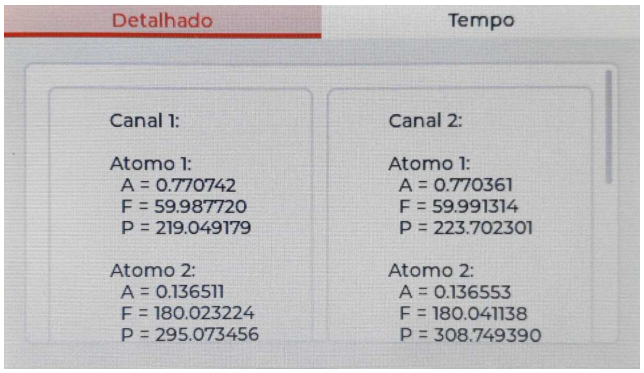


(a) No osciloscópio **(b) No *display* do *kit* de desenvolvimento**

Fonte: Autoria própria.

Para evitar anexar diversas figuras com os resultados das estimações de cada átomo de cada sinal neste documento, foram feitas gravações das estimações e disponibilizadas no GitHub para *download*. O vídeo da estimação do sinal de corrente do transformador sobre-excitado pode ser baixado neste ***link***.

Figura 10 – Parâmetros dos átomos do sinal de corrente do transformador sobre-excitado



Fonte: Autoria própria.

Comparando as formas de onda da Figura 9, é observado que o sinal adquirido pelo ADC do microcontrolador (9b) condiz com o sinal do osciloscópio (9a), apesar da escala de amplitude dessas duas imagens serem diferentes. No sinal do osciloscópio, a amplitude é apresentada em *volts*, equivalente à tensão de 0 a 3,3 V de entrada do ADC do microcontrolador. Já no sinal apresentado no *display*, temos uma amplitude apresentada em percentual em referência à escala *por unidade*, em que 100 equivale a 1 pu (ou 3,3 V na escala de tensão), 0 a 0 pu (ou 1,65 V) e -100 a -1 pu (ou 0 V).

Em relação a verificação dos parâmetros dos átomos estimados na Figura 10 e no vídeo disponibilizado no GitHub, algumas observações podem ser feitas. Uma vez que não foi possível iniciar a captura dos sinais pelo ADC quando a amplitude do sinal está em 0 pu, os valores de

estimação da fase são flutuantes, resultando em valores incorretos para as estimações de fase de quaisquer sinais.

O sinal de corrente do transformador sobre-excitado, como já mencionado, é composto de três componentes de frequência, enquanto que na estimação apresentada no vídeo são encontrados cinco átomos. Com exceção da fase, os átomos 1, 2 e 4 condizem com as componentes de frequência da Equação 12 em amplitude e frequência, porém são estimadas outras duas componentes de frequência não presentes no sinal do transformador sobre-excitado.

O átomo 3, apesar de apresentar uma amplitude alta se comparada as demais, possui frequência nula, isso é, uma componente de tensão contínua, indicando que o ADC do microcontrolador esta recebendo um *offset* em torno de 72 mV (0,0434 pu * 3,3 V / 2 $\approx$ 0,072 mV). Enquanto o átomo 5 possui baixa amplitude se comparado aos demais átomos e apresenta também um frequência múltipla da fundamental, causada por um resto de energia do resíduo, de fonte desconhecida, mas que pode ser desconsiderada.

4.1.2 Fonte de corrente contínua monofásica

O segundo sinal estimado foi o de uma fonte de corrente contínua monofásica (BOLLEN; GU, 2006), conforme a equação

$$\boldsymbol{i}_2(h,t) = \begin{cases} +\frac{2\sqrt{2}}{h}\sin(h\omega t), & h = 1,5,9,13,... \\ -\frac{2\sqrt{2}}{h}\sin(h\omega t), & h = 3,7,11,15,... \\ 0, & h = 0,2,4,6,... \end{cases} \tag{13}$$

Assim como o sinal de corrente do transformador sobre-excitado, esse sinal é composto apenas por componentes de frequência exclusivamente harmônicas e também possui frequência fundamental de 60 Hz. Para criação desse sinal foram considerados os átomos ímpares até o 15º harmônico. A Tabela 2 apresenta o ajuste da amplitude de cada átomo para que o sinal tenha amplitude máxima de 1 pu e a conversão para escala de tensão em *volts*.

Assim, o sinal ajustado para uma amplitude unitária também pode ser representado pela equação

$$\boldsymbol{i}_{2u}(t) \approx \begin{cases} 0{,}4946\sin(\omega t) - 0{,}1649\sin(3\omega t) + 0{,}0989\sin(5\omega t) \\ -0{,}0707\sin(7\omega t) + 0{,}0550\sin(9\omega t) - 0{,}0450\sin(11\omega t) \\ +0{,}0380\sin(13\omega t) - 0{,}0330\sin(15\omega t). \end{cases} \tag{14}$$

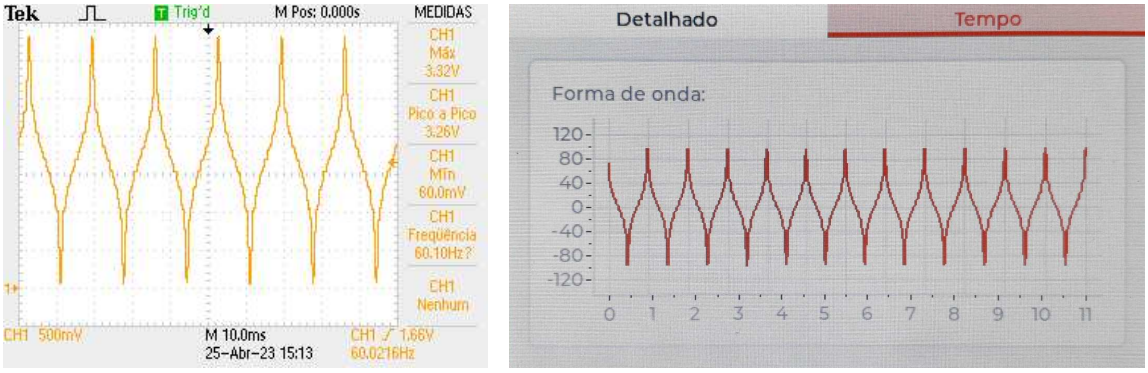
A Figura 11 apresenta a forma de onda da fonte de corrente contínua monofásica, pelo osciloscópio (11a) e pelo *display* do *kit* de desenvolvimento (11b). A Figura 12 ilustra os resultados da estimação no *display*, e o vídeo com a estimação de todos os átomos pode ser acessado neste ***link***.

Tabela 2 – Ajuste da amplitude do sinal da fonte de corrente contínua monofásica

Harmônico h	A (pu)	|A| (pu)	Au=A/(Σ|A|) (pu)	|Au| (pu)	Au*(3,3/2) (V)	|Au*(3,3/2)| (V)
1	2,8284	2,8284	0,4946	0,4946	0,8161	0,8161
5	0,5657	0,5657	0,0989	0,0989	0,1632	0,1632
9	0,3143	0,3143	0,0550	0,0550	0,0907	0,0907
13	0,2176	0,2176	0,0380	0,0380	0,0628	0,0628
3	-0,9428	0,9428	-0,1649	0,1649	-0,2720	0,2720
7	-0,4041	0,4041	-0,0707	0,0707	-0,1166	0,1166
11	-0,2571	0,2571	-0,0450	0,0450	-0,0742	0,0742
15	-0,1886	0,1886	-0,0330	0,0330	-0,0544	0,0544
Somatório (Σ)		5,7185		1,0000		1,6500

Fonte: Autoria própria.

Figura 11 – Forma de onda do sinal da fonte de corrente contínua monofásica



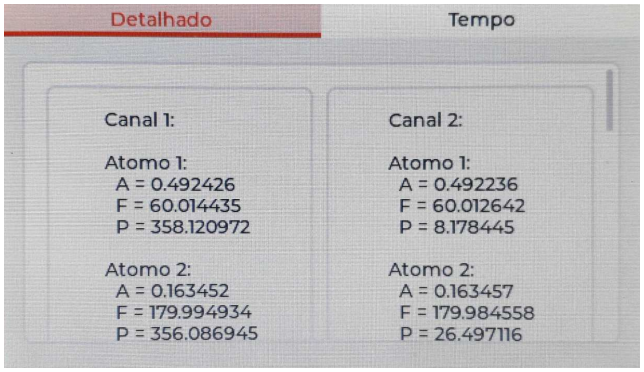
(a) No osciloscópio **(b) No *display* do *kit* de desenvolvimento**

Fonte: Autoria própria.

Como notado a forma de onda apresentada no osciloscópio (11a) é equivalente a exibida no *display* (11b), e ao comparar os parâmetros dos átomos do sinal dessa fonte de corrente contínua, como apresentado na Equação 14, com os parâmetros estimados no vídeo, todas as componentes de frequência esperadas do sinal são estimadas corretamente, com exceção do parâmetro da fase, porém novamente são encontrados outros dois átomos adicionais.

O átomo 4 desse sinal, assim como o átomo 3 da estimação do sinal do transformador sobre-excitado, possui frequência nula e amplitude em torno de 0,0444 pu, indicando que o sinal ou o microcontrolador esta recebendo um *offset* inesperado de 73 mV (0,0444 pu * 3,3 V / 2 $\approx$ 0,073 mV). Já o átomo 10, como possui amplitude baixa (0,0023 pu) e frequência alta (3412 Hz ou 57ª harmônica) se comparado aos demais átomos, pode ser classificada como ruído ou um erro causado por *aliasing* de uma frequência maior no resto da energia do resíduo no sinal estimado.

Figura 12 – Parâmetros dos átomos do sinal da fonte de corrente contínua monofásica



Fonte: Autoria própria.

4.1.3 Retificador de seis pulsos

O terceiro sinal usado e o último puramente harmônico é o sinal de um retificador de seis pulsos (RICE, 1994), como exibido na equação

$$\boldsymbol{i}_3(h) = \frac{1}{h}\sin(h\omega t), \quad \text{em que } h = 6j \pm 1 \text{ e } j = 1, 2, 3, ... \tag{15}$$

Considerando que a frequência fundamental desse sinal, também é 60 Hz, para a geração desse sinal foi considerado todos os harmônicos até o índice $j = 4$, isso é, até o 25º harmônico. A Tabela 3 mostra a conversão da amplitude dos átomos para que o sinal do retificador de seis pulsos tenha amplitude máxima de 1 pu, e apresenta também a conversão dos mesmos átomos para a escala de tensão em *volts*.

Tabela 3 – Ajuste da amplitude do sinal do retificador de seis pulsos

Harmônico h	A (pu)	Au=A/(Σ(A)) (pu)	Au*(3,3/2) (V)
1	1,0000	0,5863	0,9674
5	0,2000	0,1173	0,1935
7	0,1429	0,0838	0,1382
11	0,0909	0,0533	0,0879
13	0,0769	0,0451	0,0744
17	0,0588	0,0345	0,0569
19	0,0526	0,0309	0,0509
23	0,0435	0,0255	0,0421
25	0,0400	0,0235	0,0387
Somatório (Σ)	1,7056	1,0000	1,6500

Fonte: Autoria própria.

Outra forma de representar o sinal nessa nova escala é exibido na equação

$$\boldsymbol{i}_{3u}(t) \approx \begin{cases} 0{,}5863\sin(\omega t) + 0{,}1173\sin(5\omega t) + 0{,}0838\sin(7\omega t) \\ +0{,}0533\sin(11\omega t) + 0{,}0451\sin(13\omega t) + 0{,}0345\sin(17\omega t) \\ +0{,}0309\sin(19\omega t) + 0{,}0255\sin(23\omega t) + 0{,}0235\sin(25\omega t). \end{cases} \tag{16}$$

A Figura 13 apresenta a forma de onda do sinal do retificador de seis pulsos, tanto pelo osciloscópio (13a), quanto pela interface gráfica (13b). Já a Figura 14 ilustra apenas os resultados de estimação dos dois primeiros átomos do sinal no *display* e o vídeo com a estimação de todos os átomos pode ser acessado neste ***link***.

Figura 13 – Forma de onda do sinal de corrente do retificador de seis pulsos

(a) No osciloscópio **(b) No *display* do *kit* de desenvolvimento**

Fonte: Autoria própria.

Figura 14 – Parâmetros dos átomos do sinal de corrente do retificador de seis pulsos

Fonte: Autoria própria.

Como apresentado na Figura 13, a forma de onda do sinal de corrente do retificador de seis pulsos exibida no osciloscópio (13a) se apresenta similar a exposta no *display* (13b). No entanto, ao comparar os parâmetros das componentes de frequência estimadas no vídeo disponibilizado para *download* com os dos átomos da Equação 16, os nove átomos esperados foram estimados corretamente, tirando parâmetro de fase, porém foi encontrado mais uma componente de frequência não esperada.

O átomo 3 exibido no vídeo de estimação do sinal do retificador de seis pulsos, assim como o átomo 3 da estimação do sinal transformador sobre-excitado e átomo 4 do sinal da fonte de corrente contínua monofásica, possui uma amplitude com cerca de 0,0444 pu e frequência nula, apresentando novamente um *offset* de 73 mV.

4.1.4 Cicloconversor

O quarto sinal criado foi o sinal inter-harmônica de corrente de entrada de um cicloconversor modelado Syam *et al.* (2004), representado pela equação

$$\begin{gathered}\boldsymbol{i}_4(t) = 0{,}66\sin(2\pi f_1 t) + 0{,}48\sin(2\pi(2f_1 - 3f_0)t) + 0{,}31\sin(2\pi(4f_1 - 3f_0)t) \\ \text{em que } f_1 = 50{,}75\ Hz \text{ e } f_0 = 35\ Hz,\end{gathered} \tag{17}$$

em que f_1 é a frequência fundamental de 50 Hz com desvio de 0,75 Hz e f_0 é a frequência de saída do sistema. Esse sinal é composto de três átomos, todos inter-harmônicos. A Tabela 4 exibe as amplitudes dos átomos do sinal original, após a conversão para o sinal de amplitude unitária e a conversão referente a escala de tensão de entrada do microcontrolador em *volts*.

Tabela 4 – Ajuste da amplitude do sinal do cicloconversor

Frequência (Hz)	A (pu)	Au=A/(Σ(A)) (pu)	Au*(3,3/2) (V)
50,75	0,66	0,4552	0,7510
3,5	0,48	0,3310	0,5462
98	0,31	0,2138	0,3528
Somatório (Σ)	1,45	1,0000	1,6500

Fonte: Autoria própria.

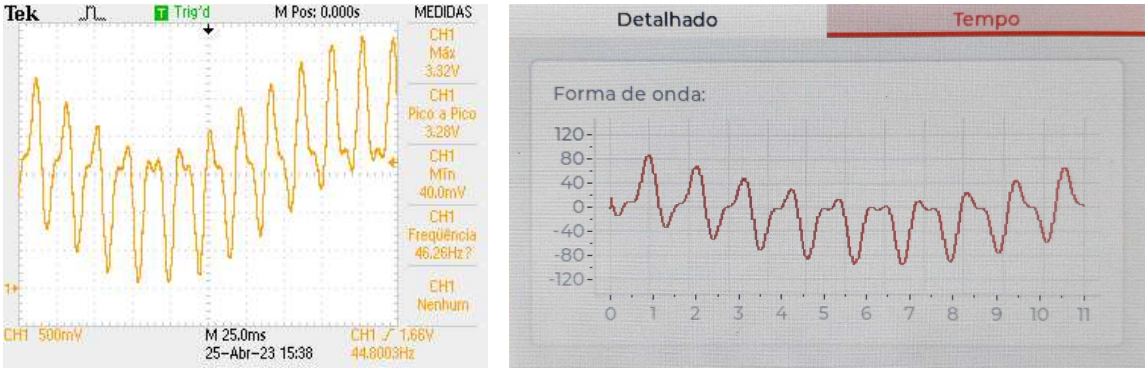
Assim, o sinal ajustado para a amplitude unitária pode ser representado pela equação

$$\boldsymbol{i}_{4u}(t) \approx 0{,}4552\sin(2\pi 50{,}75t) + 0{,}3310\sin(2\pi 3{,}5t) + 0{,}2138\sin(2\pi 98t). \tag{18}$$

A Figura 15 apresenta a forma de onda do sinal de corrente do cicloconversor, pelo osciloscópio (15a) e pela interface gráfica do *display* (15b). A Figura 16 exibe o resultado da estimação dos primeiros átomos desse sinal no *display* e os demais resultados são apresentados no video disponível no GitHub podendo ser acessado por este ***link***.

A comparação entre as formas de onda apresentadas no osciloscópio (15a) e no *display* (15b) mostra que o sinal adquirido pelo microcontrolador é similar ao reproduzido pelo gerador de funções. Porém ao verificar os átomos estimados no vídeo com as componentes de frequên-

Figura 15 – Forma de onda do sinal de corrente do cicloconversor



(a) No osciloscópio **(b) No *display* do *kit* de desenvolvimento**

Fonte: Autoria própria.

Figura 16 – Parâmetros dos átomos do sinal de corrente do cicloconversor

Fonte: Autoria própria.

cia apresentadas na Equação 18, apesar dos três átomos serem encontrados corretamente, mesmo que em ordem diferente nos dois canais, mais de 10 componentes de frequência extras foram estimadas.

No vídeo, em ambos os canais, são expostos apenas 11 componentes de frequência, mas provavelmente foram estimadas dezenas, até o terceiro átomo, as componentes de frequência são equivalentes as esperadas e a partir do átomo 4 são exibidos componentes de frequência inexistentes na criação e geração do sinal. Isso pode ser explicado pela componente de frequência de 3,5 Hz combinada a janela de medição de 5 Hz determinada pela norma IEC 61000-4-7 (IEC, 2006). Como visto no formato da onda apresentado no display (15b) o sinal não completa um ciclo, nem mesmo na imagem do osciloscópio (15a) em uma janela de 25 ms, o sinal tem seu ciclo completado.

Dessa forma, a estimação do átomo de 3,5 Hz apresentou um erro razoável tanto na amplitude quanto na frequência, o que resultou em uma sobra de energia do resíduo durante as iterações do algoritmo, levando a estimar as componentes de frequências posteriores ao terceiro átomo.

4.1.5 Máquina síncrona

O quinto sinal gerado é o sinal inter-harmônico da corrente de uma máquina síncrona (SPYROPOULOS; MITRONIKAS, 2013), como exibido na equação

$$\boldsymbol{i}_5(t) = \begin{cases} 0{,}03\sin(2\pi 24t) + 0{,}024\sin(2\pi 48t) + \\ \sin(2\pi 60t) + 0{,}023\sin(2\pi 96t) + \\ 0{,}029\sin(2\pi 264t) + 0{,}03\sin(2\pi 384t) + \\ 0{,}003\sin(2\pi 588t) + 0{,}004\sin(2\pi 708t), \end{cases} \tag{19}$$

A Tabela 5 mostra a conversão da amplitude dos átomos para que o sinal da máquina síncrona tenha amplitude máxima de 1 pu e exibe também a conversão para a escala de tensão em *volts*.

Tabela 5 – Ajuste da amplitude do sinal da máquina síncrona

Frequência (Hz)	A (pu)	Au=A/(Σ(A)) (pu)	Au*(3,3/2) (V)
60	1	0,8749	1,4436
24	0,03	0,0262	0,0433
384	0,03	0,0262	0,0433
264	0,029	0,0254	0,0419
48	0,024	0,0210	0,0346
96	0,023	0,0201	0,0332
708	0,004	0,0035	0,0058
588	0,003	0,0026	0,0043
Somatório (Σ)	1,143	1,0000	1,6500

Fonte: Autoria própria.

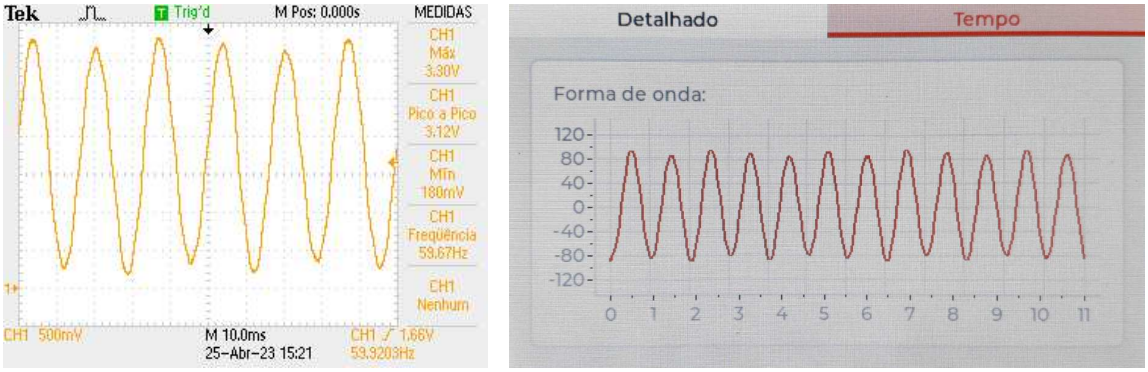
Outra forma de representar o sinal nessa nova escala é exibido na equação

$$\boldsymbol{i}_{5u}(t) \approx \begin{cases} 0{,}8749\sin(2\pi 60t) + 0{,}0262\sin(2\pi 24t) + 0{,}0262\sin(2\pi 384t) \\ +0{,}0254\sin(2\pi 264t) + 0{,}0210\sin(2\pi 48t) + 0{,}0201\sin(2\pi 96t) \\ +0{,}0035\sin(2\pi 708t) + 0{,}0026\sin(2\pi 588t). \end{cases} \tag{20}$$

A Figura 17 apresenta a forma de onda do sinal de corrente da máquina síncrona, tanto pelo osciloscópio (17a), quanto pela interface gráfica do *display* (17b). Já a Figura 18 ilustra os dois primeiros átomos estimados desse sinal, enquanto o vídeo com a estimação de todos os átomos pode ser acessado neste ***link***.

Como apresentado na Figura 17, a forma de onda do sinal de corrente da máquina síncrona exibida no osciloscópio (17a) se apresenta equivalente a exposta no *display* (17b).

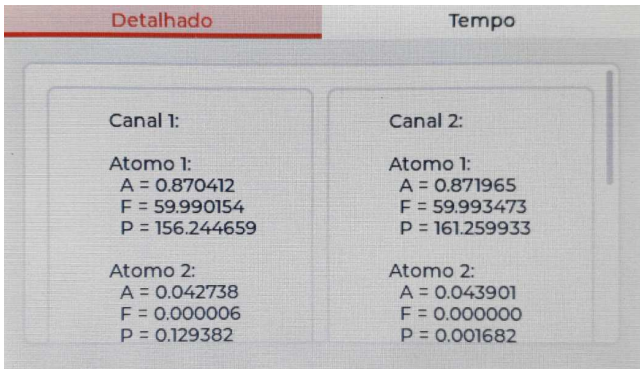
Figura 17 – Forma de onda do sinal de uma máquina síncrona



(a) No osciloscópio **(b) No *display* do *kit* de desenvolvimento**

Fonte: Autoria própria.

Figura 18 – Parâmetros dos átomos do sinal de uma máquina síncrona



Fonte: Autoria própria.

No entanto, ao comparar os parâmetros das componentes de frequência expostas no vídeo com os átomos da Equação 20, foram encontrados cinco átomos, em que apenas quatro deles apresentam parâmetros correspondentes com os dos átomos do modelo da máquina síncrona, desconsiderando o parâmetro da fase.

O átomo 2, apresentado na estimação do sinal de corrente da máquina síncrona, equivale a um dos átomos dos sinais passados, sendo uma componente de frequência de amplitude 0,043 pu e frequência nula, um *offset* de 71 mV. Apesar dos átomos de 60, 24, 384 e 264 Hz terem sido estimados corretamente, as componentes de frequência de 48, 96, 708 e 588 Hz não foram encontradas nessa estimação, uma explicação para isso, pode ser dada pela amplitude muito baixa desses átomos, uma vez que as três componentes de frequência estimadas totalizam cerca de 95% da amplitude de todo sinal, tornando a energia do resíduo irrelevante para a estimação dos próximos átomos, o que leva ao critério de parada do algoritmo.

4.1.6 Micro-rede com desvio de frequência

O sexto sinal estimado foi o sinal de corrente inter-harmônica de uma micro-rede (WANG; YANG; DU, 2012) com desvio de frequência, conforme a equação

$$\boldsymbol{i}_6(t) = \begin{cases} 0{,}0026\cos(2\pi 19{,}92t - 157{,}34) + \cos(2\pi 49{,}8t + 147{,}49) + \\ 0{,}0015\cos(2\pi 229{,}08t + 78{,}94) + 0{,}0302\cos(2\pi 249t + 101{,}91) + \\ 0{,}0245\cos(2\pi 348{,}6t - 145{,}68). \end{cases} \tag{21}$$

A frequência fundamental dessa micro-rede é de 50 Hz e possui desvio de frequência de -0,02 Hz, em que todos os átomos possuem ângulos de defasagem diferentes. A Tabela 6 apresenta o ajuste da amplitude de cada átomo de forma que o sinal tenha amplitude máxima de 1 pu e exibe também a conversão para a escala de tensão em *volts* para o ADC do micro-controlador.

Tabela 6 – Ajuste da amplitude do sinal da micro-rede

Frequência (Hz)	A (pu)	Au=A/(Σ(A)) (pu)	Au*(3,3/2) (V)
19,92	0,0026	0,0025	0,0041
49,8	1,0000	0,9445	1,5584
229,08	0,0015	0,0014	0,0023
249	0,0302	0,0285	0,0471
348,6	0,0245	0,0231	0,0382
Somatório (Σ)	1,0588	1,0000	1,6500

Fonte: Autoria própria.

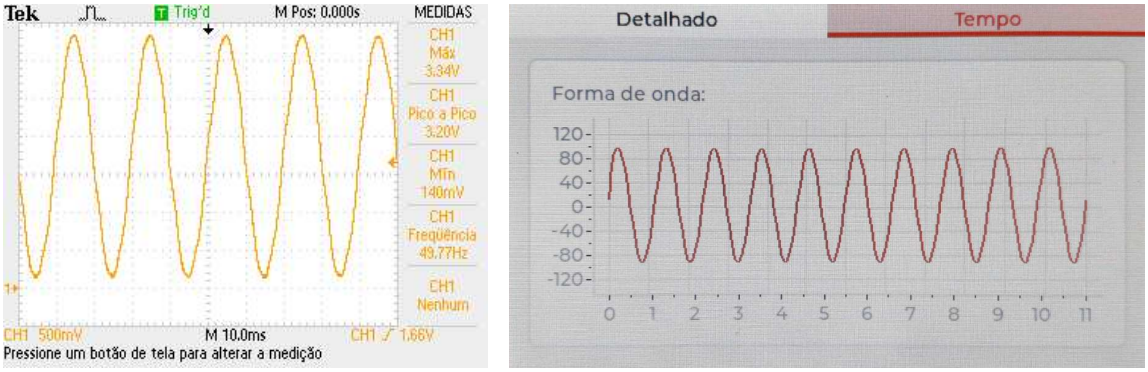
Assim, o sinal ajustado para uma amplitude unitária pode ser representado pela equação

$$\boldsymbol{i}_{6u}(t) \approx \begin{cases} 0{,}0025\cos(2\pi 19{,}92t - 157{,}34) + 0{,}9445\cos(2\pi 49{,}8t + 147{,}49) \\ +0{,}0014\cos(2\pi 229{,}08t + 78{,}94) + 0{,}0285\cos(2\pi 249t + 101{,}91) \\ +0{,}0231\cos(2\pi 348{,}6t - 145{,}68). \end{cases} \tag{22}$$

A Figura 19 apresenta a forma de onda da micro-rede, no osciloscópio (19a) e no *display* do *kit* de desenvolvimento (19b). Enquanto a Figura 20 exibe os resultados da estimação dos parâmetros dos dois primeiros átomos no *display* e o vídeo com a estimação de todos os átomos pode ser acessado nesse ***link***.

Como notado a forma de onda apresentada no osciloscópio (19a) é equivalente a exibida no *display* (19b), porém ao comparar os parâmetros dos átomos da Equação 22 para o

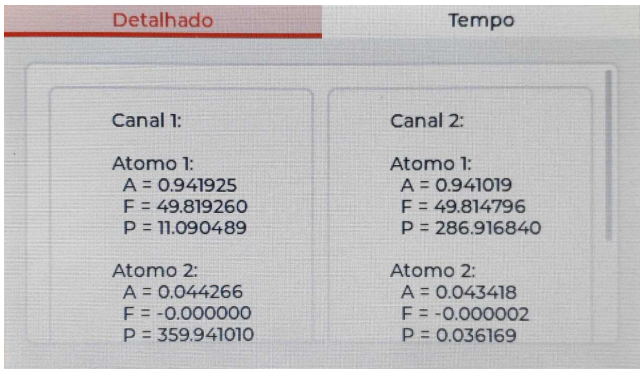
Figura 19 – Forma de onda do sinal de uma micro-rede



(a) No osciloscópio **(b) No *display* do *kit* de desenvolvimento**

Fonte: Autoria própria.

Figura 20 – Parâmetros dos átomos do sinal de uma micro-rede



Fonte: Autoria própria.

sinal dessa micro-rede com os parâmetros estimados no vídeo, apenas três componentes de frequência forma encontradas.

Os átomos 1 e 3 apresentados na estimação, correspondem aos átomos de frequência 49,8 e 249 Hz, respectivamente, sendo estimados como o esperado, porém com fase estimada incorretamente pois o sinal é adquirido pelo ADC em qualquer instante do tempo. O átomo 2, é uma componente de frequência de amplitude 0,044 pu e frequência nula, assim como as estimações anteriores, um átomo com tensão de *offset* de amplitude de 73 mV, podendo ser originado dos equipamentos utilizados, uma vez que esse átomo é apresentado em todos os sinais. Já as demais componentes que não foram estimadas, assim como o sinal da máquina síncrona, possuem amplitudes muito baixas, não tendo impacto na energia do resíduo, sendo ignoradas e encerrando o algoritmo.

4.2 Desempenho do algoritmo levmar

Durante a instalação da biblioteca `levmar` e da implementação do algoritmo ALMP foram testadas as funções `slevmar_der()` e `slevmar_dif()` para resolução de mínimos

quadrados não lineares pelo método de Levenberg-Marquardt. Ambas as funções não possuem restrição e são de precisão simples (32 *bits*) compatível com a arquitetura do microcontrolador usado.

Como mencionado na subseção 3.2.4, o protótipo utiliza a função `slevmar_der()` sendo necessário informar as derivadas da equação de otimização. A função `slevmar_dif()` também foi testada, e não requer o jacobiano da equação do átomo, uma vez que a função realiza diferenças finitas durante a execução do algoritmo, sendo essa a única diferença entre as duas funções.

Usando a `slevmar_der()` foi observado que a função leva até 230 ms para estimar um átomo da Equação 14. E com a função `slevmar_dif()` até 420 ms. O tempo de estimação de um átomo, depende principalmente da diferença entre os valores dos parâmetros iniciais e os parâmetros ideais a serem estimados.

Considerando que a média de tempo de estimação de cada átomo seja de 115 ms para a função `slevmar_der()`, para um sinal com cinco átomos, todos esses átomos levariam em torno de 575 ms para ser estimado. E para a função `slevmar_dif()` o tempo médio de estimação de um átomo seria aproximadamente 210 ms, dessa forma um mesmo sinal com cinco átomos levaria em torno de 1 segundo para ser estimado.

Essa avaliação mostra que a função `slevmar_der()` leva em torno de metade do tempo para estimar um sinal comparado a `slevmar_dif()`. Mas independente da função usada, mesmo com o microcontrolador configurado para trabalhar no clock máximo de 216 MHz e operando com RTOS, um único átomo que leve mais de 100 ms para ser estimado, mostra a alta carga computacional desse algoritmo.

Devido a esse tempo de processamento para um sinal, pode se dizer que a estimação não esta de acordo com a norma (IEC, 2006), uma vez que são perdidas diversas janelas de 5 Hz para a estimação de um sinal. Apesar de conseguir estimar as componentes de frequência harmônicas e inter-harmônicas, a biblioteca `levmar` não foi adequada para estimação de componentes de frequência em tempo real com base nessa normal, pelo menos não com o microcontrolador utilizado.

4.3 Ocupação da memória do microcontrolador

Como comentado na subseção 3.1.1, o microcontrolador STM32F746NGH6 possui 320 kB de memória RAM para dados. E a quantidade total de amostras por sinal é de 3072 pontos, como mencionado na subseção 3.1.1.

Considerando essas dados, exclusivamente para o algoritmo ALMP funcionar, foi necessário criar quatro vetores para armazenas os sinais durante as iterações, sendo eles o sinal do átomo atual, o do resíduo, o do sinal reconstruído, e o sinal para a entrada da FFT que é uma copia temporária do resíduo. Esses quatro vetores são do tipo *float* (de 32 *bits* ou 4 *bytes*) cada

um de 3072 posições, isso é, cada vetor ocupa exatamente 12 kB (3072 amostras * 4 bytes) na memória RAM, totalizando 48 kB apenas para armazenar esses sinais.

Adicionado a isso, na subseção 3.2.1 foi citado que para habilitar o multi-modo triplo do DMA, foi necessário ativar os três ADCs do microcontrolador, recebendo simultaneamente as conversões dos três ADCs. Assim, foi criado um *buffer* do tipo *uint16_t* (de 16 *bits* ou 2 *bytes*) com 9216 posições (3072 amostras * 3 ADCs). Resultando em um buffer de tamanho 18 kB (9216 amostras * 2 *bytes*) para guardar os valores das conversões dos três ADCs.

Para realizar a norma infinito do sinal da FFT e extrair o átomo de maior amplitude, com a sua frequência e fase, foi utilizado outros dois vetores, um para armazenar a parte real da FFT e outro para a parte imaginaria. Ambos os vetores são do tipo *float* com 1024 posições, resultando em 4 kB (1024 posições * 4 bytes) cada, assim, um total de 8 kB.

Para as funções da biblioteca `levmar` funcionarem é necessário alocar um *buffer* de tipo *float* como espaço de memória para realizar as operações de mínimos quadrados não lineares. Considerando uma equação de otimização com três parâmetros (A, ω, θ) e 3072 amostras, o espaço de memória requerido pela biblioteca foi 21525 posições (4 * amostras + 4 * parâmetros + amostras * parâmetros + parâmetros * parâmetros), resultando em um *buffer* de 84,08 kB (21525 posições * 4 *bytes*).

Para o RTOS foi preciso alocar um espaço de memória dinâmica para as operações do kernel, tarefas, objetos e serviços do sistema. No FreeRTOS esse espaço de memória é chamado de `ucheap` e para este projeto foi definido um tamanho de 32 kB, suficiente para o funcionamento de todo protótipo.

Já a biblioteca `LVGL` possui três grandes *buffers* internos, o primeiro chamado `work_mem_int` é alocado dinamicamente na memória, e é usado para armazenar temporariamente dados de execução da biblioteca, seu tamanho é determinado pela resolução do *display* e da quantidade de objetos usados. Os outros dois *buffers* são os `buf1_1.2` e `buf1_2.1`, sendo destinados ao desenho e renderização das imagens e *widgets* na interface gráfica. O tamanho resultante do `work_mem_int` foi de 36 kB, e dos *buffers* `buf1_1.2` e `buf1_2.1` 30 kB cada, totalizando 96 kB para a utilização dessa biblioteca.

Calculando o tamanho das variáveis mencionada, foi necessário cerca de 286,08 kB de memória RAM, porém foram utilizadas outras de menor tamanho, totalizando 295,94 kB de memória RAM usada. Dos 320 kB de memória disponível no microcontrolador, foram utilizados 295,94 kB, cerca de 92,33 % do total. Apesar de o protótipo conseguir estimar componentes harmônicas e inter-harmônicas e apresentar o resultado das estimações em um display, é possível concluir que os requisitos de memória estão no limite, por pouco impossibilitando a realização desse projeto com os recursos utilizados ou requirindo uma otimização mais profunda de código.

5 CONCLUSÃO

Este trabalho teve o objetivo de desenvolver um protótipo de medidor de componentes harmônicas e inter-harmônicas utilizando o algoritmo *Anti-Leakage Matching Pursuit* (ALMP), em que o *hardware* base para o desenvolvimento desse projeto foi o *kit* de desenvolvimento STM32F746G-DISCO.

Para implementar o algoritmo ALMP no *kit* de desenvolvimento foi necessário implementar o algoritmo de Levenberg-Marquardt (LM) para resolver problemas de mínimos quadrados não lineares. O algoritmo LM foi instalado no microcontrolador do *kit* de desenvolvimento por meio da biblioteca `levmar`, buscando estimar os parâmetros otimizados das ondas presentes em sinais elétricos, porém exigiu alta carga computacional do microcontrolador e espaço de memória RAM.

Buscando duas entradas de sinais paralelas para o ALMP, foi utilizado o multi-modo do periférico DMA do microcontrolador, para receber as conversões dos ADCs. Porém foi necessário ativar três ADCs do microcontrolador, em vez de dois, pois o *kit* de desenvolvimento disponibilizava acesso apenas aos pinos de entrada dos ADC1 e ADC3, sendo necessário ativar o multi-modo triplo do DMA, mesmo sem acesso ao ADC2.

No *kit* foi instalado o sistema operacional `FreeRTOS`, buscando realizar as atividades do protótipo em tempo real. Nele foi utilizado duas tarefas uma para aquisição de sinais e estimação das componentes de frequência e outra para exposição e atualização dos resultados em uma interface gráfica, sendo necessário quatro semáforos para sincronizar as tarefas e uma fila para indicar a primeira tarefa que os sinais terminaram de ser adquiridos.

Também foi instalada a biblioteca `LVGL` para apresentação dos resultados de medição das componentes de frequência e seus parâmetros de interesse no *display* do *kit* de desenvolvimento. Foram desenvolvidas duas guias para a apresentação dos sinais, uma para exibir cada átomo e seus parâmetros numericamente, e outra guia para mostrar a forma de onda do sinal no tempo.

Para avaliar o funcionamento do algoritmo ALMP foram recriados sinais de fontes de distorções harmônicas e inter-harmônicas no *software* Ultrawave, exportadas para o gerador de funções e ligadas aos pinos dos ADCs do microcontrolador. O protótipo conseguiu estimar corretamente os parâmetros de amplitude e frequência dos átomos desses sinais, porém o parâmetro de fase acabou sendo incorreto para qualquer átomo, uma vez que a aquisição dos sinais não iniciam quando o sinal cruza a origem ou o 0 pu, o que resulta em uma fase flutuante.

Apesar de conseguir fazer as estimações dos átomos de sinais, em alguns casos, componentes de frequência extras são encontradas, causadas por *offset* de tensão e ou ruído dos equipamentos utilizados, e em outros sinais, alguns átomos acabam não sendo encontrados devido sua baixa amplitude, sendo considerados ruído pelo algoritmo ALMP.

Foi verificado que as funções da biblioteca *levmar* exigem muito processamento, podendo levar até de 230 ms para estimar um único átomo com a função `slevmar_der()`, indi-

cando que o conjunto usado com o microcontrolador STM32F746NGH6 e a biblioteca `levmar` não esta de acordo com a norma IEC 61000-4-7 (IEC, 2006). Adicionado a isso, para todos componentes do protótipo funcionar corretamente foi utilizado 92,33 % da memória RAM disponível, mostrando que os recursos utilizados estão no limite dos requisitos de memória para este projeto.

Para projetos futuros, podem ser utilizados outros *kits* de desenvolvimento ou microcontroladores com maiores frequência de *clock* e memória RAM. Utilizar, otimizar ou desenvolver bibliotecas para resolução de mínimos quadrados não lineares próprias para microcontroladores, de preferência para os de arquitetura ARM. Desenvolver uma placa de instrumentação buscando fazer a aquisição de sinais de tensão e corrente, com sensores, amplificadores de instrumentação e filtros passa-baixa para melhorar o sinal medido, eliminando *offset*, o ruido e as componentes de alta frequência, possivelmente evitando que átomos inexistentes sejam estimados, e encontrar formas de melhorar o critério de parada do algoritmo ALMP evitando que componentes de frequência reais de baixa amplitude ainda sejam consideradas na estimação.

REFERÊNCIAS

AMAZON WEB SERVICES, INC. **FreeRTOS Kernel Quick Start Guide**: Website. 2023. Disponível em: https://www.freertos.org/FreeRTOS-quick-start-guide.html. Acesso em: 30 de maio de 2023.

ARRILLAGA, J.; WATSON, N. R. **Power system harmonics**. [*S.l.*]: John Wiley & Sons, 2004.

BALTAZAR, A. C. d. S. **Qualidade da energia no contexto da reestruturação do setor elétrico brasileiro**. 2007. 26-33 p. Tese (Doutorado) — Universidade de São Paulo, 2007.

BARROS, A. M.; GUARNERI, G. A.; LAZZARETTI, A. E. A fast gridless algorithm for harmonics and interharmonics estimation. **Electric Power Systems Research**, Elsevier, v. 196, p. 107227, 2021.

BOLLEN, M. H.; GU, I. Y. **Signal processing of power quality disturbances**. [*S.l.*]: John Wiley & Sons, 2006.

BRASIL, M. d. M. e. E. Plano nacional de energia 2050. **Plano Nacional de Energia 2050 - PNE 2050**, p. 37–209, MME/EPE, 2020.

CHEN, L. *et al.* Method based on sparse signal decomposition for harmonic and inter-harmonic analysis of power system. **Journal of Electrical Engineering and Technology**, The Korean Institute of Electrical Engineers, v. 12, n. 2, p. 559–568, 2017.

DENARDIN, G. W.; BARRIQUELLO, C. H. **Sistemas operacionais de tempo real e sua aplicaçao em sistemas embarcados**. [*S.l.*]: Editora Blucher, 2019.

HANZELKA, Z.; BIEŃ, A. Power quality application guide–section 3.1. 1: Harmonics and interharmonics. **Leonardo Power Quality Initiative (Org. & Ed.). AGH University of Science and Technology**, 2004.

HUA, Y.; SARKAR, T. K. Matrix pencil method and its performance. *In*: IEEE COMPUTER SOCIETY. **ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing**. [*S.l.*], 1988. p. 2476–2477.

IEC. **IEC 61000-4-7 - electromagnetic compatibility (EMC) part 4–7: Testing and measurement techniques general guide on harmonics and interharmonics measurements and instrumentation, for power supply systems and equipment connected thereto**. [*S.l.*], 2006.

LOURAKIS, M. I. *et al.* A brief description of the levenberg-marquardt algorithm implemented by levmar. **Foundation of Research and Technology**, v. 4, n. 1, p. 1–6, 2005.

LOURAKIS, M. I. A. **levmar : Levenberg-Marquardt nonlinear least squares algorithms in C/C++**: Technical overview. 2011. Ver. 2.6. Disponível em: http://users.ics.forth.gr/~lourakis/levmar/. Acesso em: 30 de maio de 2023.

LVGL KFT. **Documentation**: Webpage. 2023. Ver 8.3. Disponível em: https://docs.lvgl.io/master/index.html. Acesso em: 30 de maio de 2023.

LVGL KFT. **Introduction**: Webpage. 2023. Ver 8.3. Disponível em: https://docs.lvgl.io/master/intro/index.html. Acesso em: 30 de maio de 2023.

MALLAT, S.; ZHANG, Z. Matching pursuits with time-frequency dictionaries. **IEEE Transactions on Signal Processing**, v. 41, n. 12, p. 3397–3415, 1993.

NOCEDAL, J.; WRIGHT, S. **Numerical optimization**. [*S.l.*]: Springer Science & Business Media, 2006.

OLIVEIRA, W. R. d. Uma contribuição para a medição de distorções harmônicas e inter-harmônicas em instalações de geração fotovoltaica. 2015.

PENSE. **Eu Prometo Que Vou Tentar**. 2011. Disponível em: https://on.soundcloud.com/xxQbS. Acesso em: 28 de maio de 2023.

PRADO, T. D. A.; BARROS, A. M.; GUARNERI, G. A. An algorithm based on sparse decomposition for estimating harmonic and interharmonic components of stationary signals in power systems. **IEEE Access**, IEEE, v. 7, p. 163958–163968, 2019.

RICE, D. E. A detailed analysis of six-pulse converter harmonic currents. **IEEE Transactions on Industry Applications**, IEEE, v. 30, n. 2, p. 294–304, 1994.

RIGOL TECHNOLOGIES, INC. **DG1000 Series Dual-Channel Function/Arbitrary Waveform Generator**: Data sheet. [*S.l.*], 2015. Disponível em: https://beyondmeasure.rigoltech.com/acton/attachment/1579/f-001b/0/-/-/-/-/file.pdf. Acesso em: 30 de maio de 2023.

ROCHA, T. C. d. **Estudo e aplicação da transformada de Fourier na regularização de dados sísmicos na exploração de petróleo**. 2016. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2016.

SANTOSO, S. *et al.* **Electrical power systems quality**. [*S.l.*]: McGraw-Hill Education, 2004.

SINGER, S.; NELDER, J. Nelder-mead algorithm. **Scholarpedia**, v. 4, n. 7, p. 2928, 2009.

SPYROPOULOS, D. V.; MITRONIKAS, E. D. A review on the faults of electric machines used in electric ships. **Advances in Power Electronics**, 2013.

STMICROELECTRONICS. **ARM®-based Cortex®-M7 32b MCU+FPU**: Datasheet - production data. [*S.l.*], 2016. DocID027590 Rev 4. Disponível em: https://www.st.com/resource/en/datasheet/stm32f746ng.pdf. Acesso em: 30 de maio de 2023.

STMICROELECTRONICS. **Discovery kit with STM32F746NG MCU**: Data brief. [*S.l.*], 2019. DB2582 - Rev 3. Disponível em: https://www.st.com/resource/en/data_brief/32f746gdiscovery.pdf. Acesso em: 30 de maio de 2023.

STMICROELECTRONICS. **Integrated development environment for STM32 products**: Data brief. [*S.l.*], 2021. DB3871 - Rev 6. Disponível em: https://www.st.com/en/development-tools/stm32cubeide.html. Acesso em: 30 de maio de 2023.

STMICROELECTRONICS. **STM32 configuration and initialization C code generation**: Data brief. [*S.l.*], 2023. DB2163 - Rev 19. Disponível em: https://www.st.com/en/development-tools/stm32cubemx.html. Acesso em: 30 de maio de 2023.

SYAM, P. *et al.* Simulation and experimental study of interharmonic performance of a cycloconverter-fedsynchronous motor drive. **IEEE Transactions on Energy Conversion**, IEEE, v. 19, n. 2, p. 325–332, 2004.

TEIXEIRA, D. Â. Análise das distorções harmônicas: estudo de caso de um sistema industrial. Universidade Federal de Minas Gerais, p. 12, 2009.

TEKTRONIX, INC. **Digital Storage Oscilloscopes**: Tds2000c series datasheet. [*S.l.*], 2019. Disponível em: https://download.tek.com/datasheet/TDS2000C-Digital-Storage-Oscilloscope-Datasheet-3GW256458.pdf. Acesso em: 30 de maio de 2023.

TESTA, A. *et al.* Interharmonics: Theory and modeling. **IEEE Transactions on Power Delivery**, IEEE, v. 22, n. 4, p. 2335–2348, 2007.

WANG, J.; YANG, F.; DU, X. Microgrid harmonic and interharmonic analysis algorithm based on cubic spline interpolation signal reconstruction. *In*: IEEE. **IEEE PES Innovative Smart Grid Technologies**. [*S.l.*], 2012. p. 1–5.