Projeto Baseado em Modelo no Desenvolvimento de um Controlador de Velocidade para Motor BLDC^{*}

Christian Charles Dias^{*} Rafael Bezerra Correia Lima^{**}
Péricles Rezende Barros^{***}

^{*} Laboratório de Instrumentação Eletrônica e Controle, Universidade
Federal de Campina Grande, PB, (e-mail:
christian.dias@ee.ufcg.edu.br).

^{**} Laboratório de Instrumentação Eletrônica e Controle, Universidade
Federal de Campina Grande, PB, (e-mail: rafael.lima@dee.ufcg.edu.br)

^{***} Laboratório de Instrumentação Eletrônica e Controle, Universidade
Federal de Campina Grande, PB, (e-mail: prbarros@dee.ufcg.edu.br)

Abstract: The increase of embedded electronics and the improvements of the functionalities realized by the embedded software make that the adoption of methodologies that support the management and the development of complex systems are necessary. From this perspective, many engineers are utilizing to Model-Based Design (MBD) methodology, especially those who perform complex and critical system development. In this work, the authors present the use of the MBD methodology applied to the development of a three-phase BLDC motor speed controller. The aspects related to the development of models for simulation and code generation are discussed. Comparisons between simulated and experimental results are also presented well with the use of memory resources and runtime algorithms.

Resumo: O crescimento da eletrônica embarcada e o expressivo aumento das funcionalidades realizadas pelos softwares embarcados fazem com que a adoção de metodologias que possibilitem o gerenciamento e o desenvolvimento de sistemas complexos sejam necessárias. Sob essa ótica, muitos engenheiros estão se voltando para a metodologia de projeto baseado em modelo (MBD), principalmente aqueles que realizam o desenvolvimento de sistema complexos e críticos. Dessa forma, neste trabalho é abordado a utilização da metodologia MBD aplicada ao desenvolvimento de um controlador de velocidade para motor BLDC trifásico. São abordados os aspectos relacionados ao desenvolvimento dos modelos para simulação e geração do código. Comparativos entre os resultados simulados e experimentais também são apresentados bem com o uso dos recursos de memória e tempo de execução dos algoritmos.

Keywords: Model Based Design; BLDC motor; Automatic code generation; embedded systems; NUCLEO-F401RE board.

Palavras-chaves: Projeto Baseado em Modelo; motor BLDC; geração automática de código; sistemas embarcados; placa NUCLEO-F401RE.

1. INTRODUÇÃO

O desenvolvimento utilizando a metodologia de projeto baseado em modelo (MBD - *Model-Based Design*) tem sido um tópico de discussão há décadas, mas só nos últimos anos evoluiu para um fluxo de projeto completo, O'Sullivan et al. (2015). Aliado a isso, o crescimento da eletrônica embarcada e o expressivo aumento das funcionalidades realizadas pelos softwares, ou firmware, embarcados fazem com que a adoção de metodologias que possibilitem o gerenciamento e o desenvolvimento de sistemas complexos sejam necessárias. Sob essa ótica, muitos engenheiros estão se voltando para a metodologia MBD, MathWorks (2014), principalmente aqueles que realizam o desenvolvimento de sistema complexos e críticos, como na indústria

automotiva, aeroespacial e naval. Parte desse interesse se deve à possibilidade de modelar a lógica de controle e a planta, permitindo a utilização de um ambiente de simulação para acelerar o desenvolvimento de seu algoritmo, NXP (2017).

Tal qual o MBD, que vem sendo aplicado em diversos setores industriais, os motores elétricos de corrente contínua sem escovas (BLDC) também estão ganhando destaque nas mais diversas áreas, principalmente as que envolvem mobilidade humana utilizando motores elétricos, como bicicletas, *skates*, patinetes, motos e carro. Não se restringido a essas aplicações, eles também são utilizados em sistemas de direção elétrica, limpadores de para-brisa, máquinas de lavar roupas, elevadores e impressoras, por exemplo.

Segundo relatório divulgado em Research (2017), estima-se que o mercado mundial para esse tipo de motor tenha

^{*} Agências de fomento CNPq e CAPES.

faturado cerca de 8,4 bilhões de dólares em 2017, podendo superar o faturamento anual de 12 bilhões de dólares até 2025. Assim, é sinalizada uma tendência de crescimento no número de aplicações utilizando motor BLDC.

Dessa forma, os autores abordam neste trabalho a utilização da metodologia MBD aplicada ao desenvolvimento de um controlador de velocidade para motor BLDC trifásico, embarcado em uma plataforma com microcontrolador ARM Cortex-M4, a placa NUCLEO-F401RE. Aspectos relacionados ao desenvolvimento dos modelos para simulação e geração do código são abordados. Comparativos entre os resultados simulados e experimentais também são apresentados bem como o uso dos recursos de memória e tempo de execução dos algoritmos.

O restante do trabalho está organizado conforme segue: descrição de alguns trabalhos relacionados ao MBD e motor BLDC na seção 2; introdução à metodologia MBD, seção 3; apresentação da plataforma experimental, estratégia de comutação e controle na seção 4; o desenvolvimento dos modelos são apresentados na seção 5; seguindo pelas simulações e experimentos na seção 6; finalizando com a análise dos resultados e conclusão nas seções 7 e 8, respectivamente.

2. TRABALHOS RELACIONADOS

Em Potnuru et al. (2018) foi descrito o desenvolvimento de um sistema em malha fechada para controle da velocidade de um motor BLDC, realizado sobre um hardware RCP (*Rapid Control Prototyping*). A principal vantagem dessa abordagem foi a redução no tempo de desenvolvimento e no teste dos algoritmos. Porém, os RPCs apresentam como desvantagem o fato de serem empregados apenas para validação do algoritmo, não sendo, portanto, utilizados no produto final.

O desenvolvimento de um controlador de velocidade para um motor síncrono de ímãs permanente (PMSM) e as etapas para aplicação do MBD foram abordadas em O'Sullivan et al. (2015), sendo o código da solução gerado automaticamente e embarcado numa plataforma ARM Cortex-M4. Um plataforma Arduino foi utilizada em Hat et al. (2015) para implementar um controlador de velocidade para motor BLDC. Contudo, apenas o controlador PID foi embarcado na lógica executada no firmware, ficando a estratégia de comutação e medição da posição do rotor a cargo de um driver.

Uma comparação de desempenho entre um controlador PI Antiwindup e o controlador PI convencional foi realizada em Tariq et al. (2016). A estratégia de comutação seis passos foi empregada e os testes experimentais foram realizados com algoritmo de controle executado por um DSP TMS320F2808. Contudo, aspectos relacionados ao desenvolvimento do código não fizeram parte do escopo do trabalho.

Dessa forma, os autores deste trabalho visam contribuir por intermédio da exposição de como a metodologia MBD pode ser empregada no desenvolvimento de um controlador de velocidade para motor BLDC; no qual o código da lógica de comutação seis passos, com modulação H_PWM-L_ON, e estratégia de controle PI cascata são geradas automati-

Figura 1. Representação dos testes MIL, PIL e no sistema real, utilizados durante o processo de desenvolvimento com MBD.

camente e embarcado em uma placa microcontrolada de baixo custo, cerca de 13 dólares.

3. PROJETO BASEADO EM MODELO (MBD)

A metodologia MBD é baseada na abordagem matemática e visual para o desenvolvimento de sistemas de controle complexos no qual o modelo é o elemento central do desenvolvimento segundo Aarenstrup (2015). O modelo, que pode consistir em diagramas de blocos, programas textuais e/ou outros elementos gráficos, agrega todas as informações pertinentes ao sistema em especificações executáveis, tais como: modelo dos componentes físicos, propriedade intelectual, lógica de controle, cenários de testes e algoritmos.

3.1 Testes em MBD

Uma das grandes vantagens no MBD é que os testes podem começar ainda na fase de requisitos, possibilitando simular as especificações executáveis dos modelos para verificar se os requisitos são atendidos. Como resultado, os defeitos são localizados e solucionados nas fases iniciais do projeto, reduzindo o custo total do desenvolvimento Lin (2011).

No processo de desenvolvimento com MBD podem ser aplicados os testes em Model-In-the-Loop (MIL), Software-In-the-Loop (SIL), Processor-In-the-Loop (PIL) e Hardware-In-the-Loop MathWorks (2018). Para este trabalho foi utilizada a sequência de testes exibida na Figura 1, na qual foram aplicados os testes em MIL, PIL e com o sistema real, sendo a descrição de cada um deles dada a seguir:

- **MIL** - consiste em realizar simulações dos modelos da planta e do controlador em um mesmo ambiente computacional. Essas simulações são realizadas em linguagem nativa da ferramenta utilizada no desenvolvimento dos modelos, permitindo assim, ter um completo domínio sobre todo o sistema simulado, Figura 1.a;
- **PIL** - o modelo do controlador, convertido em código objeto, passa a ser executado pelo microcontrolador, enquanto a planta continua a ser simulada em linguagem nativa da ferramenta utilizada para o desenvolvimento dos modelos. Uma camada de código é adicionada ao firmware para permitir o envio/recebimento de dados entre a ferramenta de simulação e o microcontrolador, por intermédio da comunicação UART, Figura 1.b;
- **Sistema real** - por fim, os testes são realizados com o sistema embarcado interligado à planta, possibilitando verificar o funcionamento da lógica de controle no microcontrolador em conjunto como as interfaces de entrada/saída, Figura 1.c.

Figura 2. Arquitetura para o controle de velocidade do motor BLDC.

O objetivo com esses testes é realizar a verificação e validação dos requisitos do sistema em diferentes níveis do desenvolvimento, pois à medida que o desenvolvimento da solução avança, o funcionamento e o desempenho do sistema são comparados em cada fase de testes.

3.2 Geração automática do código

A geração automática do código executável, a partir do modelo simulado, é um dos grandes atrativos para a adoção da metodologia MBD no desenvolvimento de sistema complexos. Um dos propósitos é a redução do tempo no desenvolvimento devido à codificação manual bem como os erros provenientes dela.

No desenvolvimento da solução apresentada neste trabalho, a biblioteca de drives fornecida pela empresa STMicroelectronics, fabricante da placa NUCLEO-F401RE, e os códigos desenvolvidos pelos autores, foram integrados a plataforma computacional MathWorks de forma a possibilitar a geração automática do código da solução com a inclusão de códigos legados.

4. ARQUITETURA DA PLATAFORMA, ESTRATÉGIA DE COMUTAÇÃO E CONTROLE

4.1 Plataforma experimental

A arquitetura da plataforma experimental é composta pelos elementos principais exibidos na Figura 2 e descritos a seguir:

- **Sistema embarcado** - a placa NUCLEO-F401RE é responsável por executar a lógica de comutação e a estratégia de controle, além de realizar a leitura dos sinais de corrente e dos sensores de efeito Hall. O microcontrolador foi configurado para funcionar em 84 MHz e utilizar periféricos como conversor A/D, *advanced Timer* e GPIOs digitais;

Tabela 1. Comutação das fases para rotação horária de acordo com a saída dos sensores Hall.

Hall A	Hall B	Hall C	Q1	Q2	Q3	Q4	Q5	Q6
0	1	0	PWM	OFF	OFF	ON	OFF	OFF
0	1	1	PWM	OFF	OFF	OFF	OFF	ON
0	0	1	OFF	OFF	PWM	OFF	OFF	ON
1	0	1	OFF	ON	PWM	OFF	OFF	OFF
1	0	0	OFF	ON	OFF	OFF	PWM	OFF
1	1	0	OFF	OFF	OFF	ON	PWM	OFF

Figura 3. Diagrama em blocos da estratégia de controle de velocidade do motor BLDC.

- **Driver de potência** - corresponde a placa driver X-Nucleo-IHM08M1, que possui uma topologia de comutação em ponte completa para motor BLDC trifásico. Os sinais para acionamento dos MOSFETs de potência são conectados ao *driver gate* L6398, impedindo que uma lógica de comutação errada acione, simultaneamente, as chaves de potência de um mesmo braço. Também são disponibilizados circuitos para condicionamento da leitura dos sinais dos sensores de efeito Hall, corrente nas fases do motor e tensão na fonte;
- **Motor BLDC** - trata-se de um motor trifásico de 36V/350 W com rotor externo ao estator, modelo *outrunner*. As fases estão interligadas em estrela sem neutro aparente e possuem três sensores de posição por efeito Hall. As constantes de tempo, elétrica e mecânica, são equivalentes a 2 ms e 12 ms, respectivamente.

4.2 Estratégia de comutação e controle

Dentre os modelos de circuitos empregados para comutação do motor BLDC – meia ponte, ponte completa, *C-dump*, ponte H e quatro chaves, por exemplo – foi utilizado o modelo de comutação em ponte completa com condução em duas fases, ou seja, as chaves de potência são comutadas para circular corrente por apenas duas fases do estator em um mesmo instante de tempo.

A estratégia de comutação seis passos foi empregada em conjunto com a modulação HLPWM-LON. Essa modulação apresenta menores perdas por chaveamento, quando comparada com a modulação HLPWM-LPWM, isto é, quando as chaves são acionadas por um sinal PWM complementar.

Na Tabela 1 são mostradas as sequências de comutação das fases, para provocar uma rotação no sentido horário do motor, de acordo com os valores de saídas dos sensores de efeito Hall.

Tabela 2. Valores das variáveis utilizadas nas equações (1) e (2).

Equações	Parâmetros		
$C_i(z)$	$K_{pi} = 0,04$	$K_{ii} = 0,002$	$T_i = 0,0002$
$C_w(z)$	$K_{pw} = 0,02$	$K_{iw} = 0,8$	$T_w = 0,001$
$G_w(z)$	$T_f = 0,002$	$T = 0,0002$	—

Na Figura 3 é exibida a estrutura do algoritmo utilizado no sistema embarcado bem como os blocos que correspondem ao driver de potência e ao motor BLDC. A estratégia de comutação é representada pelo bloco **Comutação 6 passos e PWM**, que envia os sinais para comutação das chaves de potência de acordo com o sinal de controle e a posição do rotor. Os demais blocos são descritos nos parágrafos que seguem.

O cálculo da corrente do motor é representado pelo bloco **Calcula corrente**, no qual um sinal de tensão lido pelo conversor A/D é convertido para o valor da corrente do motor. Ao passo que o bloco **Calcula velocidade** representa o algoritmo para o cálculo da velocidade de rotação, se utilizando dos sinais dos sensores de efeito Hall. Sobre esse valor da velocidade é aplicado um filtro passa baixas discreto de primeira ordem, $G_w(z)$, dado por

$$G_w(z) = \frac{(T_f/T)z^{-1}}{1 + (T_f/T - 1)z^{-1}} \quad (1)$$

sendo T_f a constante de tempo do filtro e T o tempo de execução do algoritmo.

A estratégia de controle aplicada foi a de controladores PI em cascata, com o controlador da malha interna, $C_i(z)$, responsável pelo controle da corrente e o controlador da malha externa, $C_w(z)$, pelo controle da velocidade de rotação, Figura 3. Os controladores PI foram discretizados empregando a aproximação de forward, sendo a equação dada por

$$C_i(z) = K_{pi} + K_{ii} \frac{T_i}{z - 1} \quad (2)$$

$$C_w(z) = K_{pw} + K_{iw} \frac{T_w}{z - 1}$$

no qual os subscritos i e w são referentes a corrente do estator e a velocidade de rotação, respectivamente. Para minimizar o efeito de *windup* da ação integral, foi empregado o método *clamping* para ambos os controladores. A saída do controlador $C_i(z)$ foi limitada entre 0 e 1, o que corresponde ao valor de *duty cycle* entre 0% e 100%.

As variáveis T_i e T_w , 0,2 ms e 1 ms, são referentes ao período de tempo para a execução de cada controlador, sendo escolhido um valor, aproximadamente, dez vezes mais rápido que as constantes de tempo elétrica e mecânica do motor. Por sua vez, a escolha dos ganhos dos controladores foi realizada de forma a apresentar uma resposta com sobressinal inferior a 10% e tempo de subida superior a 200 ms, objetivando uma resposta menos agressiva quando comparado com o sistema em malha aberta.

Na Tabela 2 estão os valores para as variáveis citadas nas equações (1) e (2).

Figura 4. Modelo em blocos Simulink para simulação do sistema.

Figura 5. Subsistemas internos ao bloco Planta.

5. DESENVOLVIMENTO DOS MODELOS

Para a aplicação da metodologia MBD, o conhecimento das características técnicas do motor BLDC bem como do driver de potência e dos periféricos utilizados na placa de controle, são de fundamental importância para o desenvolvimento de modelos que representem o comportamento do sistema.

Um aspecto importante quando se deseja a geração automática do código, a partir dos modelos realizados, é a separação entre os modelos que representam o algoritmo da solução dos demais elementos do sistema. Essa separação está representada na Figura 3, blocos no interior da linha tracejada, e na figura 4, bloco **Microcontrolador**.

5.1 Modelos para simulação

A modelagem do sistema foi realizada se utilizando das ferramentas disponibilizadas pela MathWorks. Na Figura 4 são exibidos quatro blocos ou subsistemas: **Comandos** que modela os sinais para a velocidade de referência e da carga empregada sobre o motor; **Planta** que modela o comportamento da fonte de alimentação, driver de potência e motor BLDC; **Microcontrolador** que modela a lógica de comutação e controle; e o **Gráficos**, utilizado para exibir e armazenar os valores como velocidade de rotação, sinais de controle, *duty cycle* e outros.

Na Figura 5 são exibidos três subsistemas internos ao bloco **Planta**. O bloco **Fonte DC** corresponde a uma fonte de tensão ideal com valor de saída de +20 Vcc. Ao passo que o bloco **Driver IHM08M1** modela o comportamento da comutação em ponte completa, do *driver gate* L6398 e dos circuitos para condicionamento dos sinais de corrente do motor e da tensão da fonte. Por fim, o bloco **Motor BLDC** modela o comportamento do motor BLDC trifásico por intermédio do elemento *Brushless DC Motor* disponibilizado no Simscape.

Os subsistemas internos ao bloco **Microcontrolador** são mostrados na Figura 6. A estratégia de comutação seis passos foi modelada pelo bloco **Comutacao 6 passos**. Enquanto o bloco **Controlador** incorpora a lógica do cálculo

Figura 6. Subsistemas internos ao bloco Microcontrolador.

da corrente do motor, do filtro passa baixa, $G_w(z)$ e da estratégia de controle PI em cascata, $C_i(z)$ e $C_w(z)$.

Os blocos **Comandos** e **Gráficos** são do tipo *subsystem*, enquanto os blocos **Planta** e **Microcontrolador** são do tipo *model*. Essa subdivisão em *subsystem* e *model* é definida como *componentization* dentro do ambiente MathWorks. O uso da *componentization* apresenta diversas vantagens como o desenvolvimento de um modelo referenciado independente dos modelos que os utilizam, produção de código que reflita a estrutura do *model*, adicionar hierarquia para organizar e simplificar visualmente um modelo, dentre outras.

5.2 Para geração automática do código

Para geração automática do código um novo arquivo de modelo do Simulink foi criado, Figura 7. Esse novo arquivo foi utilizado apenas para produzir o código da solução, não sendo possível realizar a sua simulação.

Na Figura 7 é verificada a existência dos blocos **Controlador** e **Comutacao 6 passos**, esses correspondem exatamente aos subsistemas existentes no bloco **Microcontrolador**. Além disso, outros subsistemas foram desenvolvidos para realizar a configuração dos periféricos e fornecer/receber os sinais referentes ao **Controlador** e a **Comutacao 6 passos**, são eles:

- **Projeto e Inicializacao** referentes a configuração e inicialização dos periféricos da placa NUCLEO-F401RE;
- **Sinal Velocidade_Ref** para fornecer o sinal de referência para a velocidade;
- **ADC e TIM10** que incorpora o código para a leitura do A/D, tratamento da interrupção do Timer e execução dos blocos **Controlador** e **Valor PWM**;
- **Hall e RPM** que compreende as funções para leitura dos sinais dos sensores de efeito Hall, cálculo da velocidade de rotação e execução dos blocos **Comutacao 6 passos** e **Gates MOSFETs**;
- **Valor PWM** que incorpora as funções para atualização do valor do *duty cycle*;

Figura 7. Modelo em blocos Simulink utilizado para a geração automática de código.

- **Gates MOSFETs** que realiza a atualização do estado dos pinos referentes ao acionamento das chaves de potência.

À exceção do bloco **Sinal Velocidade_Ref**, em todos os outros foram utilizados elementos como *Matlab Function* e/ou *S-Function* para integrar a biblioteca STM32HAL e a biblioteca desenvolvida pelos autores, durante a produção automática do código da solução.

Outra informação representada na Figura 7 é a periodicidade com que a lógica de comutação e controle serão executadas no sistema embarcado. Observa-se que o algoritmo correspondente aos blocos **Comutacao 6 passos** e **Gates MOSFETs** será executado de maneira assíncrona, devido ao sinal **IRQ_TIM2_TRG** que equivale a interrupção de hardware do periférico *Timer 2 Capture* gerada no momento em que ocorre uma transição dos sinais de saída dos sensores de efeito Hall. Por outro lado, os blocos **Controlador** e **Valor PWM** serão executados numa periodicidade de 2 kHz, devido a interrupção de hardware do periférico *Periodic Timer 10*, ocorrida no momento de estouro do timer e representado pelo sinal **IRQ_TIM10**.

6. SIMULAÇÃO E EXPERIMENTOS

As simulações foram realizadas utilizando os modelos representados nas Figuras 4 - 6. Na primeira etapa, o objetivo foi realizar a validação do modelo da planta em conjunto com a estratégia de comutação 6 passos, mediante simulações com o sistema em malha aberta. Posteriormente, foi adicionada a estratégia de controle ao bloco **Controlador** e efetuadas novas simulações, estando agora o sistema em malha fechada.

Em ambas as etapas das simulações foram realizados os testes em MIL (Figura 1.a) e em PIL (Figura 1.b) sendo os valores das entradas e saídas do controlador salvos. O propósito foi analisar a consistência numérica entre os sinais quando obtidos via simulação do modelo do controlador e via algoritmo de controle executado pelo microcontrolador.

Figura 8. Plataforma experimental para acionamento do motor BLDC.

Figura 9. Velocidade de rotação e sinal de entrada com o sistema em malha aberta.

Para a realização dos experimentos foi utilizada plataforma experimental mostrada na Figura 8, correspondendo a arquitetura citada na seção 4. Tal qual nas simulações, foram efetuados testes com o sistema em malha aberta e em malha fechada, tendo os valores da velocidade de rotação, sinais de controle e *duty cycle* salvos.

Nas Figuras 9 e 10 são exibidos tanto os resultados simulados quanto os experimentais, para o sistema em malha aberta e em malha fechada, respectivamente. Na Figura 9, são mostrados os gráficos para a velocidade de rotação e sinal de entrada quando um degrau com amplitude de 9% é aplicado ao sinal de entrada, estando o sistema em malha aberta. Enquanto na Figura 10, são mostrados os gráficos para a velocidade de rotação e os sinais de controle, saída do controlador PI de velocidade e saída do controlador PI de corrente, na condição em que um degrau de 30 rpm é aplicado ao sinal de referência, estando o sistema em malha fechada.

Tanto nas simulações quanto nos experimentos o motor BLDC foi atuado sem carga.

Resposta em malha fechada

Figura 10. Velocidade de rotação e sinal de controle com o sistema em malha fechada.

7. ANÁLISE DOS RESULTADOS

Nas simulações em MIL e em PIL, os valores das entradas e saídas do bloco Microcontrolador foram comparados mediante o uso da ferramenta Simulink Data Inspector, que realiza essa comparação entre os sinais tendo como restrição, por exemplo, um valor de erro absoluto. Caso alguma restrição seja violada, a ferramenta informa que a análise foi reprovada. Para a análise dos sinais em MIL e PIL foi configurado o valor para o erro absoluto igual a zero, pois era esperada precisão de 1 bit. Como não ocorreu violação da restrição, foi constatado que os sinais capturados em MIL e em PIL eram exatamente iguais.

Para comparar os valores simulados e experimentais foi empregado o cálculo do erro quadrático médio normalizado (EQMN), dado por

$$EQMN = 1 - \frac{\|x_{exper} - x_{modelo}\|_2^2}{\|x_{exper} - \bar{x}_{exper}\|_2^2}, \quad (3)$$

sendo x_{exper} os dados experimentais, x_{modelo} os dados provenientes do modelo, $\bar{x}_{exper}$ o valor médio dos dados experimentais e $\|x\|_2$ que equivale a norma 2 de um dado vetor x .

Os valores do *EQMN* para os vetores de dados da velocidade de rotação e sinal de entrada (*duty cycle*), para o sistema em malha aberta, e da velocidade de rotação, sinal de controle do PI de velocidade e do PI de corrente (*duty cycle*), para o sistema em malha fechada, são mostrados na Tabela 3. Como era de se esperar, com base nas curvas mostradas nos gráficos, os valores de *EQMN* para os sinais

Tabela 3. Valor do EQMN entre os resultados experimentais e simulados.

Valores do Erro quadrático médio normalizado (EQMN)			
	Velocidade	PI velocidade	Duty cycle
Malha aberta	0,9689	—	0,9888
Malha fechada	0,9836	0,9647	0,9484

Tabela 4. Consumo dos recursos de memória.

Conjuntos de funções	FLASH		SRAM	
	Bytes	%	Bytes	%
Biblioteca desenvolvedor	1.098	5,8	128	4,6
Comutação 6 passos	332	1,8	184	6,6
Controlador	1.316	7,0	268	9,6
Demais funções	151	0,8	228	8,1
Projeto STM32HAL	15.913	84,6	1.997	71,2
Total	18.810	100,0	2.805	100,0

avaliados são próximos do valor 1, significando que o modelo simulado reproduz, com boa aproximação, os valores experimentais tanto com o sistema funcionando em malha aberta quanto em malha fechada.

É possível verificar, nas Figuras 9 e 10, que a dinâmica de aumento da velocidade de rotação possui um comportamento diferente da dinâmica de redução dessa velocidade. Essa diferença na dinâmica é decorrente, principalmente, do momento angular imposto pelo conjunto roda e pneu. Com relação ao comportamento do sistema em malha fechada, verifica-se que o mesmo apresenta uma resposta mais lenta, quando comparado ao sistema em malha aberta, atendendo, portanto, ao requisito do projeto do controlador.

Nos dados experimentais, foi observada uma oscilação de aproximadamente 5 rpm em torno do valor de referência. Essas oscilações na velocidade de rotação são ocasionadas, principalmente, pelas variações no conjugado magnético decorrente do modo de comutação do motor e de fatores construtivos. Do mesmo modo, uma distribuição não simétrica dos sensores de efeito Hall afetaria o valor da velocidade de rotação medido.

Quanto ao código gerado para a aplicação, são exibidos na Tabela 4 o consumo dos recursos de memória FLASH e SRAM realizado pelo algoritmo de comutação seis passos, controlador e códigos legados. À exceção do Projeto STM32HAL, os conjuntos de funções que consomem maior percentual dos recursos de memória são os algoritmos de controle e de comutação seis passos, respectivamente. No que tange ao tempo de execução dos algoritmos, verifica-se que são necessários 9 μ s para realizar a leitura do valor da corrente (ADC + DMA), 70 μ s para o algoritmo de controle e 30 μ s para o de comutação seis passos, Figura 11. Assim, ficam disponíveis cerca de 91 μ s para a realizar outras tarefas, antes que um novo passo de cálculo para o controlador ocorra.

8. CONCLUSÃO

Os modelos da planta e do controlador apresentaram uma reposta satisfatória quando comparados com os resultados experimentais. A geração do código da solução, a partir do modelo do controlador, permitiu reduzir o tempo do desenvolvimento e testes dos algoritmos. Assim, foi possível aplicar a metodologia MBD em todas as etapas do

Figura 11. Tempo de execução dos algoritmos.

desenvolvimento de um controlador de velocidade para motor BLDC.

Em trabalhos futuros, os autores pretendem analisar outras estratégias de comutação e de controle bem como otimizações para reduzir o consumo de memória e o tempo de execução dos algoritmos.

AGRADECIMENTOS

Os autores agradecem as agências de fomento CNPq e CAPES pelo apoio financeiro prestado.

REFERÊNCIAS

- Aarenstrup, R. (2015). *Managing Model-Based Design*. MathWorks, Inc.
- Hat, M.K., Ibrahim, B.S.K.K., Mohd, T.A.T., and Hassan, M.K. (2015). Model based design of pid controller for bldc motor with implementation of embedded arduino mega controller. In *ARNP Journal of Engineering and Applied Sciences*, volume 10.
- Lin, J. (2011). Measuring return on investment of model-based design. Technical article, MathWorks.
- MathWorks (2014). Why adopt model-based design for embedded control software development? Technical article, MathWorks.
- MathWorks (2018). Validation and verification for system development. <https://www.mathworks.com/help/ecoder/gs/v-model-for-system-development.html#brufb98-9/>. [Acessado em 14/03/2019].
- NXP (2017). Model-based design tool-box enabling fast prototyping and design. <https://www.nxp.com/docs/en/supporting-information/MBDT-General-Presentation.pdf>, [Acessado em 04/03/2019].
- O'Sullivan, D., Sorensen, J., and Murray, A. (2015). Model-based design streamlines embedded motor control system development. Technical article, Analog Devices Inc.
- Potnuru, D., K., A.M., and Ch., S. (2018). Design and implementation methodology for rapid control prototyping of closed loop speed control for bldc motor. In *Journal of Electrical Systems and Information Technology*, volume 5, 99–111.
- Research, P.M. (2017). Global frameless brushless dc motor market to surpass US\$12 billion by 2025. <https://www.persistencemarketresearch.com/mediarelease/frameless-brushless-dc-motors-market.asp>, [Acessado em 20/02/2019].
- Tariq, M., Bhattacharya, T.K., Varshney, N., and Rajapan, D. (2016). Fast response antiwindup pi speed controller of brushless dc motor drive: Modeling, simulation and implementation on dsp. In *Journal of Electrical Systems and Information Technology*, volume 3, 1–13.