DOI: 10.48011/sbse.v1i1.2213

Ferramenta de desenvolvimento para estudos em redes de distribuição ativas

Guilherme Sapede Scofano*, Bruno Amaral*, Marília Novais*, Bruno Wanderley França*, Guilherme Gonçalves Sotelo*

**Universidade Federal Fluminense, Niterói, RJ, 24210-240 Brasil (e-mail: guilhermescofano@id.uff.br, amaral_bruno@id.uff.br, mariliamello@id.uff.br, bwfranca@id.uff.br, gsotelo@id.uff.br)*

Abstract: Differences between the computational code designed to control simulated electronic equipment in Active Distribution Networks and the embedded code to control such equipment in real life scenarios may lead to divergences between results obtained in these steps. This article reports the development of a C programming language set of libraries with portability among different platforms to control three phase inverters, avoiding code refactoring, reducing software development times and mitigating differences between simulation code and micro-controller code. The libraries were called by an application running in a Texas Instruments TMS320F28379D micro-controller and the simulation software PSIM to demonstrate the tool effectiveness and compare results.

Resumo: Diferenças entre o código computacional utilizado para o controle de equipamentos eletrônicos de Redes de Distribuição Ativas (RDA) simulados e o código embarcado para o controle dos equipamentos reais podem criar discrepâncias entre os resultados obtidos nas duas etapas. Foi desenvolvido um conjunto de bibliotecas na linguagem C de programação para o controle de inversores trifásicos com portabilidade entre diferentes plataformas de processamento que evita a refatoração de código, reduzindo o tempo de desenvolvimento de software e as diferenças entre código de simulação e código embarcado. As bibliotecas foram empregadas em um microcontrolador da Texas TMS320F28379D e no software de simulação PSIM para demonstrar a eficácia da ferramenta e comparar de resultados.

Keywords: Inverter; power eletronics; Active Distribution Networks; TMS320F28379D; PSIM.

Palavras-chaves: Inversor; eletrônica de potência; Rede de distribuição ativa; TMSF28379D; PSIM

1. INTRODUÇÃO

De acordo com (Hua et al., 2015) e (Karthikeyan et al., 2017), Redes de Distribuição Ativas (RDA) são redes bidirecionais fornecedoras de energia que incluem geração distribuída empregada em larga escala e cargas flexíveis. Esse tipo de rede, entretanto, traz cada vez mais desafios à operação e ao gerenciamento dos sistemas como a deterioração da qualidade de energia, a baixa capacidade de restabelecimento do sistema após defeitos, os custos econômicos elevados, a manutenção do balanço de potência entre geração e demanda, dentre outros.

Dentre os equipamentos tipicamente encontrados em RDAs, destacam-se painéis fotovoltaicos, veículos elétricos, bancos de baterias e outros acumuladores de energia (Xiaorui et al., 2018). Tais equipamentos são caracterizados por terem componentes que trabalham em regime de Corrente Contínua (CC) e, portanto, necessitam de um dispositivo que converta o sinal para regime de Corrente Alternada (CA) de forma que a conexão com a rede elétrica ou um motor CA possa ser feita em segurança. Os inversores de frequência são empregados para satisfazer essa necessidade, ganhando seu destaque nas RDAs.

Parte fundamental do desenvolvimento de conversores de potência é realizar simulações em tempo real para validar tanto o projeto de hardware quanto o de software (Herrera et al., 2015) e reduzir tempo e custos de desenvolvimento (Sang-kyu Kwak et al., 2017, p.). Entretanto, a programação utilizada para implementar o controle desses conversores em simuladores como o PSCAD/EMTDC e o PSIM, da PowerSim, pode diferir da programação desenvolvida para os microcontroladores que governam esse tipo de equipamento. Este trabalho propõe uma forma de reduzir, tanto quanto possível, as diferenças decorrentes da transcrição de código através da utilização de um padrão de desenvolvimento com alto grau de portabilidade entre diferentes plataformas – sejam elas simuladores ou microcontroladores – permitindo, portanto, que as mesmas sub-rotinas validadas em simulação sejam utilizadas no próprio equipamento com uma quantidade mínima de modificações. Uma das motivações para o desenvolvimento da ferramenta proposta é que ela possa ser utilizada nas atividades do Núcleo de Inovação Tecnológica em Engenharia Elétrica (Nitee), laboratório do Departamento de Engenharia Elétrica da Universidade Federal Fluminense, auxiliando no processo de desenvolvimento e teste tanto de software para inversores quanto novas estratégias de controle.

Outras formas de se validar controladores são encontradas na literatura. (Peric et al., 2019) faz o comparativo entre o desenvolvimento de um conversor de potência utilizando a

técnica *Hardware in the Loop* (HiL) e a técnica *Software in the Loop (SiL)*. (Teixeira et al., 2017) reporta a implementação da portabilidade de código entre PSIM e microcontrolador através de diretivas de pré-compilação da linguagem C de programação apenas em determinados trechos do programa, em que a divergência entre as plataformas eram incontornáveis.

2. METODOLOGIA

Foi criada uma divisão funcional da aplicação, tornando possível implementar os conceitos de modularização e encapsulamento. (White, 2012) afirma que esses dois conceitos permitem que diferentes partes do código sejam desenvolvidas simultaneamente sem que haja conflito. Cada módulo é implementado através de uma biblioteca da linguagem C de programação que provê um tipo de dados estruturados – conhecido como "struct", na linguagem – e um conjunto de funções para interagir com esse tipo de dados. Cada tipo de dados criado modela um componente do inversor de frequência e as funções os inicializam, os removem da memória (evitando *memory leaks*) e os utilizam para realizar algoritmos de controle.

2.1 Modularização

Os módulos desenvolvidos interagem entre si e com o hardware do equipamento de acordo com a topologia apresentada na Fig. 1:

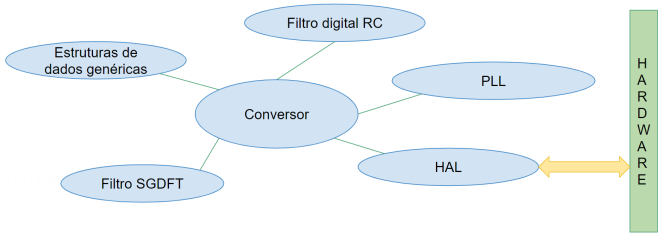

Fig. 1: Topologia da aplicação

O módulo principal é implementado pela biblioteca Conversor.h, que declara o tipo de dados "Conversor" para modelar o próprio conversor de potência. Esse tipo de dados armazena características inerentes ao equipamento, tais como as leituras de tensão e corrente mais recentes de cada fase, a frequência nominal para qual o conversor está configurado, a referência angular de cada fase, se o conversor está habilitado pelo usuário ou não, dentre outras informações. O tipo "Conversor", por sua vez, utiliza variáveis de outros tipos de dados definidos nas demais bibliotecas, podendo acessá-los e trocar dados por meio de ponteiros e funções.

Os recursos fornecidos pelas outras bibliotecas são:

• PLL: implementa uma Malha de Captura de Fase (do inglês *Phase Locked Loop* – PLL) baseada em uma das topologias apresentadas por (Karimi-Ghartemani, 2014).

• Filtro SGDFT: fornece um filtro digital baseado no algoritmo *Sliding Goertzel Discrete Fourier Transform* (SGDFT) (Lyons, 2003).

• Filtro digital RC: fornece um filtro digital de primeira ordem, passa-baixa, de resposta infinita ao impulso cuja parametrização se baseia nas equações de um filtro com resistor e capacitor.

• Estruturas de dados genéricas: fornece componentes genéricos de software. A versão atual disponibiliza apenas buffers circulares, mas em versões posteriores os desenvolvedores podem agregar novas estruturas que venham a ser necessárias.

• HAL: realiza a interface da aplicação com o ambiente no qual ela é executada, seja o hardware de um microcontrolador ou uma simulação.

2.2 Encapsulamento

A linguagem C de programação permite separar as declarações das funções das suas definições (Griffiths and Griffiths, 2012). As declarações se dão por meio dos protótipos, geralmente escritos em arquivos de cabeçalho (terminados na extensão ".h"). Os protótipos contêm apenas os nomes das funções, seus argumentos e seu tipo de retorno. Os tipos de dados estruturados, da mesma forma, podem ter seus nomes definidos nesses arquivos de cabeçalho. Já o conteúdo tanto das funções quanto dos tipos de dados – ou seja, seus detalhes de implementação – são definidos, em geral, nos arquivos-fonte (aqueles terminados na extensão ".c"). Sendo assim, cada biblioteca é composta por dois arquivos: um de cabeçalho e outro fonte.

Os demais arquivos da aplicação conseguem acessar as funções de uma determinada biblioteca incluindo seu arquivo de cabeçalho através da diretiva de pré-compilação #include. Da mesma forma, é possível criar ponteiros para variáveis de tipos definidos nessa biblioteca. Mesmo que os detalhes de implementação fiquem no arquivo-fonte correspondente, são inacessíveis do ponto de vista do arquivo que incluiu o arquivo de cabeçalho. Assim sendo, a modificação de uma variável cujo tipo foi definido no arquivo-fonte de uma biblioteca só é possível por meio das funções que ela disponibiliza.

2.3 Portabilidade entre plataformas

É possível explorar os conceitos de encapsulamento e modularização para prover portabilidade entre diferentes plataformas à aplicação. (Huang and Wu, 2018) mostra um artifício conhecido como Camada de Abstração de Hardware (do inglês *Hardware Abstraction Layer* – HAL). Esse artifício consiste em criar uma camada de abstração que transforma instruções particulares de cada plataforma em funções padronizadas e conhecidas pelo resto da aplicação.

Na aplicação desenvolvida, a HAL foi implementada através da biblioteca “hal.h”. O módulo HAL traz funções para realizar leitura e escrita de dados que representam sinais, mas também seria possível incluir funções para gerenciamento de *threads*, comunicação entre processos, utilização de periféricos e demais interações com hardware. Os arquivos de cabeçalho permanecem constantes quando ocorre uma mudança de plataforma, declarando sempre os mesmos tipos de dados e as mesmas funções, com mesmos argumentos e tipos de retorno. Os arquivo-fonte, entretanto, que definem a implementação de cada função, são modificados quando ocorre a portabilidade, adequando os códigos de entrada e saída de dados aos padrões da nova plataforma. A Fig. 2 ilustra esse processo.

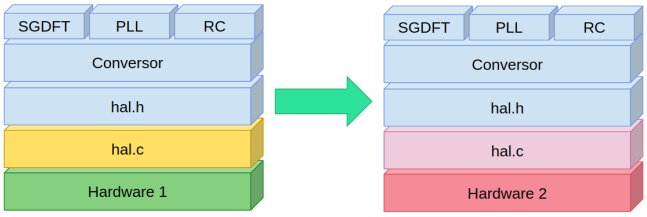

Fig. 2: Migração da aplicação para uma nova plataforma

Na Fig. 2, os módulos SGDFT, PLL e RC implementam funções com maior grau de abstração para serem utilizadas pelo módulo Conversor. O módulo Conversor, por sua vez, utiliza sempre as mesmas funções declaradas pelo arquivo de cabeçalho hal.h. O arquivo-fonte, entretanto, muda a forma como essas funções implementam a interação com hardware conforme ocorre uma migração para outra plataforma, apesar de seus nomes, argumentos e tipos de retorno permanecerem os mesmos.

2.4 Utilização das bibliotecas

Foi proposto um circuito trifásico de inversor com filtro LC alimentando uma carga puramente resistiva, conforme mostra a representação monofásica na Fig. 3.

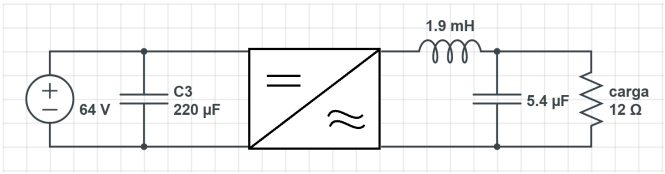

Fig. 3: Circuito de teste proposto

A frequência fundamental do circuito foi definida em 60 Hz. O indutor, o capacitor e o resistor foram escolhidos de acordo com componentes reais disponíveis no laboratório. Esses componentes tinham erros de até 2,1% em relação aos dados de placa. Foi estipulado que a frequência de comutação do inversor deveria ser um harmônico ímpar e múltiplo de 3 da frequência fundamental para que as componentes harmônicas da tensão se concentrassem em nuvens ao redor de múltiplos da frequência de comutação conforme mostrado em (Mohan et al., 2003). O múltiplo 171 foi empiricamente escolhido, resultando em uma frequência de 10.260 Hz. Dado o caráter empírico dessas decisões, é necessário avaliar se o circuito atende critérios de desempenho. Os critérios escolhidos foram que nenhuma componente harmônica de tensão deveria passar de 5% do valor da fundamental e que o indutor tivesse uma queda de tensão menor ou igual a 5% da tensão no nó de comutação.

O circuito foi montado em bancada com o inversor 3PhGanInv da Texas Instruments acoplado no Launchpad F28379D, do mesmo fabricante, que atuou como controlador e recebeu a aplicação embarcada. O launchpad é uma plataforma de avaliação do microcontrolador TMS320F28379D. O ensaio está mostrado na Fig. 4:

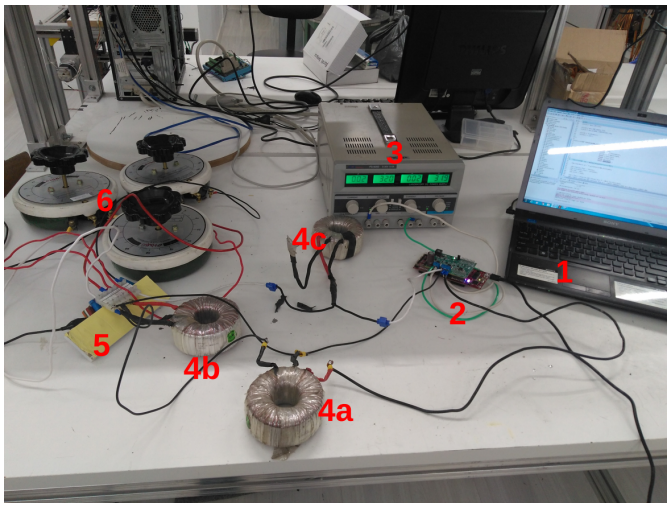

Fig. 4: Ensaio para validar a aplicação desenvolvida

A Fig. 4 mostra: computador para debug do código no controlador (1), inversor acoplado à unidade de controle (2), fonte de tensão CC (3), os indutores do filtro LC (4A, B e C), o banco de capacitores do filtro LC (5) e a carga (6).

Foi realizada uma simulação no PSIM considerando os dados de placa dos componentes conforme mostra a Fig. 5. Estão representados: a fonte de tensão CC (1); um par de chaves do inversor (2); o condicionamento de corrente (3); o filtro LC, a carga e a medição de tensão na carga (4); o ponto neutro do elo CA do inversor (5); um comparador utilizado para modulação em largura de pulso que possui, como entrada, uma onda triangular na frequência de comutação e um sinal de referência proveniente do controlador (6); o bloco de programação em C onde se carrega a aplicação desenvolvida (7); a saídas do bloco de programação em C que são, respectivamente, 3 sinais de referência e um sinal para habilitar a comutação das chaves (8); as entradas do bloco de programação em C que são, respectivamente, as tensões medidas no nó de comutação com a mesma referência do elo CC, as medições de corrente provenientes do condicionamento visto no número 3 e um gerador de sinal utilizado para que os algoritmos do bloco de programação em C sejam executados

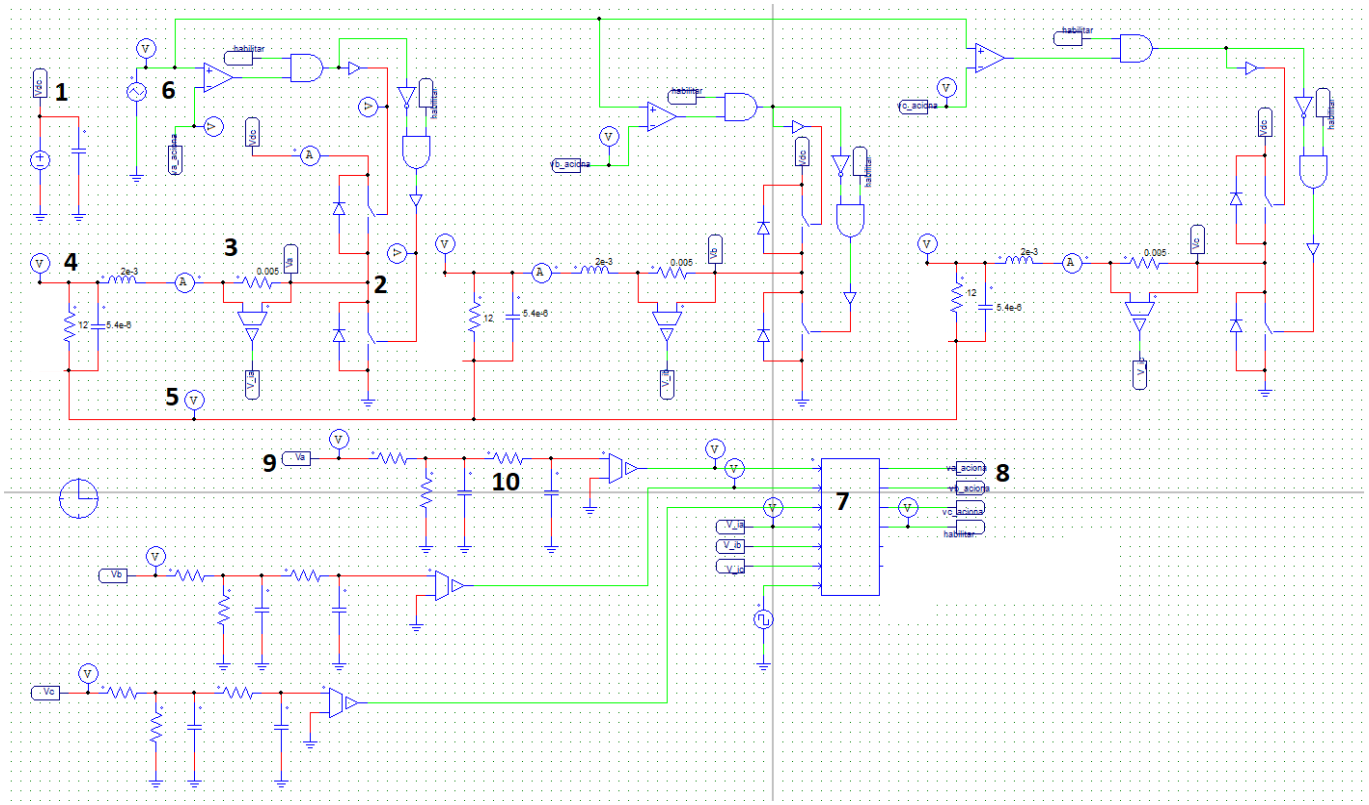

Fig. 5: Simulação no PSIM

com a mesma frequência das interrupções do microcontrolador (9); e os filtros do condicionamento das leituras de tensão (10). As portas lógicas E no circuito permitem que um pulso de habilitação proveniente do bloco de programação em C desative a comutação nas chaves, tal como ocorre no inversor físico.

O bloco de programação em C do PSIM possui uma seção a ser executada somente uma vez e uma seção a ser executada repetidamente ao longo a simulação. Da mesma forma, o microcontrolador executa uma vez o conteúdo da função "main()" e nela é possível configurar interrupções temporizadas para serem executadas sucessivamente ao longo do ensaio. Para utilizar a ferramenta, foi declarada uma variável global do tipo Conversor para armazenar as informações do equipamento e utilizada a função Conversor_criar para inicializá-la. Para executar os algoritmos de controle durante a operação do inversor, foi chamada a sub-rotina Conversor_computa a cada interrupção do microcontrolador e na seção a ser executada repetidamente no bloco destinado à programação em C do PSIM, utilizando a variável do tipo Conversor criada como argumento. Essa sub-rotina executa a estratégia de controle programada para o conversor.

A sub-rotina Conversor_computa é responsável por ditar o sinal de referência de cada fase para ser modulado em largura de pulso. Como a carga não foi conectada a nenhuma outra fonte de tensão, o inversor pode ditar a amplitude e a referência angular da tensão CA. A amplitude da tensão de cada fase é igual a uma constante m_a, definida pelo usuário, multiplicada pela tensão no elo CC do conversor. A constante m_a foi definida em 0,8 para que não houvesse sobremodulação do sinal.

3. Resultados

Foram medidas a tensão na carga e a corrente na saída do inversor (antes do filtro LC). No ensaio, foi utilizado o osciloscópio Yokogawa 8152, cujos resultados foram exportados para o Octave 5.1. No PSIM, o próprio software já é capaz de plotar os resultados das medições.

3.1 Resultados da simulação

A simulação mostrou que os sinais trifásicos de tensão e corrente ficaram em 60 Hz e equilibrados em amplitude e fase. O PSIM indicou um valor de tensão de pico igual a 25,58 V e um valor de corrente de pico igual a 2.1321 A. A Tabela 1 apresenta a amplitude das componentes harmônicas dos sinais de tensão e corrente em percentuais das amplitudes das suas respectivas fundamentais. Está representado apenas o valor máximo para cada nuvem harmônica no entorno dos múltiplos da frequência de comutação.

Tabela 1. Amplitude percentual das componentes harmônicas dos sinais de tensão e corrente em relação à fundamental

Frequência (Hz)	10.260	20.520	30.780	41.040
Tensão (% da fundamental)	0,61	0,22	0,05	0,01
Corrente (% da fundamental)	2,6	1,9	0,7	0,3

A maior amplitude de componente harmônica de tensão ficou a menos de 1% da amplitude da fundamental. A distorção harmônica total (do inglês Total Harmonic Distortion – THD) de tensão encontrada foi 9,231 x 10^{-3}.

3.2 Resultados experimentais

A plotagem da tensão na carga pode ser vista na Fig. 6:

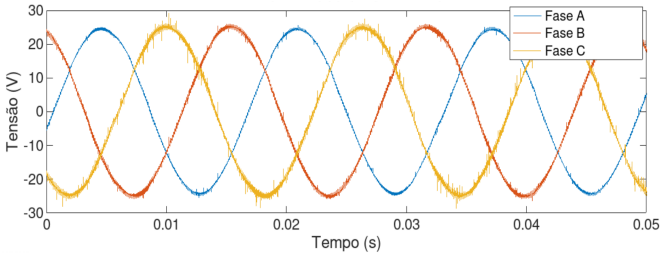

Fig. 6: Tensão de fase na carga durante ensaio

A tensão de fase na carga está em 60 Hz, mas nota-se um desequilíbrio de amplitude. O valor de pico registrado da componente fundamental foi de 24,119 V. A componente fundamental da tensão no nó de comutação foi 24,740 V, implicando que a queda de tensão no indutor foi de 621 mV. Essa queda de tensão é de aproximadamente 2,5% do valor da tensão no nó de comutação. Percebe-se, também, de forma qualitativa, que a tensão da fase C contém uma distorção harmônica superior às demais. Os desvios dos valores de resistência, capacitância e indutância dos componentes utilizados em relação aos seus dados de placa podem ter provocado a distorção harmônica e o desequilíbrio constatados.

Foi avaliado o espectro de frequências da tensão na fase C, considerado o caso de maior distorção, na Fig. 7. A componente fundamental foi omitida para facilitar a visualização das demais. As componentes harmônicas restringem-se a nuvens no entorno dos múltiplos da frequência de comutação. A nuvem perceptível de maior frequência ficou em torno de 41.040 Hz. A maior amplitude harmônica encontrada teve seu valor de pico próximo de 500 mV, que seria aproximadamente 2% da amplitude da componente fundamental.

Já a corrente de saída do inversor está plotada na Fig. 8. Nota-se que as correntes estão em 60 Hz mas, tal como ocorreu nas formas de onda de tensão, percebe-se um desequilíbrio de amplitude decorrente atribuido à divergência entre os valores reais dos componentes utilizados na prática e seus dados de placa. A amplitude da componente fundamental foi de 2,489 A. O espectro harmônico pode ser visto na Fig. 9, onde foi omitida a componente fundamental para facilitar a visualização.

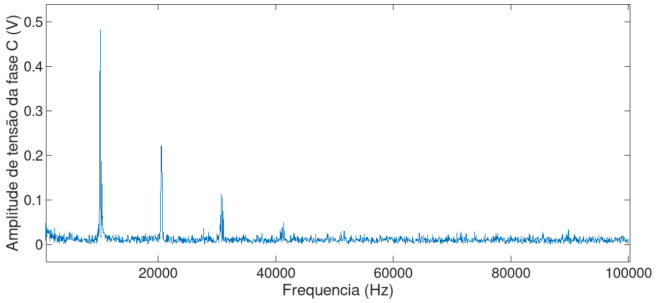

Fig. 7: Espectro de frequências da tensão na fase C da carga durante ensaio

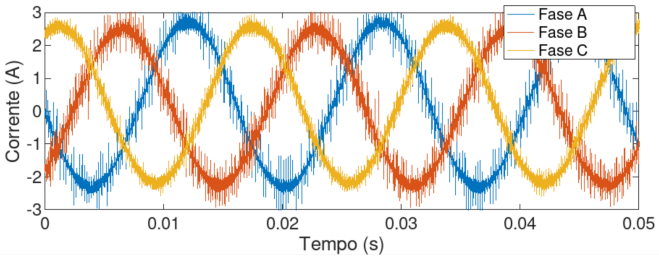

Fig. 8: Corrente de fase na saída do inversor durante o ensaio

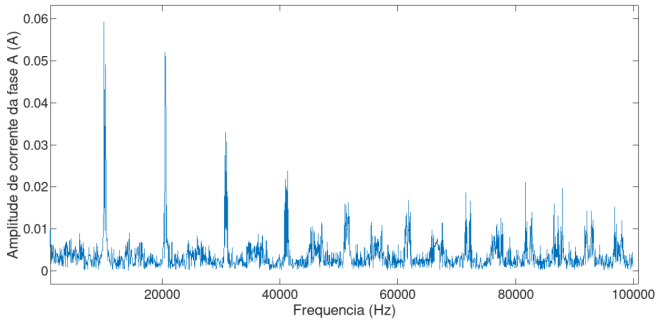

Fig. 9: Espectro de frequências da corrente de saída do inversor durante ensaio

As componentes harmônicas vistas na Fig. 9 estão dispostas no entorno dos múltiplos da frequência de comutação do inversor. A maior amplitude harmônica observada foi de 60 mA, menos que 2,5% da amplitude da componente da fundamental.

4. Conclusão

A ferramenta proposta foi capaz de controlar corretamente o inversor. Os sinais de saída foram senoidais e com a frequência e a amplitude desejados. A queda de tensão no indutor foi de 2,5% do valor da fundamental, estando abaixo do limite estipulado em 5%. O critério de que nenhuma componente harmônica deveria ultrapassar 5% do valor de amplitude da fundamental também foi respeitado, uma vez

que a maior amplitude encontrada foi de aproximadamente 2% da fundamental.

A maior proposta da ferramenta foi alcançada: foram necessárias modificações somente no arquivo da camada de abstração de hardware para adequar as entradas e saídas da aplicação ao PSIM e ao microcontrolador. Os demais componentes de software permaneceram inalterados, podendo ser executados independentemente da plataforma na qual estejam inseridos.

Propõe-se que novos algoritmos de controle e aprimoramentos sejam implementados. Uma primeira proposta seria um algoritmo que possibilite o inversor sincronizar com a rede elétrica através do PLL desenvolvido e injetar corrente em um sistema caracterizado como barramento de tensão infinito. Destaca-se, para tal finalidade, o controle por banda de histerese, proporcional-integral ou proporcional-ressonante. Um algoritmo de compensação de componente de sequência negativa seria interessante para corrigir os desbalanços de tensão e corrente encontrados nos resultados experimentais. Temas recorrentes na literatura atual – como, por exemplo, filtros ativos (Babu et al., 2016) e máquinas síncronas virtuais (Kumar et al., 2018) também podem ser investigados desenvolvendo-se novos módulos dentro da arquitetura disponibilizada pela ferramenta.

5. Referências

Babu, P.N., Kar, B., Halder, B., 2016. Modelling and analysis of a hybrid active power filter for power quality improvement using hysteresis current control technique, em: 2016 7th India International Conference on Power Electronics (IICPE). Apresentado em Patiala, India, páginas 1–6. https://doi.org/10.1109/IICPE.2016.8079381

Griffiths, David, Griffiths, Dawn, 2012. Head first C, 1st ed. ed, Head first series. O'Reilly, Beijing ; Sebastopol, CA.

Herrera, L., Li, C., Yao, X., Wang, J., 2015. FPGA-Based Detailed Real-Time Simulation of Power Converters and Electric Machines for EV HIL Applications. IEEE Trans. Ind. Appl. 51, 1702–1712. https://doi.org/10.1109/TIA.2014.2350074

Hua, X., Yuxi, C., Jian, W., Agelidis, V., 2015. Design of energy dispatch strategy of active distribution network using chance-constrained programming, em: 2015 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC). Apresentado em Brisbane, Australia, páginas 1–5. https://doi.org/10.1109/APPEEC.2015.7380932

Huang, D., Wu, H., 2018. Mobile cloud computing: foundations and service models. Morgan Kaufmann Publishers, an imprint of Elsevier, Cambridge, MA, United States.

Karimi-Ghartemani, M., 2014. Enhanced phase-locked loop structures for power and energy applications. John Wiley & Sons Inc, Hoboken, New Jersey.

Karthikeyan, N., Pokhrel, B.R., Pillai, J.R., Bak-Jensen, B., 2017. Multi-level control framework for enhanced flexibility of active distribution network, em: 2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT). Apresentado em Washington, DC, USA, páginas 1–5. https://doi.org/10.1109/ISGT.2017.8085963

Kumar, T.V., Thomas, V., Kumaravel, S., Ashok, S., 2018. Performance of virtual synchronous machine in autonomous mode of operation, em: 2018 5th International Conference on Renewable Energy: Generation and Applications (ICREGA). Apresentado em Al Ain, páginas 310–314. https://doi.org/10.1109/ICREGA.2018.8337612

Lyons, R., 2003. dsp tips & tricks – the sliding DFT. IEEE Signal Process. Mag. 20, 74–80. https://doi.org/10.1109/MSP.2003.1184347

Mohan, N., Undeland, T.M., Robbins, W.P., 2003. Power electronics: converters, applications, and design, 3rd ed. ed. John Wiley & Sons, Hoboken, NJ.

Peric, A., Paukovic, H., Miletic, M., Sunde, V., 2019. Development of voltage source converter using HiL simulation system, em: 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Apresentado em Opatija, Croatia, páginas 168–173. https://doi.org/10.23919/MIPRO.2019.8757020

Sang-kyu Kwak, Sang-Jung Lee, Jee-Hoon Jung, Hyung-Jun Jeon, 2017. Control Hardware-in-the-Loop Simulation test-bed of Power Management System for ship's power system applications, em: 2017 IEEE 3rd International Future Energy Electronics Conference and ECCE Asia (IFEEC 2017 - ECCE Asia). Apresentado em Kaohsiung, Taiwan, páginas 1241–1245. https://doi.org/10.1109/IFEEC.2017.7992220

Teixeira, C.A., Holmes, D.G., McGrath, B.P., Wilkinson, R.H., McGoldrick, P., McIver, A., 2017. A hardware/software co-simulation approach for power converter firmware design and debugging, em: 2017 Australasian Universities Power Engineering Conference (AUPEC). Apresentado em Melbourne, VIC, páginas 1–6. https://doi.org/10.1109/AUPEC.2017.8282430

White, E., 2012. Making embedded systems: design patterns for great software, 1. ed. ed. O'Reilly, Beijing.

Xiaorui, G., Xiang, M., Tiantian, Q., Wenbo, M., 2018. Optimization Allocation Method for Flexible Load as Peaking Resource, em: 2018 China International Conference on Electricity Distribution (CICED). Apresentado em Tianjin, páginas 2800–2804. https://doi.org/10.1109/CICED.2018.8592201