



UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

JORGE LUIZ WATTES OLIVEIRA JUNIOR

DESENVOLVIMENTO DE PLATAFORMA EMULADORA DE TURBINA EÓLICA
PARA ESTUDOS DE ALGORITMOS DE MPPT EÓLICOS INTELIGENTES

FORTALEZA

2016

JORGE LUIZ WATTES OLIVEIRA JUNIOR

DESENVOLVIMENTO DE PLATAFORMA EMULADORA DE TURBINA EÓLICA
PARA ESTUDOS DE ALGORITMOS DE MPPT EÓLICOS INTELIGENTES

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Ceará, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica

Orientador: Paulo Peixoto Praça
Coorientador: Demercil de Souza Oliveira Júnior

FORTALEZA

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

- O47d Oliveira Junior, Jorge Luiz Wattes.
Desenvolvimento de plataforma emuladora de turbina eólica para estudos de algoritmos de MPPT eólicos inteligentes / Jorge Luiz Wattes Oliveira Junior. – 2016.
208 f. : il. color.
- Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Fortaleza, 2016.
Orientação: Prof. Dr. Paulo Peixoto Praça.
Coorientação: Prof. Dr. Demercil de Souza Oliveira Junior.
1. MPPT. 2. Energia eólica. 3. Redes Neurais Artificiais. 4. Aprendizagem por Reforço. I. Título.
CDD 621.3
-

JORGE LUIZ WATTES OLIVEIRA JÚNIOR

**DESENVOLVIMENTO DE PLATAFORMA EMULADORA DE TURBINA
EÓLICA PARA ESTUDOS DE ALGORÍTMOS DE MPPT
EÓLICOS INTELIGENTES**

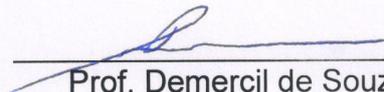
Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Ceará, como requisito à obtenção do título de Mestre em Engenharia Elétrica. Área de Concentração: Sistema de Energia Elétrica.

Aprovada em: 29/06/2016

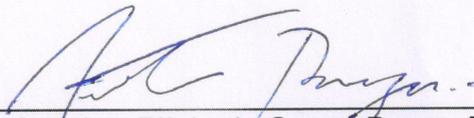
BANCA EXAMINADORA



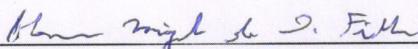
Prof. Paulo Peixoto Praça, Dr. (Orientador)
Universidade Federal do Ceará (UFC)



Prof. Demercil de Souza Oliveira Júnior, Dr.
Universidade Federal do Ceará (UFC)



Prof. Arthur Plínio de Souza Braga, Dr.
Universidade Federal do Ceará (UFC)



Prof. Hermínio Miguel de Oliveira Filho, Dr.
Universidade da Int. Internacional da Lusofonia Afro-Brasileira (UNILAB)

À Deus.

Aos meus pais, Ruth e Jorge.

À minha companheira, Lais.

AGRADECIMENTO

Primeiramente a Deus por me guiar nesta minha caminhada e pelas oportunidades que me foram concedidas.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro com a manutenção da bolsa de auxílio.

À minha mãe, pelo apoio, suporte e todos os esforços feitos, que permitiram hoje chegar onde cheguei.

À minha namorada Lais Policarpo por todos os milhares motivos que não cabem nessas folhas.

Ao meu pai, por servir de motivação para que eu sempre alcance mais longe.

Aos professores Paulo Praça, Demercil Oliveira Júnior e Arthur Plínio Braga, pela orientação, paciência e disponibilidade durante todo este tempo.

Aos colegas e amigos do laboratório e do departamento de engenharia elétrica e do GPEC em especial ao Mcs. Eng. Marcus Anderson Almeida Bezerra

A todos os outros professores do Departamento de Engenharia Elétrica da UFC e aos demais funcionários do departamento responsáveis diretamente ou indiretamente pela minha formação acadêmica.

“Eu não creio que exista algo mais emocionante para o coração humano do que a emoção sentida pelo inventor quando ele vê alguma criação da mente se tornando algo de sucesso. Essas emoções fazem o homem esquecer comida, sono, amigos, amor, tudo.”

(Nikola Tesla)

“A primeira ideia que uma criança precisa ter é a da diferença entre o bem e o mal. E a principal função do educador é cuidar para que ela não confunda o bem com a passividade e o mal com a atividade.”

(Maria Montessori)

RESUMO

A dinâmica dos ventos dentro do contexto da geração eólica de pequeno porte é uma problemática uma vez que a velocidade de operação da máquina deve acompanhá-lo para poder extrair a máxima potência do vento. Neste trabalho, são propostos o projeto e desenvolvimento de uma bancada emuladora de turbina eólica, bem como modelos de algoritmos de rastreamento de máxima potência baseados em Redes Neurais Artificiais e aprendizagem por reforço. O sistema de emulação tem como objetivo permitir a avaliação experimental de algoritmos previamente validados via simulação, já os algoritmos propostos visam alcançar uma boa performance frente a algoritmos clássicos. Além de revisão bibliográfica, foram implementadas simulações computacionais em software PSIM e Matlab, bem como o projeto, desenvolvimento e validação da bancada emuladora de turbina eólica baseada em motor de corrente contínua. São apresentados todos os passos de projeto dos conversores do emulador e do controlador de carga responsável pela realização do rastreamento de máxima potência, bem como todo o material necessário para a reprodução do trabalho, na forma de apêndices. Além da bancada emuladora, dois algoritmos são propostos nesse trabalho: um baseado numa modificação do algoritmo perturba e observa, através da inserção de uma rede neural que define o tamanho da perturbação; já o segundo baseia-se em recentes algoritmos de aprendizagem por reforço do tipo Atuador-Critico (CACLA), que ainda não haviam sido utilizados com essa finalidade.

Palavras-chave: MPPT, Energia eólica, Redes Neurais Artificiais, Aprendizagem por Reforço.

ABSTRACT

The dynamics of the wind within the wind power small context is problematic since the operating speed of the machine must accompany him in order to extract the maximum wind power. In this paper, we propose the design and development of an emulator bench wind turbine, as well as models of maximum power tracking algorithms based on neural networks and reinforcement learning. The emulation system aims to allow the algorithms of experimental evaluation previously validated by simulation, since the proposed algorithms aim to achieve a good performance compared to classical algorithms. In addition to literature review, computer simulations were implemented in PSIM and Matlab software, as well as the design, development and validation of emulator bench wind turbine based on DC motor. They present all design steps the emulator converters and charge controller responsible for carrying out the maximum power tracking, as well as all the material necessary for reproduction of the work in the form of appendices. In the emulator bench, two algorithms are proposed in this work: one based on a modification algorithm and disturbs observed through insertion of a neural network that defines the size of the perturbation; already the second is based on recent learning algorithms for reinforcement type Actuator-Critical (CACLA) , which had not been used for this purpose.

Keywords: MPPT, Wind energy, Artificial neural networks, Reinforcement learning.

LISTA DE ILUSTRAÇÕES

Figura 1 – Dados de 2014 da geração de energia elétrica no Brasil.....	19
Figura 2 – Gráfico $C_p(\beta,\lambda)$ (esq) e potência extraída em função do vento e da rotação (dir) ..	26
Figura 3 – Anemômetro mecânico (esquerda) e ultrassônico (direita) da <i>Logotronic</i>	28
Figura 4 – Operação do algoritmo P&O eólico	29
Figura 5 – Problemática do tamanho de passo do P&O	30
Figura 6 – Modelo de neurônio biológico	32
Figura 7 – Modelo de neurônio artificial.....	32
Figura 8 – Modelo de arquitetura <i>Multi-Layer Perceptron</i> de Rede Neural	35
Figura 9 – Funções de ativação degrau e linear por partes.....	38
Figura 10 – Função de ativação sigmoide logística e sua derivada.....	39
Figura 11 – Função de ativação tangente hiperbólica e sua derivada	39
Figura 12 – Diagrama de interação entre agente e ambiente.....	40
Figura 13 – Gráfico de aprendizagem por Programação Dinâmica	42
Figura 14 – Diagrama do sistema experimental	46
Figura 15 – Modelo de encoder Hengstler.	46
Figura 16 – Diagrama do sistema de emulação.....	48
Figura 17 – Esquemático do circuito Chopper de acionamento do motor CC.....	50
Figura 18 – Esquemático e tabela verdade do opto acoplador HCPL 3120	52
Figura 19 – Diagrama da malha de controle de corrente de armadura do MCC.....	52
Figura 20 - Manipulação do valor do sinal de corrente de armadura do MCC	53
Figura 21 – Diagrama de Bode de malha aberta sem compensador da corrente do MCC.....	54
Figura 22 – Diagrama de Bode de malha de aberta com compensador da corrente do MCC..	55
Figura 23 - Diagrama de Bode de malha de fechada da corrente de armadura do MCC.....	56
Figura 24 – Resposta ao degrau unitário da malha de corrente de arma do MCC.....	56
Figura 25 - Diagrama de controle de torque do emulador de turbine eólica.	57
Figura 26 – Diagrama de cálculo da corrente de referência do sistema emulador.....	58
Figura 27 – Diagrama do sistema de cálculo de corrente de referência do emulador	58
Figura 28 – Perfil de velocidade e vento Vento_II.....	59
Figura 29 – Perfil de velocidade e vento Vento_III	60
Figura 30 – Perfil de velocidade e vento Vento_IV	60
Figura 31 – Diagrama do sistema controlador de carga tipo Boost.....	62

Figura 32 – Gráfico da corrente em função da rotação	64
Figura 33 – Gráfico da variação de corrente do indutor em função da rotação	64
Figura 34 – Diagrama das malhas de controle do conversor de gerenciamento de carga.....	66
Figura 35 - Manipulação do valor do sinal de corrente do indutor do conversor Boost	67
Figura 36 - Diagrama de Bode de malha aberta sem compensador da corrente no indutor do Boost.....	68
Figura 37 – Diagrama de Bode de malha de aberta compensada da corrente no indutor do Boost	69
Figura 38 - Diagrama de Bode de malha fechada da corrente de armadura do MCC.....	70
Figura 39 – Resposta ao degrau unitário da malha de corrente no indutor do conversor Boost	70
Figura 40 - Diagrama de Bode de malha aberta sem compensador da rotação do eixo	71
Figura 41 – Algoritmo do PI digital com Anti-Windup	72
Figura 42 - Diagrama de Bode de malha de aberta compensada da velocidade de rotação	73
Figura 43 – Diagrama de Bode de malha fechada da velocidade de rotação do eixo	73
Figura 44 – Resposta ao degrau unitário da malha de velocidade de rotação no eixo	74
Figura 45 – Módulo de interface USB baseado em FT232RL	75
Figura 46 – Interface Homem Máquina desenvolvida	76
Figura 47 – Rede neural utilizada no NASPO.....	79
Figura 48 – Fluxograma do MPPT NASPO.....	81
Figura 49 – Rede neural utilizada no MPPT CACLA.....	84
Figura 50 – Fluxograma do MPPT CACLA	85
Figura 51 – Diagrama de simulação da malha de corrente de armadura do MCC.....	88
Figura 52 – Gráfico da corrente de armadura do MCC com degraus na referência.....	88
Figura 53 – Detalhe de chaveamento da corrente de armadura do MCC.....	89
Figura 54 – Diagrama do cálculo da referência de corrente de armadura do MCC	90
Figura 55 – Gráfico comparativo dos valores de c_p e Λ , teóricos e calculados	90
Figura 56 – Gráficos de potência e torque da simulação de referência de corrente do MCC..	91
Figura 57 – Gráfico do valor de corrente de armadura de referência do MCC em função da rotação	91
Figura 58 – Gráficos de C_p em função da velocidade de rotação.....	92
Figura 59 - Diagrama de simulação da malha de corrente do indutor do conversor Boost.....	93

Figura 60 – Gráfico da corrente no indutor e das tensões de entrada e saída do controlador de carga.....	93
Figura 61 - Detalhe de chaveamento da corrente no indutor do conversor Boost.....	94
Figura 62 - Diagrama de simulação da malha de rotação do PMSG.....	95
Figura 63 – Gráfico de atuação do <i>Ant-Windup</i> na malha de rotação do controlador de carga.....	96
Figura 64 – Detalhe do comportamento dinâmico da malha de rotação	96
Figura 65 – Bancada experimental	97
Figura 66 – Malha experimental de corrente no indutor do conversor Boost	98
Figura 67 – Detalhe da dinâmica experimental da corrente no indutor do conversor Boost ...	99
Figura 68 - Malha experimental de corrente no de armadura do MCC.....	100
Figura 69 - Detalhe da dinâmica experimental da corrente no de armadura do MCC	100
Figura 70 – Resposta da malha de velocidade a degraus de referência.....	101
Figura 71 - Resposta da malha de velocidade a degraus de referência e variações do torque de carga.....	101
Figura 72 – Potência de entrada do conversor Boost extraída do emulador	102
Figura 73 – Corrente do indutor do conversor Boost extraída do emulador	103
Figura 74 – Gráfico comparativo da corrente de armadura do MCC e sua referência.....	104
Figura 75 – Perfil de potência extraída com emulação de vento variável.....	104
Figura 76 – Simulação de rotação fixa para o perfil Vento_II	105
Figura 77 - Simulação de rotação fixa para o perfil Vento_III	106
Figura 78 - Simulação de rotação fixa para o perfil Vento_IV	106
Figura 79 – Simulação do P&O para o perfil Vento_I.....	107
Figura 80 - Simulação do P&O para o perfil Vento_II	108
Figura 81 - Simulação do P&O para o perfil Vento_III.....	108
Figura 82 - Simulação do P&O para o perfil Vento_IV	109
Figura 83 - Simulação do MPPT NASPO para o perfil Vento_I	110
Figura 84 - Simulação do MPPT NASPO para o perfil Vento_II.....	110
Figura 85 - Simulação do MPPT NASPO para o perfil Vento_III.....	111
Figura 86 - Simulação do MPPT NASPO para o perfil Vento_IV	111
Figura 87 - Simulação do MPPT CACLA para o perfil Vento_I.....	112
Figura 88 - Simulação do MPPT CACLA para o perfil Vento_II	112
Figura 89 - Simulação do MPPT CACLA para o perfil Vento_III	113
Figura 90 - Simulação do MPPT CACLA para o perfil Vento_IV	113

Figura 91 – Resultado experimental de rotação fixa para o perfil Vento_I	115
Figura 92 - Resultado experimental de rotação fixa para o perfil Vento_II.....	115
Figura 93 - Resultado experimental de rotação fixa para o perfil Vento_III	116
Figura 94 - Resultado experimental de rotação fixa para o perfil Vento_IV	117
Figura 95 - Resultado experimental do P&O para o perfil Vento_I.....	118
Figura 96 - Resultado experimental do P&O para o perfil Vento_II	118
Figura 97 - Resultado experimental do P&O para o perfil Vento_III.....	119
Figura 98 - Resultado experimental do P&O para o perfil Vento_IV.....	119
Figura 99 - Resultado experimental do MPPT NASPO para o perfil Vento_I.....	120
Figura 100 - Resultado experimental do MPPT NASPO para o perfil Vento_II.....	120
Figura 101 - Resultado experimental do MPPT NASPO para o perfil Vento_III.....	121
Figura 102 - Resultado experimental do MPPT NASPO para o perfil Vento_IV	121
Figura 103 - Resultado experimental do MPPT CACLA para o perfil Vento_I.....	122
Figura 104 - Resultado experimental do MPPT CACLA para o perfil Vento_II	122
Figura 105 - Resultado experimental do MPPT CACLA para o perfil Vento_III	123
Figura 106 - Resultado experimental do MPPT CACLA para o perfil Vento_IV	123

LISTA DE TABELAS

Tabela 1 – Dados da geração de energia elétrica de 2014.....	19
Tabela 2 – Dados de ensaio do motor CC	49
Tabela 3 – Dados da turbina eólica fictícia emulada.....	49
Tabela 4 – Dados nominais do conversor Chopper para acionamento do motor CC.....	50
Tabela 5 – Dados do sistema de geração Gerar 246 da Enersud.....	61
Tabela 6 – Dados de ensaio do gerador Gerar 246 – Enersud.....	62
Tabela 7 – Dados nominais do conversor Boost	63
Tabela 8 – Parametros do MPPT P&O.....	79
Tabela 9 – Parametros do MPPT NASPO.....	80
Tabela 10 – Parametros do MPPT CACLA	83
Tabela 11 – Dados de ensaio da corrente de armadura do emulador.	103
Tabela 12 – Dados comparativos de simulação dos algoritmos implementados	114
Tabela 13 – Dados comparativos entre os algoritmos e perfis de vento	124

LISTA DE ABREVIATURAS E SIGLAS

CACLA	<i>Continuous Actor-Critic Learning Automato</i> (Autômato de aprendizagem Atuador-Crítico Contínuo)
CC	Corrente Contínua
DSP	<i>Digital Signal Process</i> (Processamento Digital de Sinais)
MCC	Motor de Corrente Contínua
MLP	<i>Multi-Layer Perceptron</i> (Perceptron de Multi Camadas)
MPP	<i>Maximum Power Point</i> (Ponto de Máxima Potência)
MPPT	<i>Maximum Power Point Tracker</i> (Rastreador de Ponto de Máxima Potência)
NASPO	<i>Neural Adaptive Step Size Perturb and Observe</i> (Perturba e Observa com Passo Adaptativo Neural)
P&O	<i>Perturb and Observe</i> (Perturba e observa)
PI	Proporcional-Integral
PMSG	Permanent Magnet Synchronous Generator (Gerador Síncrono de Ímã Permanente)
PSF	<i>Power Signal Feedback</i> (Realimentação do Sinal de Potência)
PWM	<i>Pulse Width Modulation</i> (Modulação por Largura de Pulso)
QEI	<i>Quadrature Encoder Interface</i>
RL	<i>Reinforcement Learning</i> (Aprendizagem por reforço)
RPM	Rotações Por Minuto
TD	Temporal Difference (Diferença temporal)
USB	<i>Universal Serial Bus</i> (Barramento Serial Universal)
WECS	<i>Wind Energy Converter System</i> (Sistema de Conversão de Energia Eólica)
WSM	<i>Wind Speed Measurement</i> (Medição de Velocidade do Vento)
WTE	<i>Wind Turbine Emulator</i> (Emulador de Turbina Eólica)

LISTA DE SIMBOLOS

λ	Razão da velocidade da ponta da pá (Tip Speed Ratio)
β	Angulo de passo das pás (Blade Pitch Angle)
E_w	Energia cinética total dos ventos
M_{ar}	Massa de ar
V_{ar}	Velocidade do vento (Massa de ar)
P_w	<i>Potência total disponível no vento</i>
ρ	Densidade do ar
$A_{pás}$	Área total varrida pelas pás de uma turbina
C_p	Coefficiente de potência
v	<i>Velocidade do vento</i>
r	Raio das pás
ω_m	Velocidade angular de rotação das pás
$C_{\omega i}$	controlador PI com anti wind-up da malha de rotação (WTE)
H_{iL}	Ganho de ajuste da referência de corrente (WTE)
C_{iLd}	controlador PI da malha de corrente (WTE)
K_{Lpwm}	ganho de modulação PWM (WTE)
G_{iLd}	Planta de corrente (WTE)
$G_{\omega i}$	Planta de rotação (WTE)
H_{iL}	Ganho de realimentação de corrente (WTE)
H_{ω}	Ganho de realimentação de rotação (WTE)

SUMÁRIO

1	INTRODUÇÃO	18
1.1	Motivação	20
1.2	Objetivos Principal e Específicos	20
1.3	Metodologia	21
1.4	Publicações	22
1.5	Estrutura da Dissertação	22
2	REVISÃO BIBLIOGRÁFICA DE ALGORITMOS DE RASTREIO DE MÁXIMA POTÊNCIA	24
2.1	Rastreo de ponto de máxima potência	25
2.2	Algoritmos de rastreo de máxima potência básicos	27
2.3	Revisão de Inteligência Artificial	30
2.3.1	<i>Redes Neurais Artificiais</i>	31
2.3.2	<i>Aprendizagem por Reforço</i>	40
2.4	Comentários parciais	44
3	PLATAFORMA DE VALIDAÇÃO EÓLICA	45
3.1	Sensoriamento da velocidade de rotação	46
3.2	Sistema de Emulação	47
3.2.1	<i>Parâmetros da turbina emulada/Motor de Corrente Contínua</i>	48
3.2.2	<i>Projeto do sistema de acionamento do emulador</i>	49
3.2.3	<i>Projeto de controle do emulador</i>	52
3.3	Perfis de ventos adotados	58
3.4	Sistema de operação do gerador (Boost / PMSG)	61
3.4.1	<i>Projeto do sistema de operação do gerador</i>	62
3.4.2	<i>Projeto as malhas de controle do sistema de gerenciador de carga</i>	66
3.5	Sistema de controle via Matlab	74
3.5.1	<i>Sistema de aquisição e controle</i>	75

3.5.2	<i>Metodologia de implementação do sistema de MPPT</i>	76
3.6	Comentários parciais	77
4	ALGORITMOS MPPT PROPOSTOS	78
4.1	Algoritmo Perturba e Observa de referência	78
4.2	Algoritmo MPPT NASPO	79
4.3	Algoritmo MPPT CACLA	82
4.4	Comentários parciais	86
5	RESULTADOS	87
5.1	Simulações do emulador de turbina eólica	87
5.1.1	<i>Simulações da malha de corrente de armadura do MCC</i>	87
5.1.2	<i>Simulações do cálculo da referência de corrente do conversor chopper</i>	89
5.2	Simulações do controlador de carga do gerador	92
5.2.1	<i>Simulação da malha de corrente do conversor Boost</i>	92
5.2.2	<i>Simulação da malha de rotação do sistema gerador</i>	94
5.3	Validação da bancada experimental	97
5.3.1	<i>Validação da malha de corrente do indutor do conversor Boost</i>	98
5.3.2	<i>Validação da malha de corrente de armadura do MCC</i>	99
5.3.3	<i>Validação da malha de rotação feita pelo controlador de carga</i>	100
5.3.4	<i>Validação do sistema de emulação baseado em MCC</i>	102
5.4	Simulações do algoritmo rastreador de máxima potência	105
5.4.1	<i>Simulações com velocidade de rotação fixa</i>	105
5.4.2	<i>Simulações do MPPT P&O de referência</i>	107
5.4.3	<i>Simulações do MPPT NASPO</i>	109
5.4.4	<i>Simulações do MPPT CACLA</i>	111
5.5	Validação experimental do algoritmo rastreador de máxima potência	114
5.5.1	<i>Resultados experimentais para velocidade de rotação fixa</i>	114
5.5.2	<i>Resultados experimentais do MPPT P&O de referência</i>	117

5.5.3	<i>Resultados experimentais do MPPT NASPO</i>	119
5.5.4	<i>Resultados experimentais do MPPT CACLA</i>	121
5.5.5	<i>Análise experimental comparativa</i>	123
6	CONCLUSÃO	125
7	REFERÊNCIAS BIBLIOGRÁFICAS	126
	APÊNDICE A: ENSAIO DO MOTOR CC	129
	APÊNDICE B: CÓDIGOS MPLAB DO CHOPPER	133
	APÊNDICE C: ESQUEMÁTICOS DO CHOPPER	156
	APÊNDICE D: CÓDIGOS MPLAB DO BOOST	158
	APÊNDICE E: ESQUEMÁTICOS DO BOOST	171
	APÊNDICE F: CÓDIGOS DE COMUNICAÇÃO MATLAB	176
	APÊNDICE G: CÓDIGOS DE MPPT MATLAB	180
	APÊNDICE H: CÓDIGO DE PARÂMETROS DO PROJETO MATLAB	186

1 INTRODUÇÃO

Com o constante aumento nos custos de combustíveis de origem fóssil, e a crescente conscientização da importância dos cuidados com a emissão de poluentes no meio ambiente, a diversificação nas formas de geração tornou-se uma realidade em países desenvolvidos, e atualmente tem sido impulsionada em países em desenvolvimento, dentre os quais o Brasil se inclui. Há algum tempo a Organização das Nações Unidas (ONU), em conjunto com diversos países, vem realizando conferências que visam auxiliar o desenvolvimento de forma sustentável. Essa preocupação com o meio ambiente impulsiona a nível global o desenvolvimento de tecnologias relacionadas a geração de energias renováveis (FREITAS; DATHEIN, 2013).

Tratando-se de fontes de geração de energia elétrica, o comparativo das diversas fontes entre 1973 e 2014 demonstra uma estagnação positiva nas termelétricas a carvão, além de um acentuado crescimento tanto na geração baseada em turbinas a óleo quanto na geração por meio de hidrelétricas, seguindo a tendência da geração brasileira que é cerca de 2/3 proveniente de fontes hídricas. Já a geração termonuclear e as termoelétricas a gás demonstraram um crescimento expressivo. Os crescimentos em percentuais da biomassa sólida, eólica e solar também foram significativos, embora ainda representem apenas 6,3% da energia global (MINISTERIO DE MINAS E ENERGIA, 2015).

A Figura 1 apresenta os valores percentuais de geração de eletricidade para cada fonte, analisando independentemente o Brasil, os 34 membros da Organização para a Cooperação e Desenvolvimento Económico (*Organisation de Coopération et de Développement Économiques* - OCDE), e os demais países. Em âmbito nacional os percentuais de geração diferem dos valores globais. No Brasil cerca de 65,27% da energia elétrica produzida é proveniente da geração hidroelétrica, que para critério de classificação, é renovável. A Tabela 1 apresenta dados de geração de energia elétrica no Brasil.

A geração de energia elétrica através de rios, embora não produza diretamente dióxido de carbono, apresenta outros impactos: (i) a necessidade do alagamento de grandes áreas para a criação dos reservatórios, o que por influencia o microclima da região, alterando a taxa de precipitação, (ii) a necessidade de realocação de famílias que vivem na área de alagamento (NASCIMENTO; STIVAL; FONSECA, 2013), e (iii). em períodos de estiagem, as variações consideráveis dos regimes de chuvas podem levar à subcapacidade do sistema de geração nacional. Com relação a este último ponto, adota-se no país, para contornar estas possíveis subcapacidade da geração hídrica, a ativação de usinas de geração termelétricas, que

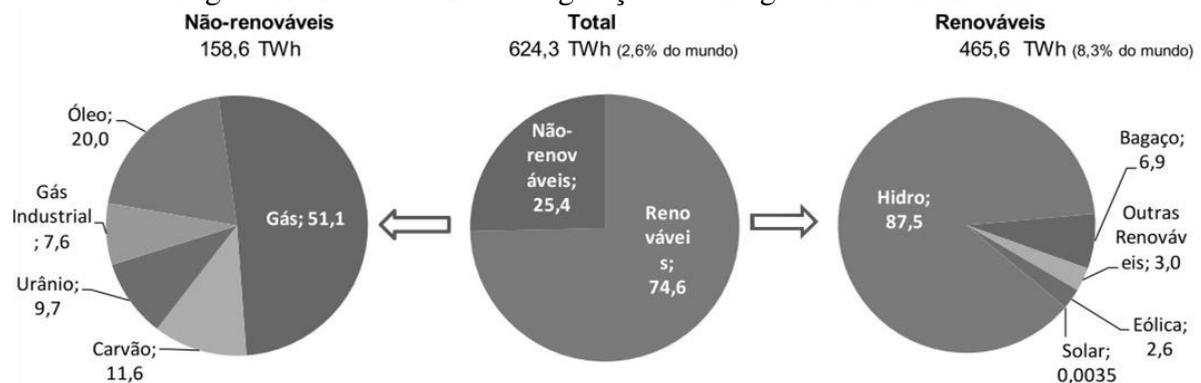
apresentam custos de operação e geração superiores aos de hidrelétricas. Consequente, a solução para as eventuais instabilidades hídricas leva a um aumento no custo de geração que, após a regra normativa N° 547 (ANEEL, 2013), passou a ser revertida aos consumidores através das bandeiras tarifárias.

Tabela 1 – Dados da geração de energia elétrica de 2014

Fonte	Brasil		OCDE		Outros		Mundo	
	1973	2014	1973	2014	1973	2014	1973	2014
Óleo	7,2	5,1	25,4	3,3	23,1	6,0	24,6	4,8
Gás	0,5	14,9	11,6	24,5	14,2	20,2	12,2	22,0
Carvão	1,7	2,9	37,9	31,2	40,9	47,8	38,3	39,2
Urânio	0,0	2,5	4,2	17,9	0,9	4,5	3,3	10,4
Hidro	89	65,2	20,5	13,4	19,3	17,8	21,0	17,1
Outras	1,2	9,4	0,3	9,7	1,6	3,8	0,6	6,6
Biomassa Sólida	1,2	7,4	0,1	3,0	1,6	0,7	0,5	1,9
Eólica	0	2,0	0,01	4,9	0,03	1,8	0,02	3,2
Solar	0	0,003	0	1,3	0	1,0	0	1,1
Geotérmica	0	0	0,1	0,4	0	0,2	0,1	0,3
Total (%)								
Dos quais renováveis	90,6	74,6	20,8	23,1	20,9	21,5	21,6	23,6
Total (TWh)	65 1,1%	624 2,6%	4.472 73,1%	10.623 44,7%	1.579 25,8%	12.534 52,7%	6.115	23.782

Fonte:(MINISTERIO DE MINAS E ENERGIA, 2015)

Figura 1 – Dados de 2014 da geração de energia elétrica no Brasil.



Desta forma, torna-se importante realizar a diversificação da matriz energética, de forma a permitir uma maior robustez do sistema de geração nacional, admitindo que a falta de recursos a um dado tipo de geração não impacte tão fortemente o mercado energético (JUNIOR; RODRIGUES, 2015). Seguindo essa linha, energias alternativas como a eólica vem se desenvolvendo e ganhando espaço dentro da matriz energética, tanto no âmbito nacional quanto

global. Como pontuado por (ABREU *et al.*, 2014), alguns fatores são determinantes para o crescimento desse tipo de geração, dentre eles, destaca-se o desenvolvimento de tecnologias, fator que motiva diversos trabalhos acadêmicos, incluindo este.

Com o objetivo de controlar a turbina eólica a fim de fazê-la operar extraindo sempre a máxima potência disponível no vento, existem algoritmos denominados de rastreadores de ponto de máxima potência (do inglês, *Maximum Power Point Trackers* - MPPT) (Capítulo 2). Para que o sistema tenha máxima eficiência é fundamental que ele sempre esteja operando no ponto de máxima potência. A proposta deste trabalho é implementar uma plataforma de ensaios para a avaliação de algoritmos de MPPT, e propor algoritmos de MPPT a serem avaliados nesta plataforma que contribuam para uma melhoria da geração eólica. As próximas seções organizam esta proposta de pesquisa apresentando: motivação (Seção 1.1), principais objetivos (Seção 1.2), metodologia empregada no trabalho (Seção 1.3), publicações geradas (Seção 1.4) e a estrutura adotada nesta dissertação para descrever o trabalho realizado (Seção 1.5).

1.1 Motivação

Dado o crescimento contínuo dos percentuais de geração eólica, é fundamental que a comunidade acadêmica busque desenvolver conhecimentos e tecnologias que favoreçam o crescimento da mesma. A busca por um melhor aproveitamento do potencial da energia eólica vem despertando diversas pesquisas, entre elas a melhoria dos algoritmos de rastreamento de ponto de máxima potência (MPPT). Dessa forma, propõe-se nesse trabalho: (i) o desenvolvimento de algoritmos de rastreamento de ponto de máxima potência que apresentem melhorias quando comparado com estratégias MPPT tradicionais, no percentual de potência extraída do vento, e (ii) implementar uma plataforma de ensaios que permita avaliar este ganho comparativo entre estratégias MPPT.

1.2 Objetivos Principal e Específicos

Em projetos na área de geração de energia elétrica, dois tópicos tornam-se fundamentais: (i) a potência instalada, ou seja, o valor nominal da potência da usina elétrica e (ii) a eficiência de geração da mesma. Esses fatores são complementares, uma vez que a potência de produção de energia é dada pela ponderação da potência instalada pela eficiência de geração. Dessa forma, utilizar sistemas que apresentem eficiência elevada é imprescindível

para tornar o sistema economicamente viável. Em sistemas hidrelétricos e termelétricos o valor de eficiência é atrelado ao percentual de potência produzida, uma vez que é possível ajustar o fluxo de água e a quantidade de comburente. Já em sistemas eólicos a potência produzida oscila de acordo com a velocidade dos ventos, de forma que o ponto de operação do sistema gerador deve se ajustar para manter uma boa eficiência do sistema.

Partindo desse conceito da importância da eficiência de geração de energia elétrica através dos ventos, o objetivo principal deste trabalho é propor e avaliar algoritmos de rastreamento de máxima potência que, na presença de oscilações do vento, consigam apresentar resultados comparativamente melhores que estratégias MPPT tradicionais. Na busca por este objetivo principal, relaciona-se os seguintes objetivos específicos: Estudar a utilização da capacidade de aprendizagem das Redes Neurais Artificiais (RNA) para a proposição de algoritmos MPPT capazes de adaptar o ponto de operação ideal para cada situação climática;

Além disso, é motivação deste trabalho, a necessidade de aplicação de habilidades práticas interdisciplinares, envolvendo conhecimentos e práticas de áreas como eletrônica de potência, redes neurais artificiais, aprendizagem por reforço, controle digital e fundamentos de geração eólica. Estes conhecimentos se fazem necessários nos diversos passos desse trabalho, inclusive no desenvolvimento da Plataforma de Validação Eólica que emula condições diversas às quais um aerogerador pode ser submetido, permitindo avaliar o desempenho das estratégias MPPT propostas.

Outro objetivo específico deste trabalho, é desenvolver um algoritmo que mantenha o gerador eólico operando no ponto de máxima potência sem a utilização de sensoriamento de vento. É comum sistemas de grande porte realizarem, através de diversos sensores, uma medição de vento, contudo, essa alternativa torna-se muito onerosa em função dos gastos de aquisição e manutenção de tais sensores.

1.3 Metodologia

O desenvolvimento deste trabalho se dá a partir de uma breve introdução da problemática, seguido de uma pesquisa bibliográfica. Com o objetivo de realizar ajustes prévios dos algoritmos e levantar resultados prévios, são feitas simulações computacionais dos circuitos, no software PSIM e dos algoritmos, em linguagem Matlab.

Além disso, o desenvolve-se passo-a-passo a montagem de uma bancada experimental que realize a emulação de turbina eólica, fundamental para a validação experimental do trabalho realizado.

Através do controlador de carga implementado, e de uma comunicação serial feita com um computador dedicado aos cálculos dos algoritmos MPPT, realizam-se os experimentos, simultaneamente mensurando as variáveis de saída do gerador eólico.

Buscando realizar uma comparação quantitativa entre algoritmos clássicos e os propostos neste trabalho, observa-se o valor de potência acumulada, que reflete a energia elétrica extraída do gerador eólico. Dessa forma, é possível verificar percentualmente a possível contribuição do algoritmo, no nível de geração.

1.4 Publicações

Este trabalho é o resultado dos estudos realizados durante o curso de mestrado do autor. Através desses estudos, foi possível a apresentação de dois artigos em congressos internacionais:

1. J. L. Wattes, A. J. S. Dias Júnior, A. P. S. Braga, P. P. Praça, A. U. Barbosa, D. S. Oliveira Júnior. *A Neural Adaptive Step Size Method for Maximum Power Point Tracking in Low Power Wind Turbine Systems*, apresentado em Novembro de 2015 no 13º Congresso Brasileiro de Eletrônica de Potência e 1st Southern Power Electronics Conference (COBEP/SPEC).
2. Jorge L. Wattes, Paulo P. Praça, Arthur P. S. Braga, Antônio J. S. Dias Jr., Allan U. Barbosa. *A Continuous Actor-Critic Maximum Power Point Tracker Applied to Low Power Wind Turbine Systems*, apresentado em Março de 2016 na *The Applied Power Electronics Conference and Exposition 2016* (APEC).

Além disso, houve contribuição mútua no desenvolvimento do trabalho “Desenvolvimento de um emulador de turbina eólica”, realizado como requisito para conclusão de curso de graduação em Engenharia Elétrica na UFC do aluno Allan Uchoa Barbosa, voltado ao projeto e simulação da bancada de emulação de turbina eólica.

1.5 Estrutura da Dissertação

Este trabalho é dividido em seis partes: no capítulo 1 é apresentado o contexto da geração eólica a nível global e nacional, a relevância, e os desafios, de algoritmos MPPT para a eficiência da geração eólica, e um resumo do trabalho proposto.

No Capítulo 2 é realizada uma revisão bibliográfica sobre a geração eólica, bem como de algoritmos de rastreamento de máxima potência e sistemas inteligentes baseados em Redes Neurais Artificiais e Aprendizagem por Reforço.

O Capítulo 3 descreve a plataforma de validação de sistemas eólicos desenvolvida. Através da mesma, torna-se possível realizar a avaliação da performance dos algoritmos de rastreamento de máxima potência propostos. Neste capítulo são apresentadas as especificações do *hardware* (formado por um sistema de emulação de turbina eólica baseado em um motor de corrente contínua, conversores chaveados e o sistema de controle do conversor) e do *software* (desenvolvido nos ambientes PSIM e MATLAB) da plataforma.

No Capítulo 4 são propostos sistemas de rastreamento de máxima potência a serem testados através da plataforma de emulação desenvolvida. Dentre os algoritmos a serem testados, utilizam-se o clássico P&O, bem como dois sistemas de rastreamento de máxima potência baseado nos conceitos de Redes Neurais Artificiais e Aprendizagem por Reforço.

O Capítulo 5 apresenta os resultados obtidos mediante simulação realizadas nos softwares PSIM 9.1 e MATLAB 2015, e a validação experimental destes resultados através da bancada emuladora desenvolvida.

No último capítulo são apresentadas as conclusões acerca da pesquisa desenvolvida, e dos resultados observados, além de propostas de continuação do trabalho desenvolvido.

2 REVISÃO BIBLIOGRÁFICA DE ALGORITMOS DE RASTREIO DE MÁXIMA POTÊNCIA

A denominação “energia eólica”, por definição refere-se a energia cinética das massas de ar que se deslocam pela atmosfera devido a variações de temperaturas e pressões. Sistemas de conversão de energia eólica, por sua vez, são aqueles que possuem a capacidade de converter essa energia cinética em uma forma útil de energia.

A energia eólica possui características intrínsecas que tornam sua exploração algo complexo. Se por um lado os ventos são abundantes em praticamente todo o globo terrestre, por outro, sua variação, tanto no que diz respeito à periodicidade temporal, quanto a uniformidade espacial é o ponto crítico de seu desenvolvimento (BURTON et al., 2001).

Diferente das termelétricas e das hidrelétricas que mantem sua matéria prima de geração armazenada, seja na forma de comburentes ou na forma de energia potencial gravitacional das águas dos lagos, as usinas eólicas possuem uma matéria prima cuja disponibilidade é a variável mais crítica.

A conversão da energia eólica em energia elétrica é feita através de turbinas eólicas. Turbinas eólicas são mecanismos compostos de pás aerodinâmicas que, similar ao princípio das asas de aviões, permitem a conversão da energia cinética das massas de ar em energia mecânica na forma de rotação de um eixo.

Para explicar o funcionamento de uma turbina eólica parte-se do conceito de energia cinética (E_w) aplicado às massas de ar como apresentado em (1), para quantizar a potência (P_w) presente nas massas de ar em deslocamento.

$$E_w = \frac{M_{ar} \cdot V_{ar}^2}{2} \rightarrow P_w = \frac{\rho}{2} \cdot A_{pás} \cdot V_{ar}^3 \quad (1)$$

Sendo $A_{pás}$ a área varrida pelas pás da turbina, ρ a densidade do ar, M_{ar} o valor da massa de ar e V_{ar} a velocidade do vento. Contudo, é impraticável a extração da potência do vento em sua totalidade, uma vez que isso implica em absorver toda a energia cinética, parando o fluxo de ar. Dessa forma, há um coeficiente que determina o percentual de potência aerodinâmica a ser extraída das massas de ar, denominado Coeficiente de potência (C_p).

Albert Betz, um cientista de aerodinâmica alemão, definiu matematicamente que o valor máximo teórico do Coeficiente de potência é igual 16/27, embora não haja turbina que tenha alcançado tal valor (BURTON et al., 2001).

O coeficiente de potência, contudo, não assume um valor fixo para cada turbina, mas sim uma função, determinada pela aerodinâmica das pás e que relaciona a velocidade do vento e a velocidade de rotação da turbina.

2.1 Rastreo de ponto de máxima potência

A relação dada entre a velocidade tangencial da ponta das pás pela velocidade do vento é comumente denominada de relação de velocidade de ponta (*tip speed ratio*), ou simplesmente λ . Em (2) é descrita a definição de λ . Outra grandeza importante é o ângulo de passo das pás, simbolizado por β . Ele é responsável pelo ângulo aerodinâmico de incidência dos ventos nas pás, em turbinas de pequeno porte é comum não haver sistema mecânico para variação de β .

$$\lambda = \frac{r \cdot \omega_m}{v} \quad (2)$$

Sendo r o valor do comprimento das pás, ω_m a velocidade angular em rad/s das pás e v a velocidade do vento.

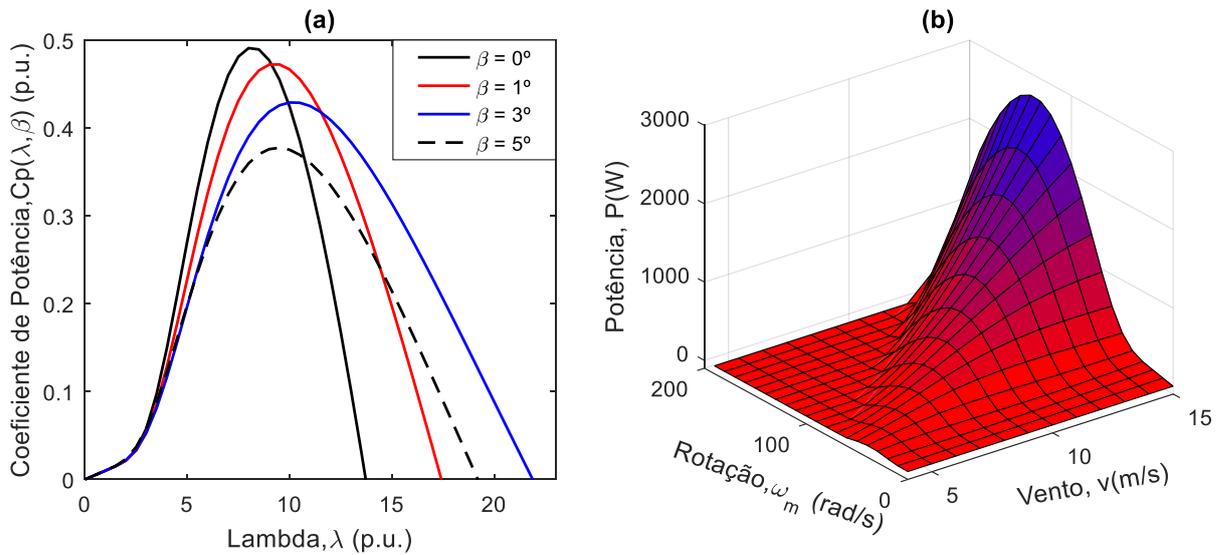
A partir de estudos realizados em turbinas de 3 pás (HEIER; WADDINGTON, 2006a) define uma equação que relaciona o valor do coeficiente de potência com os valores de λ e β . A equação (3) mostra esta função.

$$C_p(\beta, \lambda) = 0,5 \cdot (116\lambda' - 0,4\beta - 5) \cdot e^{-21\lambda'} + 0,01\lambda$$

$$\lambda' = \frac{1}{\lambda + 0,08\beta} - \frac{0,035}{\beta^3 + 1} \quad (3)$$

A Figura 2 (a) apresenta o gráfico da função $C_p(\beta, \lambda)$ para valores de β igual a 0° , 1° , 3° e 5° . Já a Figura 2 (b) apresenta o gráfico da potência extraída relacionada com ω_m e v , para um valor de β nulo.

Figura 2 – Gráfico $C_p(\beta, \lambda)$ (a) e potência extraída em função do vento e da rotação (b)



Fonte: desenvolvido pelo Autor.

Como é possível observar, há um único ponto no qual o valor de λ reflete um valor de coeficiente de potência máximo e, portanto, o ponto de operação onde é possível extrair a maior potência do vento.

Com o objetivo de controlar a turbina eólica a fim de fazê-la operar extraindo sempre a máxima potência disponível no vento, existem algoritmos denominados de rastreadores de ponto de máxima potência, que utilizam diversas técnicas para manter o valor de λ sempre próximo ao ponto no qual o valor de coeficiente de potência é máximo. Na função apresentada, este valor é próximo à 8.

Para que o sistema tenha máxima eficiência é fundamental que ele sempre esteja operando no MPP. Contudo, como definido em (3), além do parâmetro mecânico de rotação, é necessário conhecer o valor atual de vento. Sistemas de grande porte realizam, através de diversos sensores, uma medição de vento, contudo, essa alternativa torna-se muito onerosa em função dos gastos de aquisição e manutenção de tais sensores. Assim, diversos sistemas propõem alternativas para manter o gerador eólico operando no ponto de máxima potência sem a utilização de sensoriamento de vento, tais sistemas são denominados rastreadores de ponto de máxima potência, do inglês *maximum power point trackers* (MPPT).

Algoritmos de rastreo de máxima potência de sistema solares operam através da tensão do barramento de corrente contínua na saída dos painéis fotovoltaicos, pois de acordo com o nível de radiação solar incidente nos painéis, o valor de tensão que reflete o valor de corrente para máxima potência se altera. Diversos estudos já foram desenvolvidos nesse sentido (SUBUDHI; PRADHAN, 2013).

Em sistemas de geração fotovoltaicos a dinâmica de operação dos painéis (circuito elétrico) é mais veloz que as variações do meio ambiente (sol/nuvem), o que facilita a implementação de algoritmos mais robustos. Contudo, em sistemas eólicos, a inércia das turbinas muitas vezes apresenta uma dinâmica mais lenta do que as variações mais rápidas de vento, dificultando a realização do rastreamento.

2.2 Algoritmos de rastreamento de máxima potência básicos

Em diversos trabalhos (BHANDARE; BANDEKAR; MANE, 2013; PATEL et al., 2015) os sistemas de rastreamento de máxima potência principais são classificados em três tipos básicos:

- a) Com medição de velocidade de vento, do inglês *Wind Speed Measurement* (WSM);
- b) Através da realimentação do sinal de potência, do inglês *Power Signal Feedback* (PSF);
- c) Utilizando o algoritmo perturba e observa, do inglês *Perturb and Observe* (P&O).

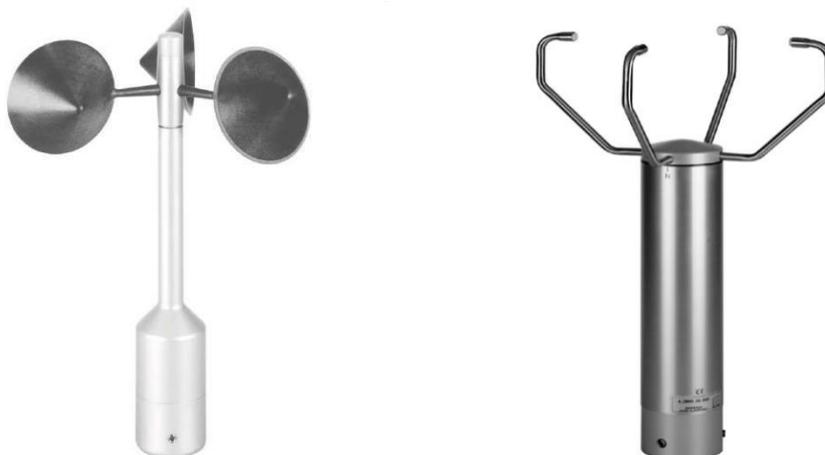
Sistemas WSM são naturalmente os primeiros a serem conjecturados. Neste tipo de sistema são necessários os conhecimentos de duas grandezas: o valor do ótimo de λ e a velocidade instantânea do vento. A partir dessas grandezas é fácil estabelecer o valor de rotação em que a turbina deve operar para extrair a maior quantidade de potência do vento. Para tal, utiliza-se a equação (3).

Contudo duas dificuldades surgem: (i) O custo dos equipamentos necessários para uma medição confiável da velocidade do vento é elevado. Além disso, estes instrumentos que necessitam de manutenção e regulação periódicas. (ii) O valor de λ ótimo adotado, normalmente é aproximado através da equação de proposta por Heier (2006). Contudo, como já mencionado, o valor ótimo de λ varia de acordo com a aerodinâmica da turbina (BHANDARE; BANDEKAR; MANE, 2013).

Como mostrado por Christos (2015) a realização de uma mensuração precisa do vento, influi consideravelmente na eficiência do algoritmo de rastreamento de máxima potência. A Figura 3 apresenta dois modelos de anemômetros da marca *Logotronic*. A esquerda um anemômetro mecânico de primeira classe e a direita um sensor anemométrico ultrassônico.

Aplicações que utilizam WSM são normalmente de elevado porte, como parques eólicos, por exemplo, uma vez que os custos referentes a aquisição e manutenção dos sensores eólicos são amortizados ante os demais gastos.

Figura 3 – Anemômetro mecânico (esquerda) e ultrassônico (direita) da *Logotronic*



Fonte: site da empresa Logotronic

Já em sistemas que utilizem PSF o controle é implementado através do conhecimento prévio da relação entre potência e velocidade do eixo, que é um fator limitante deste método. Este controle consiste em realizar a mensuração da potência gerada pelo sistema eólico-elétrico, e armazenar os dados em tabelas (*Look-Up Tables*). Dessa forma, o controle mantém o valor de potência sempre o mais próximo possível do ponto de máximo registrado na tabela (LALOUNI et al., 2014).

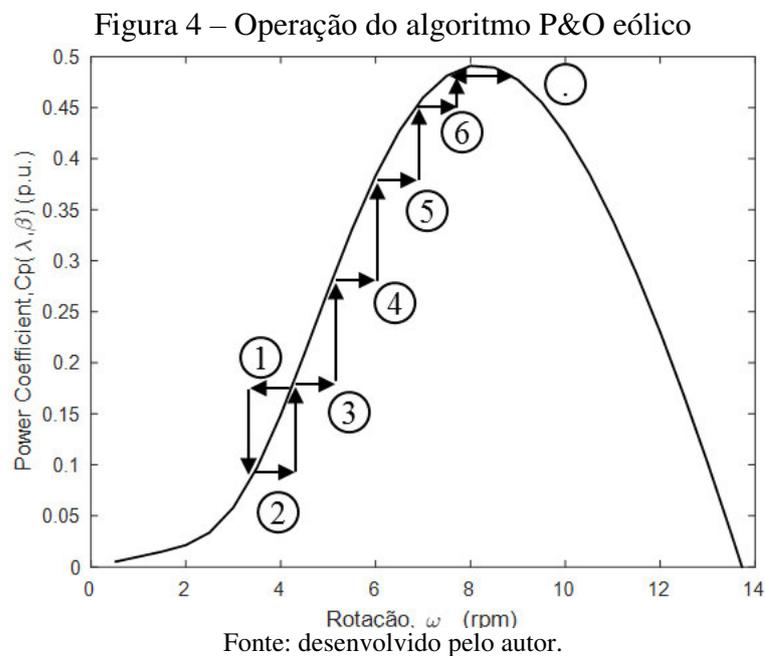
O algoritmo P&O é muito utilizado por não requerer a utilização de instrumentos de medição de vento (BHANDARE; BANDEKAR; MANE, 2013). Embora apresente um custo inferior, a eficiência de busca do ponto de máxima potência desse tipo de algoritmo possui algumas desvantagens.

Um algoritmo P&O básico opera da seguinte maneira:

- a) Iniciado em um estado determinado de operação, aplica-se uma variação (perturbação), alterando este estado;
- b) Após estabilizar-se em um novo estado, avalia-se o novo ponto de operação;
- c) Se a perturbação levou a um ponto de operação melhor, o processo se repetirá com a aplicação da mesma perturbação, caso contrário, uma perturbação de sinal oposto será aplicada.

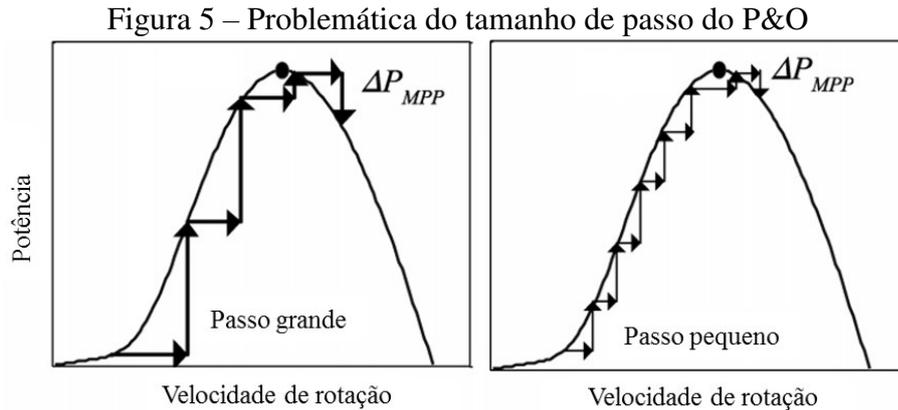
Os parâmetros que definem o ponto de operação ótimo de uma turbina eólica são a velocidade do vento a que a mesma estará submetida e a velocidade de rotação do eixo, resultando em um valor de λ . Sendo o vento a variável não controlada, o algoritmo P&O tem como objetivo então ajustar o ponto de operação através da referência de velocidade de rotação do eixo (ω_{m_ref}) (BHANDARE; BANDEKAR; MANE, 2013).

Em um sistema onde o vento apresente-se com um perfil constante, o P&O operaria similar ao apresentado na Figura 4. Partindo de um ponto inicial, tem-se a perturbação 1. Após algumas iterações de perturbação e observação o sistema aproxima-se do ponto de máxima potência (indicado pelo círculo com um ponto), e após encontra-lo, passa a oscilar em torno do mesmo.



Dentre os problemas mais críticos enfrentados no desenvolvimento de MPPT do tipo P&O, o dimensionamento do tamanho do passo de perturbação mostra-se complexo. A determinação desta variável é feita levando-se em conta a dinâmica mecânica do sistema, além da precisão com que se deseja encontrar o ponto de máxima potência.

Definir o tamanho do passo é complexo a medida que um passo muito grande, alcança rapidamente o ponto de máxima potência, contudo, manter-se-á oscilando com grande amplitude em torno do mesmo. Para reduzir este erro oscilatório em torno do ponto de máxima, é necessária reduzir o tamanho do passo, que por sua vez acarreta em um número maior de iterações para que seja possível alcançar o ponto de máxima.



Fonte: desenvolvido pelo autor.

Diversos trabalhos apresentam algoritmos baseados no P&O, com modificações que simplificam a implementação e/ou melhoram a performance do mesmo (DALALA; ZAHID; LAI, 2013; DALALA et al., 2013).

Contudo, é comum não haverem técnicas que permitam ao sistema de controle aprender ao longo de sua operação, de forma que a dinâmica de resposta do controle mantenha a característica de resposta a variações do vento, ainda que recorrentes. Uma opção de aprendizagem é dada por algoritmos inteligentes.

2.3 Revisão de Inteligência Artificial

Algoritmos inteligentes são aqueles capazes de adquirir e armazenar conhecimento, planejar eventos, solucionar problemas, interpretar linguagens e realizar aprendizagem em geral (LOPES; PINHEIRO; SANTOS, 2014).

Lopes (2014) relata que o termo “inteligência artificial” foi utilizado pela primeira vez na conferência de *Darmouth*, definida genericamente como uma Inteligência construída pelo homem, capaz de dotar inteligência a uma máquina.

A partir dessa ocasião, passam a ser desenvolvidos estudos na área de aprendizagem de máquinas. As formas de aprendizagem de máquinas são classificadas como aprendizado supervisionado, aprendizado por reforço e aprendizagem não supervisionada. Esses estudos resultaram na criação de quatro abordagens de inteligência artificial: Simbólica, Conexionista, Evolucionária e Conjuntos Difusos.

- a) A abordagem simbólica é baseada em sistemas de símbolos, estruturas simbólicas e regras de manipulações dos mesmos, que permitem a solução de problemas, baseado no modelo de raciocínio humano. Teve como

primeiros representantes John McCarthy, Marvin Minsky, Newell e Simon e foi inicialmente utilizada em aplicações específicas.

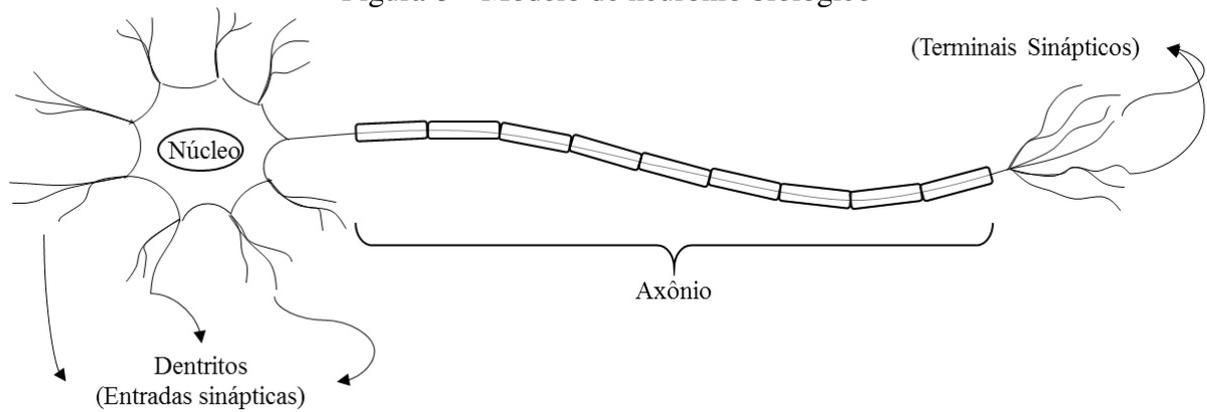
- b) Na conexionista, considera-se a hipótese de causa-efeito, de forma que um determinado número de neurônios biológicos é suficiente para reproduzir a inteligência humana através de exemplos. Essa abordagem iniciou-se com McCulloch, Pitts, Hebb, Rosenblatt e Widrow a partir de um modelo matemático do neurônio biológico. É nessa vertente que se encaixam as redes neurais artificiais.
- c) A linha evolucionária baseia-se na teoria da evolução de Charles Darwin. Nessa hipótese, a solução de um problema é modelada como o genoma de diversos indivíduos que através de recombinações e mutações, permite encontrar a melhor solução. Algoritmos genéticos são o exemplo mais difundido dessa vertente, normalmente aplicada em problemas de otimização.
- d) Nos Conjuntos Difusos (fuzzy set) interpretações linguísticas dos problemas mascaram o desenvolvimento matemático do mesmo. Essa abordagem é aplicada em sistemas onde há grande inconsistência de informações

Embora tenha sofrido certa estagnação de pesquisas após as publicações críticas de Minsky e Papert, o modelo de redes neurais artificiais recuperou sua credibilidade após os estudos realizados por Jonh Hopfield (LOPES; PINHEIRO; SANTOS, 2014).

2.3.1 Redes Neurais Artificiais

As redes neurais artificiais buscam operar de forma a mimetizar o comportamento visto em neurônios biológicos. Para isso, parte-se do princípio de funcionamento do neurônio biológico. A estrutura dessa célula é composta por três partes: Axônio, Núcleo e Detritos e é responsável pelas transmissão e processamento de impulsos nervosos (GONDIM, 2013). A Figura 6 apresenta um modelo de neurônio biológico.

Figura 6 – Modelo de neurônio biológico

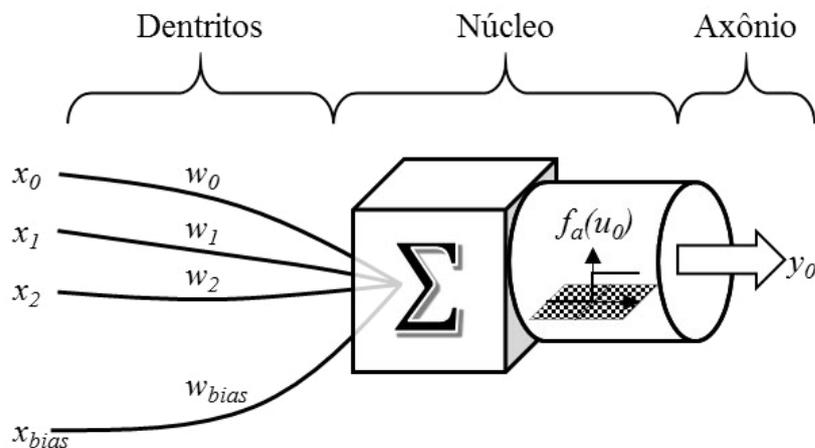


Fonte: desenvolvido pelo autor.

A transmissão de impulsos nervosos ou potenciais de ação é feita partindo-se dos dendritos em direção ao axônio. Essa transmissão é feita através impulsos elétricos, gerados a partir da alteração de íons de sódio (Na^{2+}) e potássio (K^+). Impulsos neurais aplicados nos dendritos, resultam em diferentes níveis de acréscimo da diferença de potencial no núcleo. Um impulso só será transmitido ao axônio caso a diferença de potencial no núcleo supere um valor limiar, de forma que o neurônio obedeça uma lei de tudo ou nada (GONDIM, 2013).

Os neurônios artificiais por sua vez, são constituídos por entradas e *bias* ponderadas por pesos, que equivalem à função dos dendritos; somatório das entradas ponderadas (u), e aplicação na função de ativação ($f_a(u)$), correspondente ao papel do núcleo e do potencial de ativação; já a saída corresponde ao axônio de um neurônio biológico. A Figura 7 apresenta o diagrama de um neurônio artificial.

Figura 7 – Modelo de neurônio artificial



Fonte: desenvolvido pelo autor.

O primeiro modelo neuronal proposto foi o de McCulloch-Pitts, cuja característica era uma função de ativação do tipo limiar. Baseado nesse modelo, Rosenblatt em 1958 desenvolveu o modelo do neurônio Perceptron, passa adotar funções de ativação variadas, como a sigmoide logística (HAYKIN, 2009).

O aprendizado de um neurônio biológico se dá por meio de estímulos vivenciados pelo indivíduo. Um estudo que exemplifica esta forma de aprendizado cognitivo foi o apresentado por Pavlov (MOREIRA; MEDEIROS, 2007). Neste experimento, um cão é submetido ao estímulo de um sinal sonoro que antevem uma refeição, após certo tempo de condicionamento, é observado o aumento da saliva do cão quando o sinal sonoro é acionado, mesmo que não haja refeição. O estímulo sonoro, passou a ser associado à comida, devido à repetição da vivenciada de estímulos ocorridos concomitantemente.

Um neurônio artificial simples, por sua vez, possui a capacidade de aprender através dos ajustes dos pesos que ponderam suas entradas. Dessa forma, é possível, através de treinamentos, ajustar os pesos para que determinado valor de entrada, resulte em uma saída desejada. De acordo com (HAYKIN, 1999), o processo de aprendizagem requer necessariamente a seguinte sequência de eventos:

- a) A rede neural é submetida a estímulos provenientes do ambiente.
- b) A rede neural ajusta seus parâmetros de acordo com os estímulos recebidos.
- c) A rede responde de uma nova forma quando submetida novamente a tais estímulos, devido as mudanças ocorridas.

A regra de aprendizado mais famosa e antiga é o aprendizado Hebbiano. Essa regra declara que uma relação sináptica que seja repetida várias vezes, reforça as conexões entre as sinapses que a ativam (HAYKIN, 1999). Matematicamente a lei de Hebb é descrita como em (4), onde η representa um coeficiente de aprendizagem entre 0 e 1, t é uma variável de indicativo temporal e j um index referente às entradas do neurônio.

$$\Delta w_j = \eta \times Y(t) \times X_j(t) \quad (4)$$

É possível perceber que através da regra de aprendizagem de Hebb, caso um estímulo seja repetido várias vezes, ele irá eliminar o aprendizado obtido de outros estímulos. A partir dessa limitação, Widrow e Hoff desenvolveram em 1960 a regra denominada aprendizagem por correção de erro, ou simplesmente regra delta (HAYKIN, 1999).

Considerando um valor desejado de saída do Perceptron ($D(n)$), relacionado com determinado valor de entrada ($X(n)$). Aplicando o $X(n)$ nas entradas de um neurônio, obtém-se um valor de saída ($Y(n)$), a partir daí é possível calcular o erro como em (5). O index n refere-se a uma amostra do conjunto de treinamento.

$$e(n) = d(n) - Y(n) \quad (5)$$

Esse valor é então utilizado para o cálculo do valor instantâneo da energia de erro ($\xi(n)$) como em (6). Esse valor é utilizado como indicador de qualidade do aprendizado. Quanto menor o valor acumulado de energia do erro, melhor treinado está o neurônio.

$$\xi(n) = \frac{1}{2} e^2(n) \quad (6)$$

Dessa forma, o valor de ajuste dos pesos do neurônio é feito de acordo como em (7).

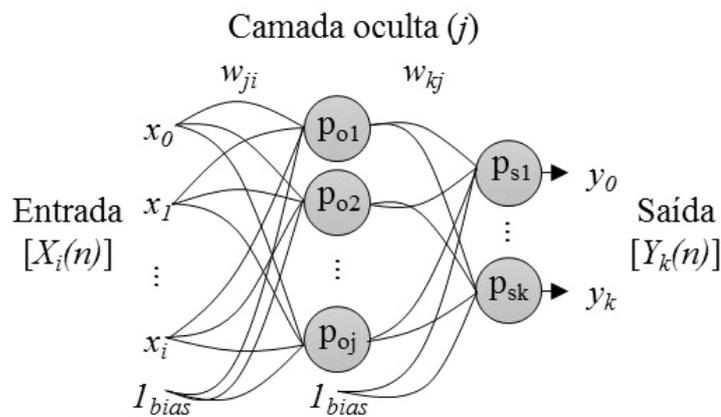
$$\Delta w_i(n) = \eta \times e(n) \times X_i(n) \quad (7)$$

O ajuste dos pesos é então feito como mostra a equação (8). Onde o index t é um indicador cronológico.

$$w_j(t+1) = w_j(t) + \Delta w_j(t) \quad (8)$$

Um neurônio do tipo Perceptron possui a capacidade de realizar a classificação de amostras em classes, contanto que as classes sejam linearmente separáveis. Além disso, um único neurônio só tem capacidade para realizar a classificação em dois grupos, sendo necessária a utilização de mais neurônios, caso haja um número maior de grupos a serem classificados (HAYKIN, 2009).

Como forma de superar a incapacidade de realizar as classificações não linearmente separáveis uma arquitetura de rede neural foi desenvolvida através da associação de Perceptron em camadas, dando-se o nome de Perceptron de Multicamadas (*Multi-Layer Perceptron - MLP*). A Figura 8 apresenta um modelo de uma rede neural do tipo MLP.

Figura 8 – Modelo de arquitetura *Multi-Layer Perceptron* de Rede Neural

O treinamento de redes neurais do tipo MLP mostra-se mais complexo, uma vez que para as camadas ocultas não há valor esperado, não havendo erro a ser calculado. Dessa forma, utiliza-se o algoritmo denominado *Back-Propagation*, para realizar o ajuste dos pesos de redes MLP.

O algoritmo *Back-Propagation* (BP) consiste na transmissão de uma parcela do erro obtido na camada de saída até as camadas intermediárias da rede MLP. Considerando o modelo apresentado na Figura 8, no qual existem i entradas, j Perceptrons na camada oculta e k saídas, para cada conjunto (n) de entrada ($X(n)$) aplicado na rede, um valor de saída ($Y(n)$) é obtida e um valor de saída desejada ($D(n)$) é definido pelo conjunto de treinamento. A partir desses parâmetros, é possível definir para cada elemento k da saída, um valor de erro definido como em (9). Onde n indica uma amostra do conjunto de treinamento.

$$e_k = d_k(n) - y_k(n) \quad (9)$$

No casos em que k é superior a 1, tem-se que o valor instantâneo da energia do erro é definido como em (10). Durante o treinamento, pode-se considerar como critério de parada o valor médio da energia de erro obtida para cada amostra apresentada à rede.

$$\xi(n) = \frac{1}{2} \sum_k e_k^2(n) \quad (10)$$

De acordo com o comportamento individual do neurônio, tem-se que a saída da rede neural comporta-se de acordo com (11), na qual $y_j(n)$ refere-se a saída do neurônio da camada oculta e $u_k(n)$ ao somatório sináptico antes da função de ativação.

$$\begin{aligned} y_k(n) &= f_a[u_k(n)] \\ u_k(n) &= \sum_j w_{kj}(n) \cdot y_j(n) \end{aligned} \quad (11)$$

Como mostram Haykin (2005), o ajuste dos pesos que conectam a camada oculta com a camada de saída (w_{kj}) são proporcionais ao valor do gradiente da energia do erro em relação aos pesos, que pode ser desenvolvido através da regra da cadeia, como mostrado em (12).

$$\frac{\partial \xi(n)}{\partial w_{kj}(n)} = \frac{\partial \xi(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial u_k(n)} \frac{\partial u_k(n)}{\partial w_{kj}(n)} \quad (12)$$

Na observância dos cálculos das derivadas parciais apresentadas em (13), (14), (15) e (16) tem-se que o gradiente apresentado em (12), que pode ser simplificado como em (17).

$$\frac{\partial \xi(n)}{\partial e_k(n)} = \frac{\partial \frac{1}{2} \sum_k e_k^2(n)}{\partial e_k(n)} = e_k(n) \quad (13)$$

$$\frac{\partial e_k(n)}{\partial y_k(n)} = \frac{\partial [d_k(n) - y_k(n)]}{\partial y_k(n)} = -1 \quad (14)$$

$$\frac{\partial y_k(n)}{\partial u_k(n)} = \frac{\partial f_a[u_k(n)]}{\partial u_k(n)} = f'_a[u_k(n)] \quad (15)$$

$$\frac{\partial u_k(n)}{\partial w_{kj}(n)} = \frac{\partial \sum_k w_{kj}(n) \cdot y_i(n)}{\partial w_{kj}(n)} = y_i(n) \quad (16)$$

$$\frac{\partial \xi(n)}{\partial w_{kj}(n)} = -e_k(n) \cdot f'_a[u_k(n)] \cdot y_i(n) \quad (17)$$

Uma vez que o gradiente maximiza o erro, uma variação dos pesos é feita em sentido contrário. Em (18) é apresentada a correção a ser aplicada nos pesos em função do gradiente local ($\delta_k(n)$) (HAYKIN, 1999).

$$\Delta w_{kj}(n) = -\frac{\partial \xi(n)}{\partial w_{kj}(n)} = \delta_k(n) \cdot y_j(n) \quad (18)$$

$$\delta_k(n) = e_k(n) \cdot f'_a[u_k(n)]$$

Seguindo o mesmo raciocínio é possível calcular o valor de ajuste dos pesos que ligam a camada de entrada à camada oculta. A regra da cadeia aplicada ao gradiente da energia do erro ($\xi_j(n)$) pelos pesos ($w_{ji}(n)$) é apresentada em (19).

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = \frac{\partial \xi(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} \frac{\partial u_j(n)}{\partial w_{ji}(n)} \quad (19)$$

A derivada parcial da energia do erro pela saída dos neurônios da camada oculta é desenvolvida em (20), (21) e (22).

$$\frac{\partial \xi(n)}{\partial y_j(n)} = \frac{\partial \frac{1}{2} \sum_k e_k^2(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial u_k(n)} \frac{\partial u_k(n)}{\partial y_j(n)} \quad (20)$$

$$\frac{\partial e_k(n)}{\partial u_k(n)} = \frac{\partial [d_k(n) - y_k(n)]}{\partial u_k(n)} = \frac{-\partial y_k(n)}{\partial u_k(n)} = -f'_a[u_k(n)] \quad (21)$$

$$\frac{\partial u_k(n)}{\partial y_j(n)} = \frac{\partial \left[\sum_j w_{kj}(n) \cdot y_j(n) \right]}{\partial y_j(n)} = w_{kj}(n) \quad (22)$$

Substituindo (21) e (22) em (20), é possível obter o valor de ajuste dos pesos da camada oculta apresentado em (23) (HAYKIN, 2009).

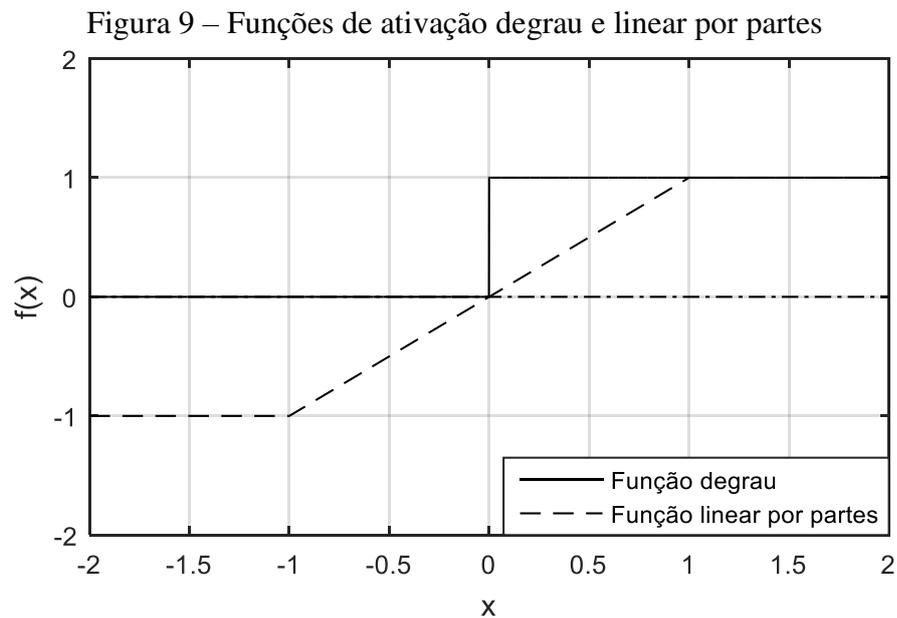
$$\Delta w_{ji}(n) = -\frac{\partial \xi(n)}{\partial y_j(n)} = -\sum_k \delta_j(n) \cdot w_{kj}(n) \quad (23)$$

$$\delta_j(n) = -f'_a[u_j(n)] \cdot \sum_k \delta_k(n) \cdot w_{kj}(n)$$

Dessa forma, as equações (18) e (23) apresentam os valores de ajuste dos pesos da camada de saída e oculta, respectivamente. Observa-se que o gradiente local da camada oculta é função do gradiente local da camada de saída, explicitando a retro propagação do algoritmo.

A função de ativação pode ser representada por diversos tipos de funções, contudo, algumas funções apresentam características favoráveis à sua utilização. As principais funções utilizadas são a linear, logística e tangente hiperbólica.

A função limiar ou degrau e a linear por partes são as mais simples delas, são mais comuns em redes de única camada, uma vez que a capacidade de aprendizagem com funções lineares apresenta baixo potencial (LOPES; PINHEIRO; SANTOS, 2014). A Figura 9 apresenta o perfil dessas funções.

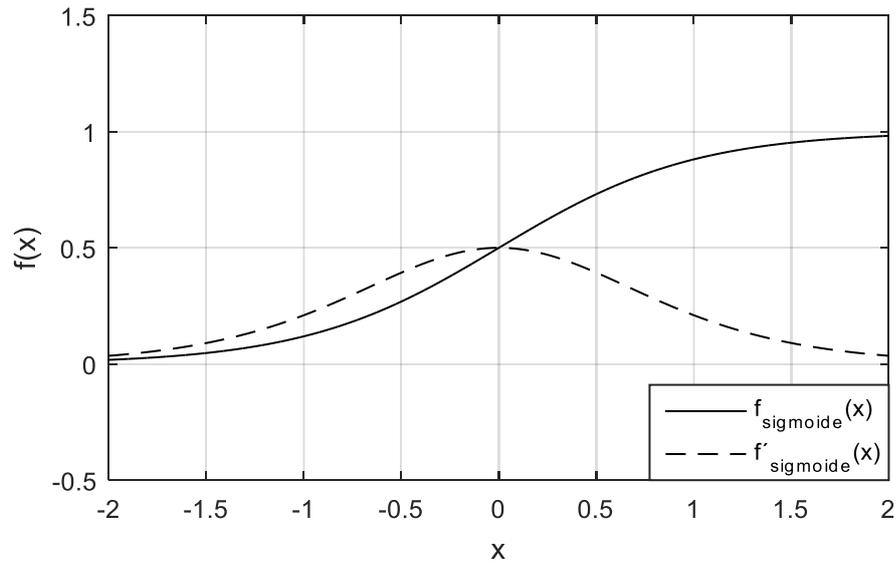


A função logística ou sigmoide, apresenta um perfil semilinear, como apresenta a Figura 10. Uma vantagem dessa função é dada pela sua derivada, cujo valor é uma função dela mesma. As equações (24) e (25) apresentam a função sigmoide e sua derivada, respectivamente.

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-2\beta x}} \quad (24)$$

$$f'_{sigmoide}(x) = 2 \cdot \beta \cdot f_{sigmoid}(x) \cdot [1 - f_{sigmoid}(x)] \quad (25)$$

Figura 10 – Função de ativação sigmoide logística e sua derivada



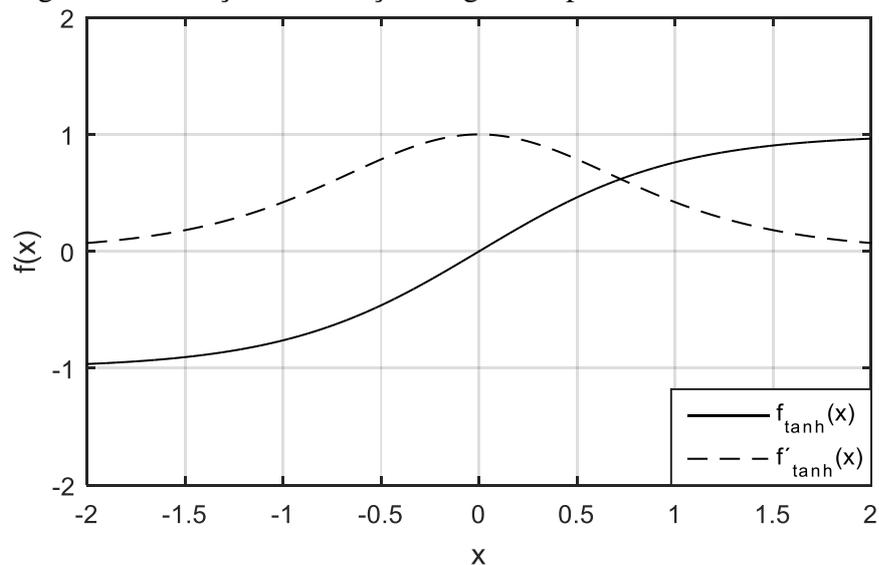
Fonte: desenvolvido pelo autor

Além da sigmoide, a função tangente hiperbólica também apresenta essa mesma característica de derivada. A Figura 11 apresenta o perfil dessa função e as equações (26) e (27) apresentam o formato da função e sua derivada.

$$f_{\text{tanh}}(x) = \tanh(\beta x) \quad (26)$$

$$f'_{\text{tanh}}(x) = \beta [1 - f_{\text{tanh}}^2(x)] \quad (27)$$

Figura 11 – Função de ativação tangente hiperbólica e sua derivada



Fonte: desenvolvido pelo autor.

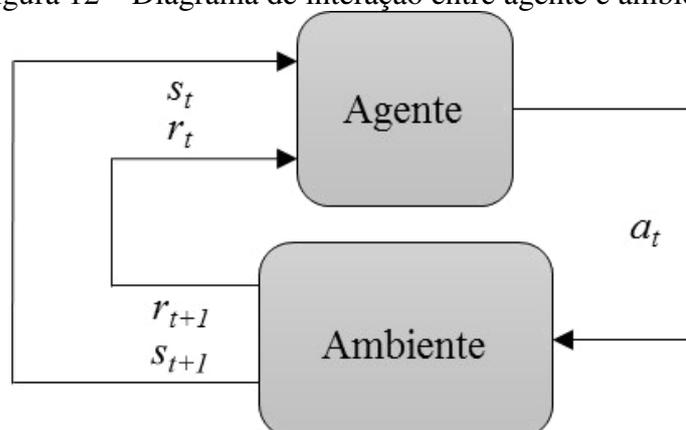
Embora as redes neurais tenham capacidade de aprender através de exemplos, existem situações onde não é possível apresentar exemplos para apresentar a rede neural. Existem certos algoritmos capazes de realizarem o aprendizado sem serem apresentados a exemplos, possuindo apenas uma informação mínima que qualifica quão bom é realizar certa ação, a tais algoritmos possuem uma aprendizagem dita por reforço.

2.3.2 *Aprendizagem por Reforço*

A forma mais básica de aprendizagem do ser humano é através da interação com o meio ambiente. Através das estruturas sensoriais e motoras, aplicamos estímulos que resultam em ações, os resultados dessas ações são então a fonte de informação qualitativa da ação tomada. Quando uma criança aprende a caminhar, não há um ensinamento de quais músculos devem ser contraídos ou de que forma é possível manter-se em equilíbrio, pois esse tipo de aprendizagem vem através dos resultados obtidos em tentativas anteriores. É possível, partindo desse conceito, desenvolver aplicações computacionais autônomas que utilizem a aprendizagem por reforço (*Reinforcement Learning – RL*) (SUTTON; BARTO, 2012).

Em abordagens computacionais de aprendizagem por reforço, o sistema inteligente, também denominado agente, não possui uma saída correta determinada, ao invés disso, ele realiza ações (a_t) através de um sistema atuador, observa o ambiente (s_t) e suas alterações por meio de sensores e por fim define um valor de recompensa (r_t) para a ação tomada com base nas observações realizadas (SUTTON; BARTO, 2012). A Figura 12 apresenta o diagrama da relação entre o ambiente e o do agente.

Figura 12 – Diagrama de interação entre agente e ambiente



Fonte: desenvolvido pelo autor

A tarefa do agente, portanto, é selecionar a ação que apresente um maior acumulado de recompensas ao final do processo. A probabilidade de escolha de ações é denominada política ($\pi_t(s_t, a_t)$), sendo essa uma função do estado do ambiente e das ações possíveis daquele estado (SUTTON; BARTO, 2012).

Ainda de acordo com Sutton e Barto (2012), em algoritmos de aprendizagem por reforço estimam-se valores que definem quão bom é determinado estado ou ação. Essas avaliações são organizadas em forma de uma função denominada Função de Valor ($V_\pi(s)$). Essas avaliações levam em conta não só a recompensa obtida em um estado, mas todas as possíveis recompensas futuras, bem como o tipo de política que é utilizada. Em um sistema onde a função de valor represente corretamente a avaliação de cada estado ou ação, a política ótima fica definida através da decisão pelos estados ou ações com maiores valores. A equação (28) apresenta a forma de cálculo da função de valor de estado, nela γ indica uma amortização de valores de recompensas futuras, que possui valor entre 0 e 1.

$$V_\pi(s) = \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t \right] \quad (28)$$

Alguns algoritmos de aprendizagem por reforço não consideram apenas a função de valor de estado, mas sim a função de valor de ação ($Q_\pi(s, a)$) que se refere a avaliação de cada possível ação para cada estado. A equação apresenta a forma de cálculo dessa variável.

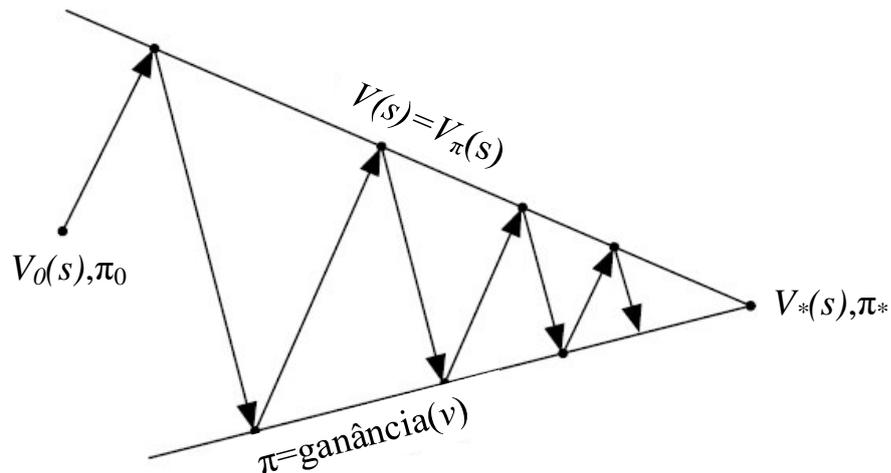
$$Q_\pi(s, a) = \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t, a_t \right] \quad (29)$$

Como é possível perceber, as funções de valores dependem das recompensas obtidas quando se segue determinada política. O valor das recompensas obtidas muitas vezes é desconhecido a priori, sendo necessário que o próprio agente realize iterações com o ambiente, que lhe permitam mensurar as recompensas obtidas. Dessa forma, existem alguns modelos de algoritmos que propõem formas de iteração e atualização das funções de valores.

Programação dinâmica (*Dynamic Programming* – DP) é um algoritmo básico, que embora possuía elevado custo computacional, é essencial para a compreensão dos métodos de aprendizagem por reforço. Em DP uma política de escolha probabilística inicial é selecionada, através dela, inicia-se as iterações que ajustam os valores de estado/ação, em seguida, ajusta-se

a política de forma a incrementar sua ganância (*greed*), isto é, aumenta-se a probabilidade escolha de uma ação com função de valor maior. Esse processo se dá de forma intercalada, até que a função de valor se torne ótima ($V^*(s)$) definindo a política ótima (π^*) ser seguida (SUTTON; BARTO, 2012). A Figura 13 demonstra graficamente o processo de intercalar os ajustes de política e de função de valor.

Figura 13 – Gráfico de aprendizagem por Programação Dinâmica



Fonte: adaptado de (SUTTON; BARTO, 2012)

No algoritmo DP o ajuste das funções de valor de estado são calculados de acordo com a equação (30).

$$V_{k+1}(s) = [r_{t+1} + \gamma V_k(s_{t+1}) | s_t, a_t] \quad (30)$$

Após um certo número de iterações com o ambiente, a avaliação descrita pela função de valor permite encontrar a política ótima, definida pela escolha determinística do estado futuro de melhor avaliação. Embora seja melhor que algoritmos de busca direta, esse método de aprendizagem por reforço, requer um modelo do espaço de busca.

Outro algoritmo classificado como aprendizagem por reforço é o método de Monte Carlo (MC). Diferente do algoritmo DP, este método não requer um modelo do espaço de busca. Os ajustes realizados através de MC ocorrem apenas ao final dos episódios, não a cada iteração entre o agente e o ambiente (SUTTON; BARTO, 2012).

A equação (31) apresenta as formas de ajuste das funções de valor de estado e de estado/ação, a partir do método de MC. O coeficiente α indica uma taxa de aprendizado,

referindo-se a proporção de ajuste que será tomado a cada episódio, R_t indica o valor de recompensas médias, calculado ao término do episódio.

$$\begin{aligned} V_{k+1}(s) &= V_k(s_t) + \alpha(R_t - V_k(s_t)) \\ Q_{k+1}(s, a) &= Q_k(s, a) + (R_t - Q_{k+1}(s, a)) \end{aligned} \quad (31)$$

Este algoritmo consiste em ajustar as avaliações de estados e estados/ações a partir do valor médio de recompensas obtidas durante um episódio. Dessa forma, episódios com muitos passos de iteração e diversas recompensas pequenas, será pouco reforçado, se comparado a um episódio com poucas iterações e apenas uma recompensa significativa, uma vez que os valores médios no segundo caso podem superar o primeiro caso.

Já o método das diferenças temporais (TD) apresenta a unificação de conceitos de DP e MC, unificados em um único algoritmo, que permite realizar ajustes de avaliações a cada iteração entre o agente e o ambiente, sem a necessidade de conhecer completamente o ambiente (SUTTON; BARTO, 2012).

Em algoritmos baseados em TD, diferentemente de MC, o ajuste é realizado levando-se em consideração a recompensa imediata, anulando assim, a necessidade de finalização de um episódio para ajustar as avaliações. A equação (32) apresenta a forma de ajuste da função de valor através do algoritmo de TD. O valor r_k neste caso, indica a recompensa obtida na última iteração e γ é um coeficiente que pondera a importância das avaliações futuras, de forma que haja uma melhoria da avaliação, mesmo que não haja uma boa recompensa instantânea. O termo apresentado dentro dos colchetes é denominado diferença temporal, responsável pelo nome do algoritmo (SUTTON; BARTO, 2012).

$$V_{k+1}(s) = V_k(s) + \alpha[r_k + \gamma V_k(s+1) - V_k(s)] \quad (32)$$

Por fim, tem-se o algoritmo *Q-learning* que é basicamente uma modificação aplicada ao algoritmo TD, utilizando a avaliação das ações, ao invés das avaliações dos estados. A equação (33) apresenta a forma de ajuste da função de valor de ação, realizada no algoritmo *Q-learning*.

$$Q_{k+1}(s, a) = Q_k(s, a) + \alpha \left[r_k + \gamma \cdot \max_a (Q_k(s+1, a) - Q_k(s, a)) \right] \quad (33)$$

Embora tenha uma excelente e comprovada eficácia em solução de problemas, sistemas de aprendizagem por reforço podem em certos casos necessitam de grandes espaços de memória, o que se mostra a grande desvantagem de aplicações com essa estrutura.

2.4 Comentários parciais

Com base nos conceitos apresentados de algoritmos de rastreamento de máxima potência, busca-se então desenvolver um sistema de rastreamento que apresente a capacidade de aprender a melhor forma de operação, sem a necessidade de sensoriamento de velocidade do vento.

Dessa forma, busca-se desenvolver um algoritmo baseado em aprendizagem por reforço que possua a capacidade de aprender a operar um sistema eólico de forma a mantê-lo atuando sempre próximo ao seu ponto de máxima potência. Devido ao espaço de operação ser contínuo, há alternativas de discretizar o espaço de estados ou utilizar um aproximador de função. Opta-se por utilizar um sistema de redes neurais para realizar tal aproximação.

3 PLATAFORMA DE VALIDAÇÃO EÓLICA

Com o objetivo de realizar a validação do algoritmo de MPPT neural, é necessário a realização de ensaios experimentais. Afim de validar experimentalmente algoritmos de MPPT algumas técnicas podem ser utilizadas, como teste em campo, túnel de vento e bancadas emuladoras.

Ao realizar a validação do algoritmo através de testes em campo, os demais elementos do sistema já devem ter sido validados e precisam apresentar elevado grau de confiabilidade, afim de evitar incidentes. Além disso, há a necessidade de uma medição de vento com precisão elevada, para que seja possível observar a eficácia do sistema MPPT proposto.

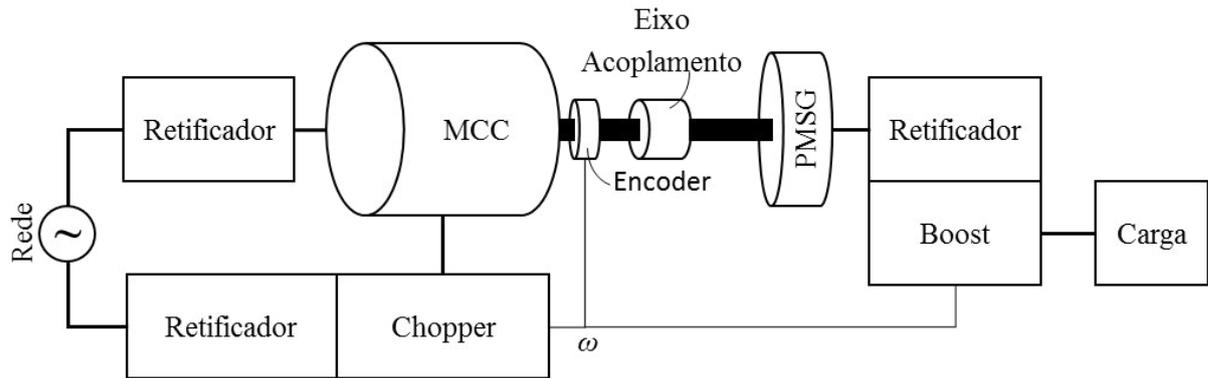
Experimentos que utilizam tuneis de vento são extremamente importantes do ponto de vista aerodinâmico do gerador, uma vez que o comportamento será o mesmo visto em aplicações de campo, mas havendo uma segurança maior com relação ao experimento. Contudo, tuneis de vento, mesmo os de pequeno porte, representam custos razoáveis (KOLTER; JACKOWSKI; TEDRAKE, 2012).

Quando o experimento realizado tem como objetivo o teste de um algoritmo de MPPT, é comum a utilização de bancadas emuladoras de turbinas eólicas baseadas na associação motor-gerador (BARBOSA, 2015a; HARDY; JEWELL, 2011; KOUADRIA et al., 2013; MAHDY; EL-HAKIM; HANAFY, 2011; SILVA, 2012; TAVEIROS; BARROS; BEZERRA COSTA, 2013).

De forma geral, utiliza-se um motor associado via caixa de engrenagens a um gerador eólico. Nessas bancadas um conversor eletrônico compatível com o tipo de motor adotado é utilizado para controla-lo, já o gerador é conectado a um outro conversor que terá dentre outras finalidades, a de controlar a carga, a fim de permitir que o gerador opere no ponto de máxima potência.

No trabalho realizado, o sistema experimental baseia-se em uma bancada emuladora composta de um motor de corrente contínua (MCC) com o eixo diretamente conectado ao eixo de um gerador síncrono de ímã permanente (PMSG), comumente utilizado em aplicações de geração de energia eólica. Ambas as máquinas utilizam conversores chaveados em seu acionamento e controle de carga, respectivamente, como apresentado na Figura 14. Todos os parâmetros do projeto são descritos e/ou calculados através de um código em linguagem Matlab apresentado no Apêndice H.

Figura 14 – Diagrama do sistema experimental



Como é possível observar na Figura 14 os controles desenvolvidos em ambos os conversores requerem leitura do valor de velocidade de rotação do eixo mecânico das máquinas. Dessa forma, utiliza-se um sensor de rotação cuja saída é conectada a ambos os conversores.

3.1 Sensoriamento da velocidade de rotação

Neste projeto a velocidade de rotação é obtida através de um *encoder* incremental acoplado ao eixo comum a ambas as máquinas. O modelo de *encoder* utilizado é do tipo RI-76TD da *Hengstler*.

Figura 15 – Modelo de encoder Hengstler.



Fonte: catalogo da *Hengstler*.

O modelo de encoder utilizado apresenta alguns protocolos de comunicação, além do sistema básico de medição, que consiste em 3 sinais: um index, que emite um pulso a cada volta e os pares A e B, que emitem um total de 10.000 pulsos a cada revolução dada pela

máquina. Os pulsos emitidos por A e B são defasados de 90° um do outro, o que permite observar o sentido de rotação da máquina. A Figura 15 apresenta a foto de um encoder do modelo utilizado.

Alguns microcontroladores das famílias dsPIC30F e dsPIC33F apresentam periféricos que permitem a utilização simplificada deste tipo de sensor. Trata-se do *Quadrature Encoder Interface* (QEI), que permite observar o número de voltas e pulsos enviados pelo encoder, sendo inclusive possível a observação do sentido de rotação do eixo. Em microcontroladores que não apresentem este periférico, é possível realizar a medição de pulsos através de uma interrupção externa.

A leitura de um encoder pode ser realizada de duas formas básicas: medindo um tempo fixo entre interrupções, na qual é realizada a contagem de pulsos durante um tempo predefinido, ou realizando a medida de um número fixo de pulsos, sendo realizado em paralelo a medição do tempo necessário para que esse número de pulsos ocorra.

Opta-se neste trabalho pela utilização de medições em tempo fixo, sendo possível assim, sincronizar a leitura com a atuação da malha de controle. Ambos estão sincronizados com um temporizador cuja interrupção ocorre a cada 10ms, resultando numa frequência de sensoriamento de velocidade de rotação de 100 Hz.

O projeto dos conversores chaveados utilizado no acionamento e os respectivos controle são descritos nas sessões seguintes.

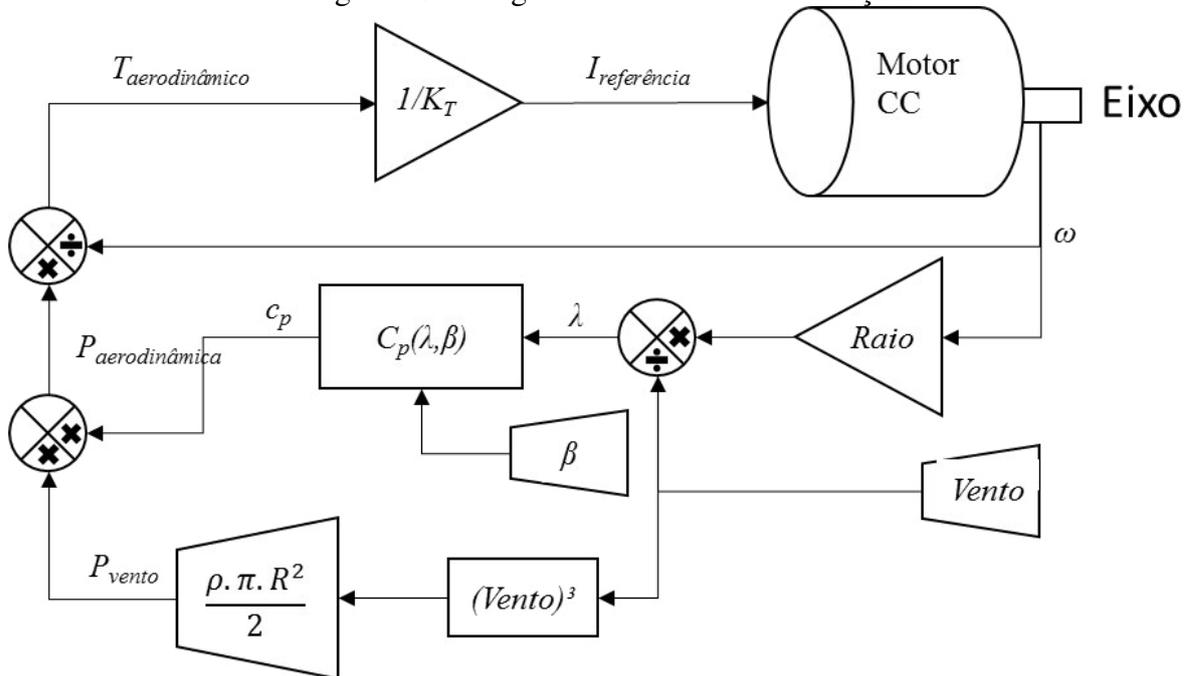
3.2 Sistema de Emulação

Um sistema de emulação de turbina deve se comportar de forma que, do ponto de vista do eixo do gerador, o torque observado seja equivalente ao imposto pelas pás de uma turbina eólica real sob certo valor de velocidade de vento.

Alguns trabalhos já apresentam métodos de utilização de motores de corrente contínua para realizar a emulação de turbinas eólicas (HARDY; JEWELL, 2011; KOUADRIA et al., 2013; MAHDY; EL-HAKIM; HANAFY, 2011; TAVEIROS; BARROS; BEZERRA COSTA, 2013). O emulador desenvolvido neste trabalho tem como base os estudos preliminares e simulações apresentados por Barbosa (2015).

Dessa forma, o sistema de acionamento do motor, deve controlar o torque imposto pelo mesmo, para que este siga o perfil estabelecido pela curva aerodinâmica das pás. A fim de implementar o sistema de emulação de turbina eólica, utiliza-se um motor de corrente contínua, cujo controle é realizado através de um *chopper* como mostra o diagrama da Figura 16.

Figura 16 – Diagrama do sistema de emulação.



Fonte: desenvolvido pelo autor.

Uma vez que a corrente se relaciona com o torque através de uma constante K_T , o *chopper* irá controlar o torque através de uma malha de corrente. Os demais elementos apresentados no diagrama, são referentes ao comportamento, discutido anteriormente, de uma turbina eólica. O valor da velocidade de vento armazenado em memória possibilita a utilização de perfis diferentes, que podem ser gravados diretamente na memória do microcontrolador do emulador.

3.2.1 Parâmetros da turbina emulada/Motor de Corrente Contínua

O motor de corrente contínua é utilizado na configuração de excitação de campo independente, o que permite um melhor controle da corrente de armadura, haja vista que nessa configuração, a variação da tensão de armadura apresenta baixa variação relacionada a variação de corrente (FITZGERALD, A. E.; KINGSLEY JR., C.; UMANS, 2006). Para realizar tal controle, é necessário apenas uma malha de controle de corrente, uma vez que a relação entre a corrente e o torque mecânico é dado pela constante K_T .

O motor CC utilizado em bancada é modelo 2-MZCC-4CB da ENIKA. Os parâmetros de placa no motor são: potência de 2KW, tensão de alimentação CC 220V, rotação nominal de 1800rpm, corrente nominal de 9,1 A, fabricado em 1981. Os demais parâmetros do

motor CC foram ensaiados de acordo com o Apêndice A: Ensaio do Motor CC e são apresentados na Tabela 2.

Tabela 2 – Dados de ensaio do motor CC

Parâmetros do motor CC	
Resistência de Armadura	1,1 Ohms
Indutância de Armadura	6,0 mH
Resistência de Campo	337 Ohms
Indutância de Campo	13,2 H
Momento de inércia (J_{mcc})	0,04160 Kg.m ²
Coefficiente de atrito viscoso (B_{mcc})	0,04160 N.m / (rad/s)
Coefficiente de torque (K_{Tmcc})	1,0403 N.m / A
Coefficiente de tensão ($K\omega_{mcc}$)	1,0403 V / (rad/s)

Fonte: desenvolvido pelo autor.

Os dados de turbina utilizados são fictícios, contudo, buscam assemelhar-se com os valores reais de uma turbina eólica de pequeno porte. Tais parâmetros são apresentados na Tabela 3.

Tabela 3 – Dados da turbina eólica fictícia emulada

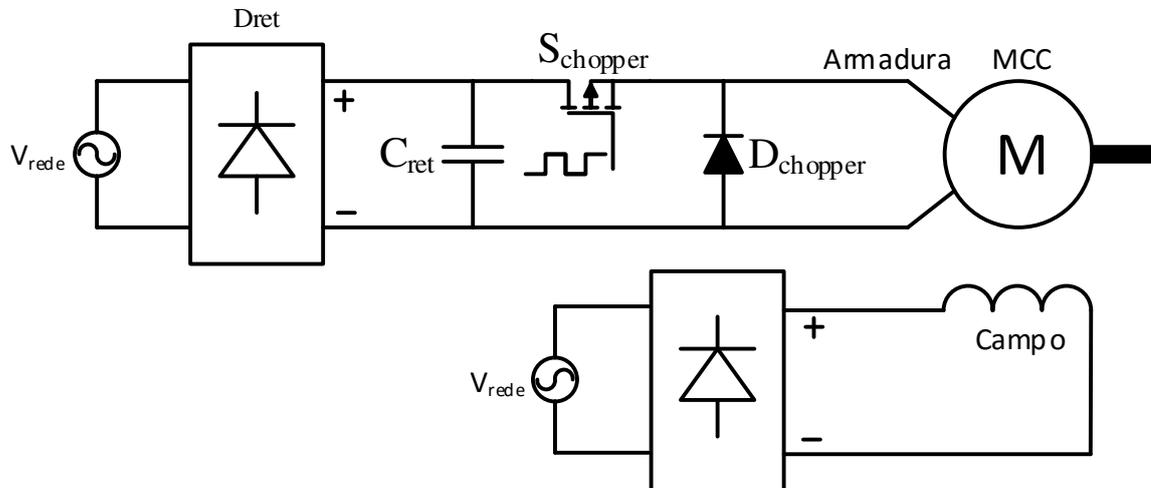
Parâmetros da turbina simulada	
Ângulo de ataque (β)	0°
Relação de velocidade (λ) ótima	8
Valor máximo do coeficiente de potência (C_p)	0,35
Comprimento das pás (Raio)	0,69 m
Densidade do ar (ρ)	1,2928 kg/m ³ .
Velocidade de vento máxima	10 m/s
Velocidade de vento mínima	4 m/s
Velocidade de rotação máxima	1108 rpm
Velocidade de rotação mínima	439 rpm

Fonte: desenvolvido pelo autor.

3.2.2 Projeto do sistema de acionamento do emulador

Para realizar o acionamento e controle do motor CC é necessário um dispositivo que tenha a capacidade de regular o nível de tensão de alimentação, permitindo o controle da corrente injetada no motor. Para tal, optou-se na utilização de retificador não controlado, associado a um conversor chaveado tipo Chopper, semelhante a um conversor Buck, mas no qual não se faz necessária a utilização de indutor para armazenagem de energia, uma vez que a carga em questão, o motor CC, já apresenta um elevado valor de indutância associada a seus enrolamentos, o que permite o armazenamento de energia. O esquemático do sistema de emulação é apresentado na Figura 17.

Figura 17 – Esquemático do circuito Chopper de acionamento do motor CC



Fonte: desenvolvido pelo autor.

No projeto do sistema retificador-*chopper*, alguns valores devem ser estabelecidos a prioritariamente. Tais valores são apresentados na Tabela 4.

Tabela 4 – Dados nominais do conversor Chopper para acionamento do motor CC

Parâmetros nominais do sistema retificador / chopper	
Pico da tensão de entrada (Rede) – (Vrede_pico)	311 V
Frequência da tensão de entrada (Rede) – (frede)	60 Hz
Tensão de saída – (Vo)	220 Vcc
Frequência de chaveamento – (fchave)	20 KHz
Corrente de saída máxima – (Io_max)	9 A
Potência de entrada – (Pi)	2 KW
Tensão retificada mínima – (Vcc_min)	249 V
Ciclo de trabalho nominal	0,707

Fonte: desenvolvido pelo autor.

A partir desses parâmetros, calcula-se, de acordo com (BARBI, 2006), os valores dos componentes passivos e os esforços sobre os componentes ativos.

No sistema retificador é necessário definir o valor da capacitância do elo CC entre o retificador e o chopper. O valor calculado é apresentado em (34). Adota-se, portanto, dois capacitores de 470 μ F com associação em paralelo, observando-se a resistência interna dos mesmos.

$$C_{ret} = \frac{P_i}{f_{rede} \cdot (V_{rede_pico}^2 - V_{cc_min}^2)} = \frac{2000}{60(311^2 - 249^2)} = 960,06 \mu F \quad (34)$$

O sistema de retificação é baseado em diodos não controlados dispostos na estrutura ponte de Graetz. Para definir corretamente o componente a ser utilizado, valor de tensão reversa a qual os diodos estarão submetidos e a corrente condução eficaz de cada diodo são calculados como em (35) e (36), respectivamente.

$$V_{Dret_reversa} = \sqrt{2} \cdot V_{rede_eficaz} = 311,13V \quad (35)$$

$$I_{Dret_eficaz} = \frac{I_{max}}{\sqrt{2}} = 6,364A \quad (36)$$

Com base nesses parâmetros, optou-se pela ponte retificadora modelo KBPC3508 da *DC Components*, cuja tensão reversa é de 800V e a capacidade de condução de corrente eficaz é de 35 A. O sobre dimensionamento deu-se pela disponibilidade de componentes.

Já no conversor chopper, operando em modo de condução contínua não há componentes passivos, sendo necessário estabelecer apenas os esforços nos componentes ativos. O diodo de roda livre, responsável por evitar sobre tensão nos terminais da armadura tem valor de tensão reversa, definido de acordo com (37), e sua corrente de condução eficaz é dada por (38). Dessa forma, optou-se pelo modelo 30EPH06 da *International Rectifier*, que apresenta uma tensão reversa de 600V e corrente de condução eficaz de 30 A, satisfazendo a ambos os requisitos. O sobre dimensionamento deu-se pela disponibilidade de componentes.

$$V_{Dchopper_reversa} = V_{ret_max} = V_{rede_pico} = 311,13V \quad (37)$$

$$I_{Dchopper_eficaz} = I_{o_max} = 9A \quad (38)$$

A chave utilizada é do tipo IGBT. A tensão de bloqueio vista pela chave é calculada de acordo com (39), já a corrente de condução é dada por (40).

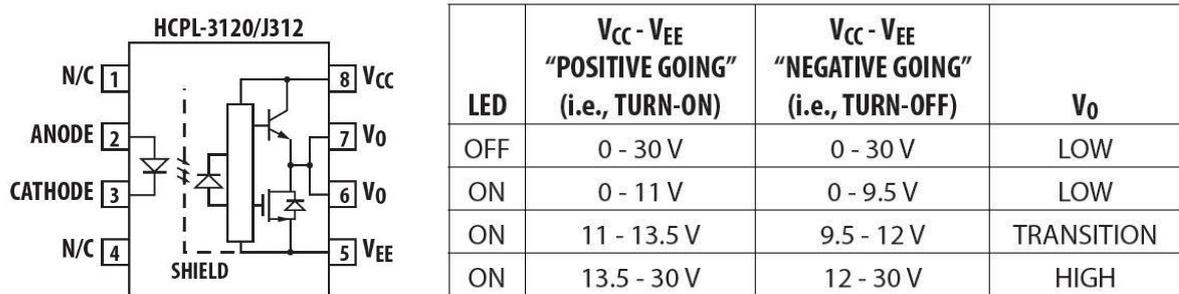
$$V_{Schopper_bloqueio} = V_{ret_max} = 311,13V \quad (39)$$

$$I_{Schopper_eficaz} = I_{o_max} = 9A \quad (40)$$

Com base nos valores calculados, a chave modelo IRG4PC50UD da *International Rectifier* foi selecionada. O IGBT utilizado suporta tensões reversas entre coletor e emissor de até 600V, e correntes de até 27 A em condução direta. O sobre dimensionamento deu-se pela disponibilidade de componentes.

Para o acionamento da chave, utilizou-se um *driver* opto isolado do tipo HCPL 3120, permitindo estabelecer a referência do mesmo no terminal emissor da chave. A Figura 18 apresenta o esquemático do circuito interno do referido opto acoplador e sua respectiva tabela verdade de operação.

Figura 18 – Esquemático e tabela verdade do opto acoplador HCPL 3120



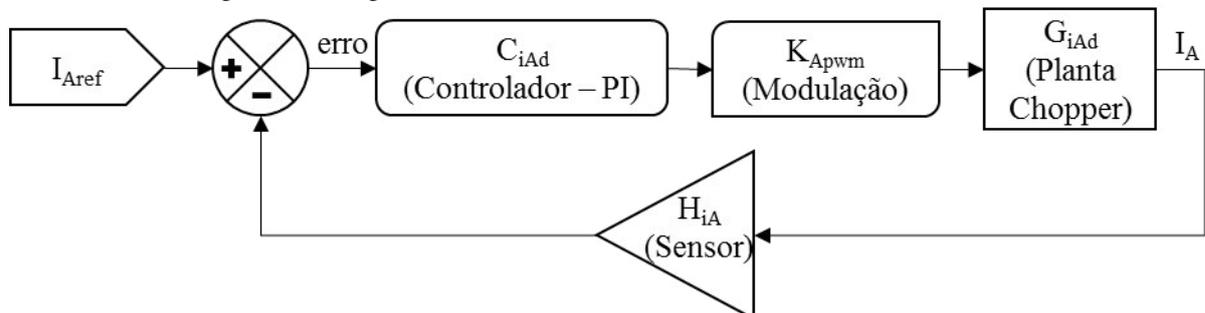
Fonte: datasheet do HCPL 3120 da Avago Technologies

O sistema de controle e modulação PWM (*Pulse Width Modulation*), foi realizado através de um microcontrolador de 16-bits do tipo dsPIC30F2020, da fabricante *Microchip*.

3.2.3 Projeto de controle do emulador

O controle do sistema de emulação, consiste de uma malha de torque/corrente cuja referência é dada através das funções aerodinâmica da turbina eólica emulada. A malha de corrente é definida como no diagrama de controle apresentado na Figura 19.

Figura 19 – Diagrama da malha de controle de corrente de armadura do MCC

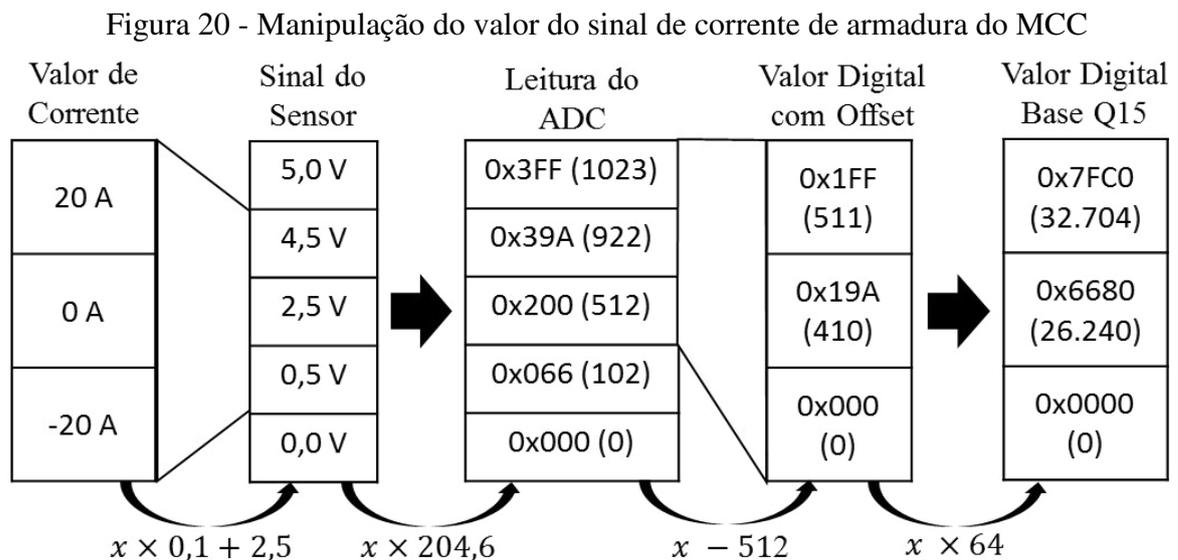


Fonte: desenvolvido pelo autor.

Para realizar o controle de corrente faz-se necessário realizar o sensoriamento da corrente no enrolamento de armadura do motor CC, tarefa feita através da utilização do sensor do tipo ACS712T-20^a da fabricante *Allegro MicroSystems*. Este tipo de sensor permite leituras de correntes de 20 A bidirecional, apresentando um offset de sinal de 2,5V e tendo uma relação

tensão/corrente de 0,1V/A. O sistema microcontrolado utilizado apresenta um conversor analógico digital de 10-bits, através do qual é feita a aquisição do sinal de corrente, e posterior remoção do offset, uma vez que o sistema não apresenta sistema de frenagem regenerativa no motor, não necessitando de leitura bidirecional de corrente.

O controlador Proporcional Integrador (PI) visto na Figura 19 é feito por meio da máquina DPS presente nos microcontroladores da família dsPIC30F. Neste dispositivo, é possível realizar a multiplicação de duas variáveis de 16-bits, armazenando o resultado em um acumulador de 40-bits. Uma vez a multiplicação é feita em números inteiros, faz-se necessário a utilização da base Q15, na qual os valores fracionários dos coeficientes do PI são transformados em valores inteiros na escala de 2^{15} . A Figura 20 apresenta a manipulação feita com o sinal de corrente.



Fonte: desenvolvido pelo autor.

Com base na manipulação do sinal de corrente descrito, obtém-se como valor de ganho final do sensoriamento de corrente (H_{iA}) é apresentado em (41). Nesse valor já estão inclusos o ganho do sensor, a manipulação de *offset* e o ganho do ADC.

$$H_{iA} = 1309,44 \quad (41)$$

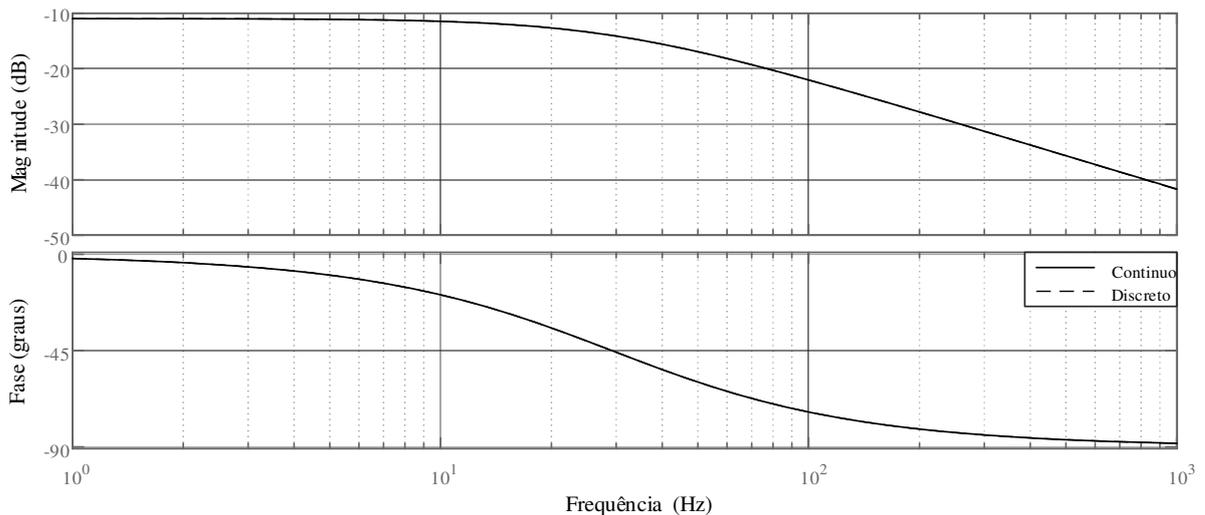
Além do ganho do sensor, é importante considerar o ganho referente à modulação PWM (K_{Apwm}) dado por um ganho inverso a amplitude da portadora digital. Considerando a frequência de operação do microcontrolador, e de chaveamento do conversor, tem-se que o pico da portadora digital é de 32.000, assim, o valor de K_{Apwm} é dado em (42).

$$K_{Apwm} = \frac{1}{32000} = 31,25 \times 10^{-6} \quad (42)$$

Já a planta em questão, a função de transferência que relaciona a corrente de armadura (I_A) com o ciclo de operação (d) do conversor chopper é representado por G_{id} e é definido em (43), obtido através do modelo da chave PWM (VORPERIAN, 1990). A Figura 21 apresenta o diagrama de Bode da função de transferência de malha aberta sem compensador da corrente no indutor pelo ciclo de trabalho.

$$G_{iAd} = \frac{V_i}{sL_a + R_a} = \frac{0,3111}{0,006s + 1,1} \quad (43)$$

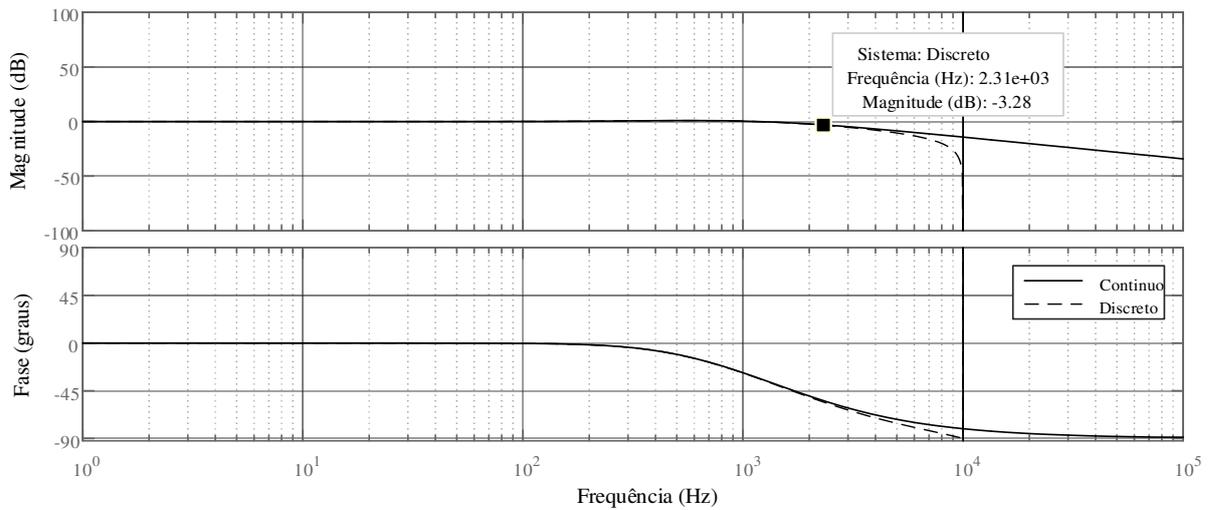
Figura 21 – Diagrama de Bode de malha aberta sem compensador da corrente do MCC



De posse de K_{Apwm} , H_{iA} e G_{iAd} é possível utilizando a função *pidtunes* do Matlab, definir a função de transferência do controlador C_{iAd} que ajusta a frequência de cruzamento e a margem de fase para às desejadas. Adotando uma margem de fase de 80° e uma frequência de cruzamento de 2kHz, de acordo com (BUSO; MATTAVELLI, 2006b), tem-se que a função de transferência do compensador é definida como em (44).

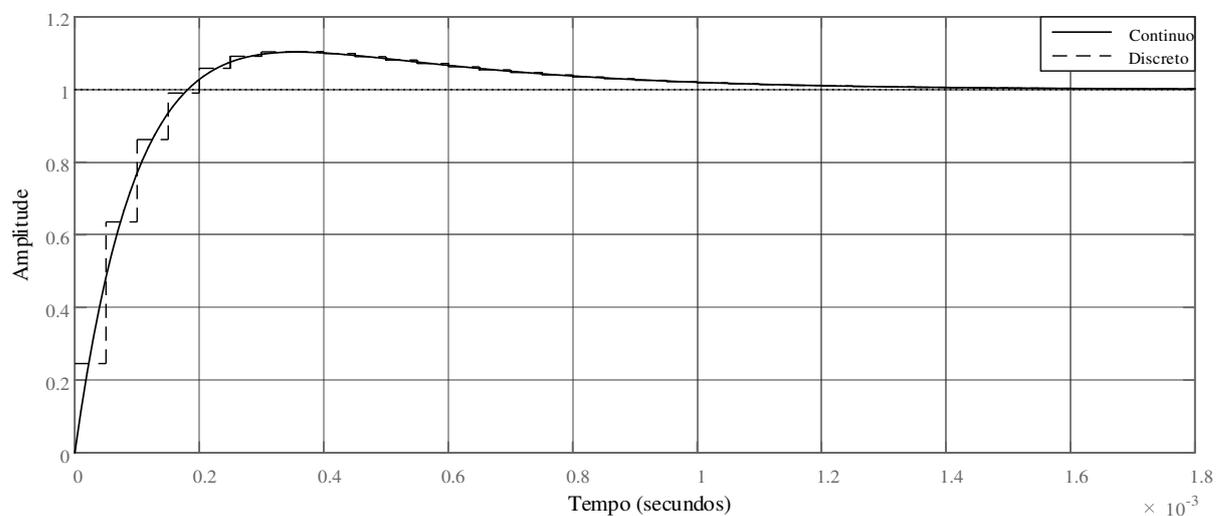
$$C_{iAd} = \frac{238s + 5,726e5}{s} \quad (44)$$

Figura 23 - Diagrama de Bode de malha de fechada da corrente de armadura do MCC.



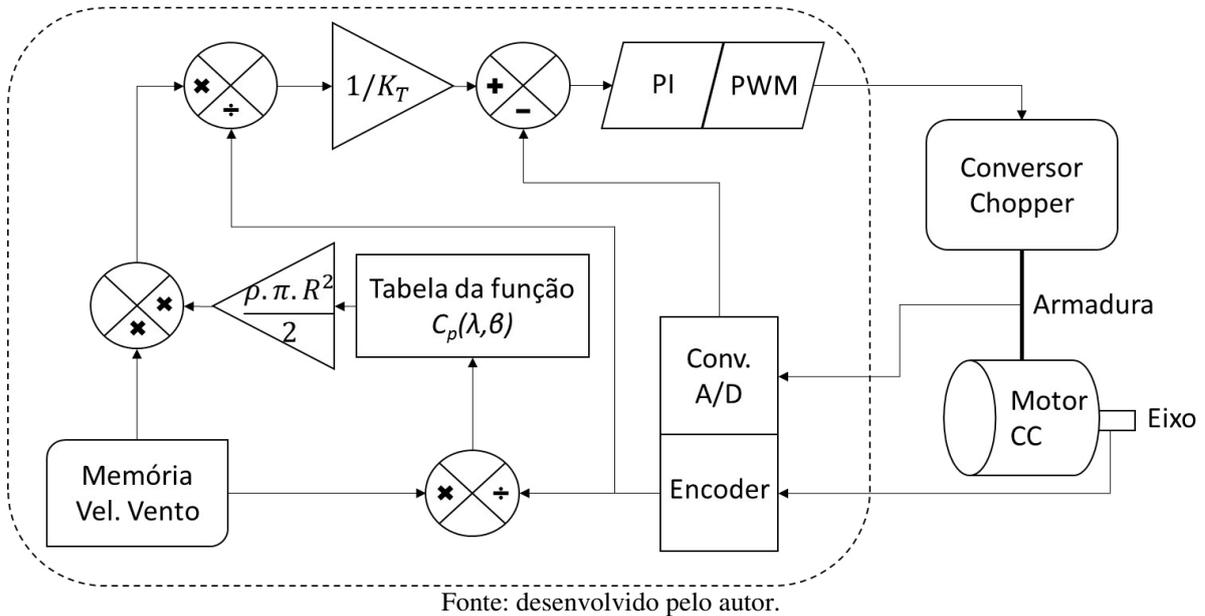
Já a Figura 24 permite observar o comportamento da corrente de armadura do MCC, quando há a aplicação de um degrau unitário na referência da mesma. Observa-se que o tempo de acomodação fica em aproximadamente 1,2 ms.

Figura 24 – Resposta ao degrau unitário da malha de corrente de arma do MCC.



Finalizado o projeto da malha de corrente, faz-se necessário desenvolver o algoritmo responsável por determinar a referência de corrente a ser enviada ao controlador. A Figura 25 apresenta o diagrama de cálculo da corrente de referência de armadura.

Figura 25 - Diagrama de controle de torque do emulador de turbina eólica.
dsPIC30F2020



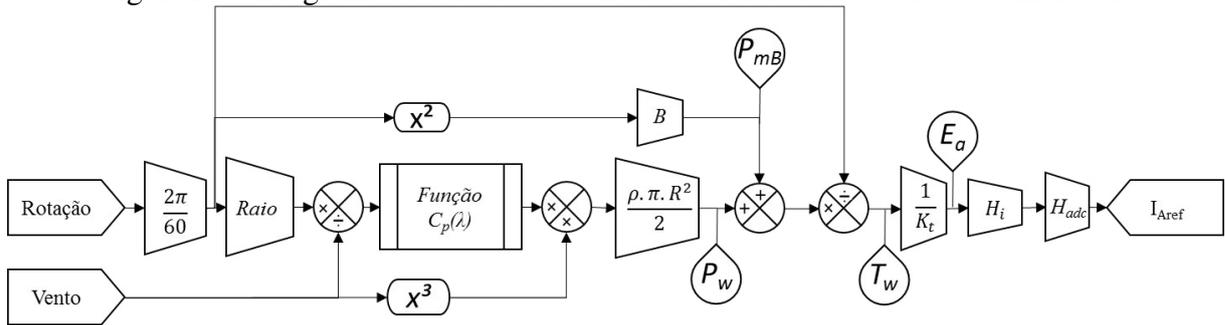
O valor de referência para a corrente do indutor (I_{Aref}) é obtido através das curvas de potência e torque aerodinâmicos da turbina emulada. Como visto na Figura 25, o valor de referência de corrente é calculado através de uma função que relaciona a velocidade do vento e a velocidade de rotação dentre outros parâmetros.

A velocidade do vento emulada pelo sistema tem seu perfil gerado via software computacional e posteriormente gravada em uma variável vetorial na memória do microcontrolador.

Na codificação do sistema de emulação, a leitura de rotação foi realizada em rpm, enquanto o valor de λ utiliza a rotação em rad/s, sendo necessário considerar um ganho para a mudança de base da mesma.

Para o cálculo do coeficiente de potência (c_p) utiliza-se uma tabela do tipo *Look-Up Table* de 256 posições formulada através da equação (3), na qual são armazenados os valores do coeficiente c_p . Visando minimizar o custo computacional, a tabela desenvolvida agrega além do valor de c_p , o ganho H_{iA} do sensor de corrente, a constante multiplicativa da potência ($\rho \cdot \pi \cdot \text{Raio}^2 / 2$) e o coeficiente de torque K_T . A Figura 26 apresenta uma versão estendida do fluxograma de cálculo de I_{Aref} , já a Figura 27 mostra a versão reduzida implementada no código do microcontrolador.

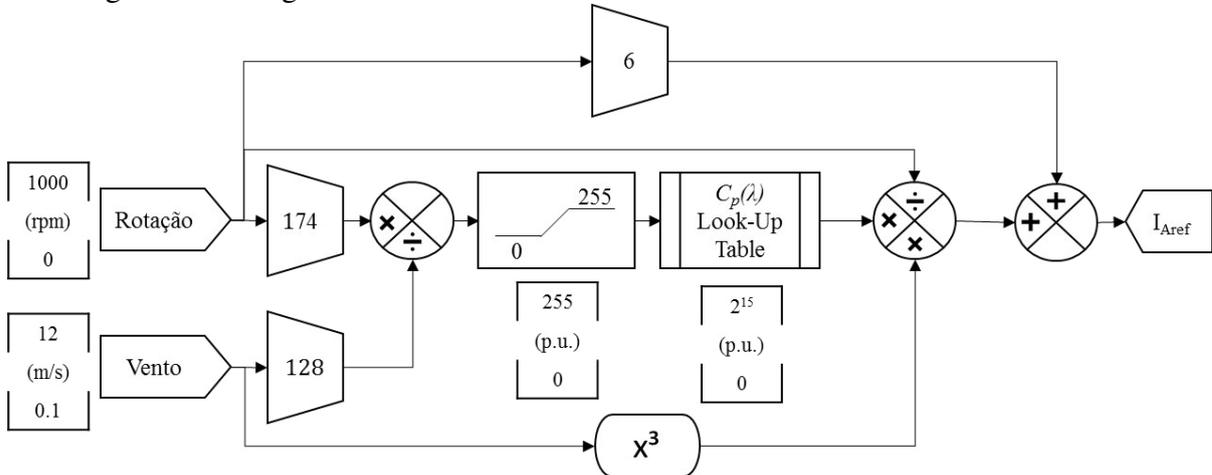
Figura 26 – Diagrama de cálculo da corrente de referência do sistema emulador.



Fonte: desenvolvido pelo autor.

Como é possível observar na Figura 27, há uma parcela de I_{Aref} que se refere à componente de torque de atrito mecânico da máquina. O valor de referência obtido através da *Look-Up Table* reflete o torque líquido aplicado no eixo, desconsiderando, portanto, o atrito mecânico, que por sua vez, necessitam de uma parcela de torque que o anule, fazendo necessária a adição da parcela supracitada. Essa desconsideração é feita devido à complexidade na definição da mesma, mas acarreta em uma divergência considerável em relação aos valores de simulação.

Figura 27 – Diagrama do sistema de cálculo de corrente de referência do emulador



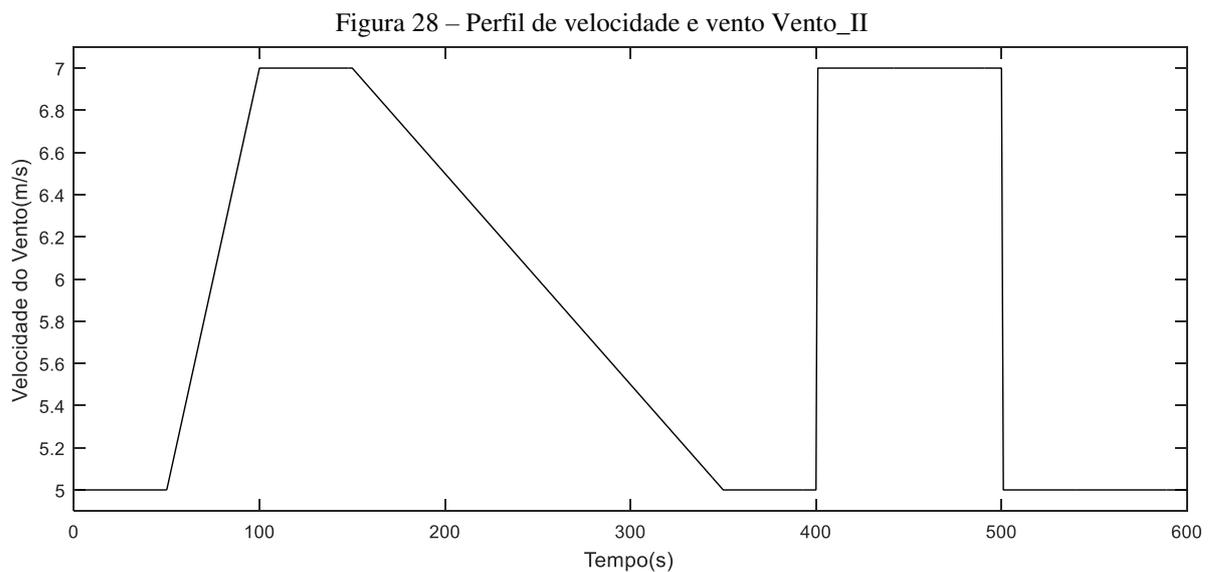
Fonte: desenvolvido pelo autor.

3.3 Perfis de ventos adotados

A fim de observar o comportamento dinâmico de algoritmos MPPT, o sistema emulador deve ser capaz de reproduzir comportamentos de ventos estáticos e dinâmicos. Dessa forma, 4 perfis de ventos são definidos como padrão neste trabalho.

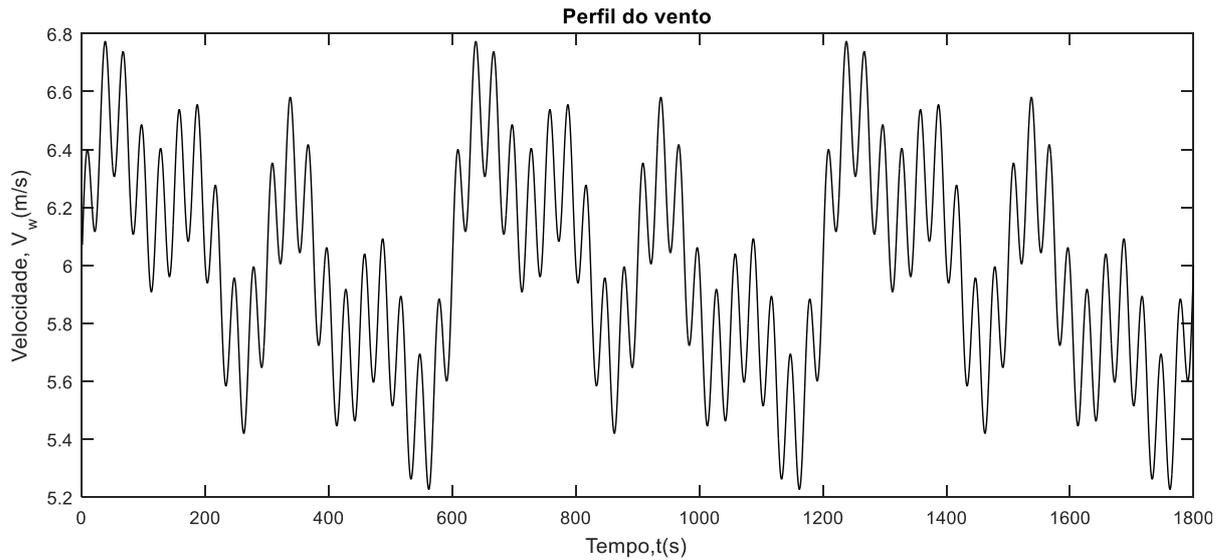
Vento I – Nesse perfil, adota-se um valor fixo de vento igual a 5m/s. O objetivo desse perfil é realizar a validação inicial dos algoritmos MPPT, permitindo observar a capacidade e convergência do mesmo.

Vento II – Tendo o algoritmo mostrando-se eficiente para o perfil anterior, o mesmo é então apresentado a um perfil de vento com variações bem definidas. Esse perfil visa iniciar os testes dinâmicos do algoritmo MPPT. A Figura 28 apresenta o gráfico em função do tempo desse perfil.



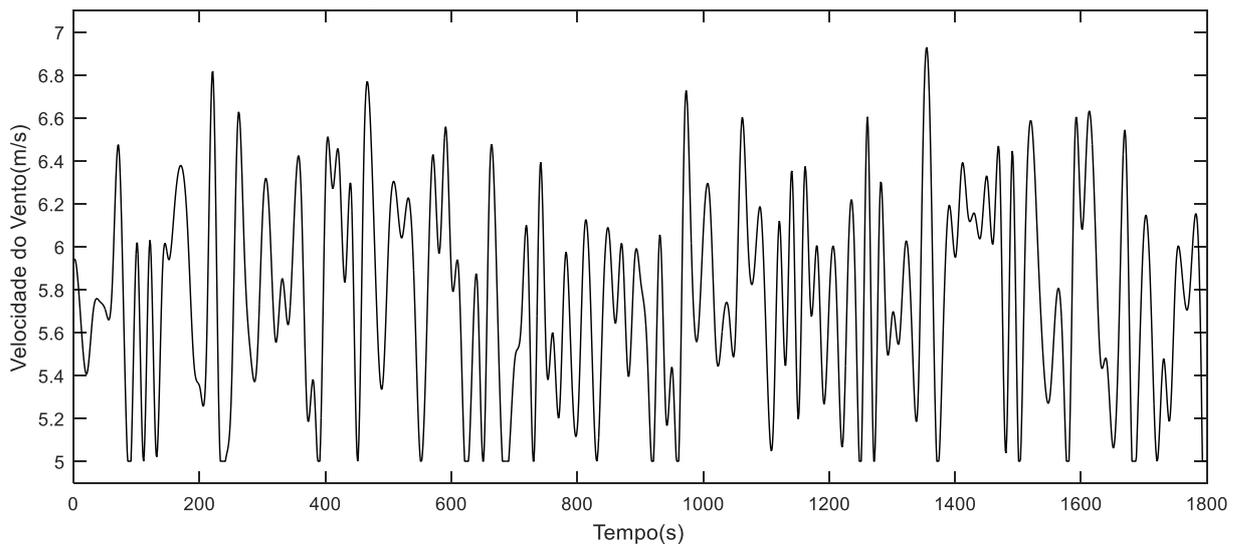
Vento III – O comportamento do vento neste perfil é definido através do somatório de 4 funções seno, cujos períodos são de 30, 150, 300 e 600 segundos, a amplitude de operação do vento varia de 5 a 7 m/s. A Figura 29 apresenta esse perfil. Salienta-se que a periodicidade do perfil facilita a aprendizagem de algoritmos neurais, contudo, não representa fielmente o comportamento real do vento.

Figura 29 – Perfil de velocidade e vento Vento_III



Vento IV – Buscando apresentar um perfil de comportamento de vento mais fiel ao real, propõe-se o quarto perfil de vento, que é gerado de forma pseudoaleatória através das funções randômicas presente no software Matlab. Utiliza-se na função aleatória a distribuição de Weibull com coeficientes de escala de 6 e de forma de 12, o que estabelece valores aleatórios, mas que estejam contidos dentro da faixa de operação do emulador de 5 a 7 m/s. A Figura 30 apresenta tal perfil.

Figura 30 – Perfil de velocidade e vento Vento_IV



Todos os códigos utilizados na programação do microcontrolador dsPIC30F2020 do emulador são apresentados no Apêndice B. Os esquemáticos dos circuitos, bem como o arquivo das placas são apresentados no Apêndice C.

3.4 Sistema de operação do gerador (Boost / PMSG)

No sistema desenvolvido, além do motor de corrente contínua utilizado para emular o comportamento aerodinâmico da turbina, utiliza-se um gerador síncrono de imã permanente acoplado diretamente ao eixo do motor. O gerador utilizado é o mesmo presente nos sistemas de geração do modelo Gerar 246 da Enersud, cujos parâmetros segundo o fabricante são dados na Tabela 5. Contudo, por não serem utilizadas pás, mas si um sistema motor a fim de emular as pás, grande parte desses dados alteram-se no projeto desenvolvido.

Este gerador foi foco de ensaios realizados em trabalhos anteriores (MACHADO, 2007), neste estudo, outros parâmetros importantes da máquina foram obtidos, tais parâmetros são apresentados na Tabela 6.

Tabela 5 – Dados do sistema de geração Gerar 246 da Enersud

Características Técnicas do Aerogerador Gerar 246	
Diâmetro da hélice	2,46m
Potência a 12 m/s	1000 Watt
Número de pás	3
Velocidade de partida	2,2 m/s
Torque de partida	0,3 Nm
Controle de velocidade	Stall
Proteção contra altas velocidades	Active Stall (Controle de Passo)
Sistema Magnético	Neodímio (imã permanente)
Sistema elétrico	Trifásico
Tensão de Saída	12/24/48/300 volts
Alternador	Fluxo Axial (encapsulado em epóxi, resistente a água)
Escovas da cabeça giratória	Redundante (duas por fase)
Peso total	32 Kg
Material Anti Corrosão	Alumínio/ Inox/ Mat. Galvanizado
Balanceamento	Dinâmico (confirmação após pintura)
Principais Funções	Carregador de baterias

Fonte: *website* da Enersud.

Tabela 6 – Dados de ensaio do gerador Gerar 246 – Enersud

Parâmetros ensaiado do gerador Gerar 246 - Enersud	
Resistência do enrolamento	0,5 Ohms
Indutância própria dos enrolamentos	3,35 mH
Indutância mútua dos enrolamentos	3,09 mH
Tensão induzida de pico fase-neutro	0,061 V/rpm
Momento de inércia	0,065 Kg.m ²
Coeficiente de atrito viscoso	0,00467 Nm/(rad/s)

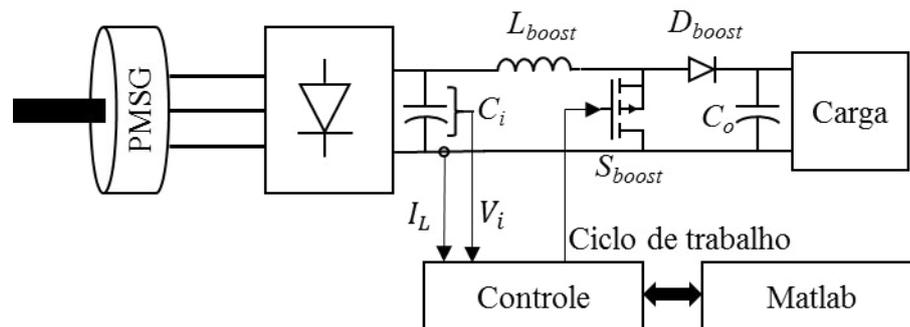
Fonte: (MACHADO, 2007)

3.4.1 Projeto do sistema de operação do gerador

Para que o gerador trifásico passe a produzir energia, é necessário apenas que o sistema emulador entregue o torque mínimo necessário para rotação do eixo e que uma carga esteja conectada à saída do gerador. Contudo, nessas condições, não há garantia que o gerador estará extraindo o valor máximo de potência. Para tal, é necessário ajustar a carga na saída do mesmo, de forma que a velocidade de rotação, mantenha-se próximo a um valor de referência ótima.

Com o objetivo de controlar a carga e conseqüentemente a velocidade de operação, utiliza-se um conversor chaveado do tipo Boost. Controlando a corrente no indutor do conversor é possível realizar a frenagem do gerador. A Figura 31 apresenta um diagrama do circuito do conversor Boost implementado. Os parâmetros básicos de projeto do conversor Boost são apresentados na Tabela 7.

Figura 31 – Diagrama do sistema controlador de carga tipo Boost



Fonte: desenvolvido pelo autor.

Tabela 7 – Dados nominais do conversor Boost

Parâmetros nominais do conversor Boost	
Tensão nominal máxima de entrada (V_i)	120 Vcc
Tensão nominal máxima de saída (V_o)	150 Vcc
Ciclo de trabalho nominal (D)	0,2
Frequência de chaveamento (f_s)	20 KHz
Oscilação máxima de corrente no indutor (ΔI_L)	20%
Oscilação máxima de tensão na entrada (ΔV_i)	10%
Oscilação máxima de tensão na saída (ΔV_o)	1%
Potência nominal máxima de entrada (P_i)	480 W

A partir dos dados nominais do projeto do conversor Boost é possível dimensionar os componentes passivos e determinar os esforços sobre os componentes tomando como base Barbi e Martins (2006).

Uma vez que o ponto de operação do conversor varia de acordo com a velocidade de rotação e a velocidade do vento, diversos parâmetros devem ser calculados em função dos mesmos, com o objetivo de evitar sobre dimensionamento dos componentes. Inicialmente calcula-se a corrente máxima levando em conta o vento máximo de 12 m/s, a potência e a tensão de entrada em função da velocidade de rotação, como mostra a equação (46), onde K_{ret} é o ganho de tensão do retificador, K_e é a constante que relaciona a rotação com a tensão de saída do PMSG, λ_{opt} é o ponto de operação onde o valor de C_p alcança seu valor máximo (C_{pmax}).

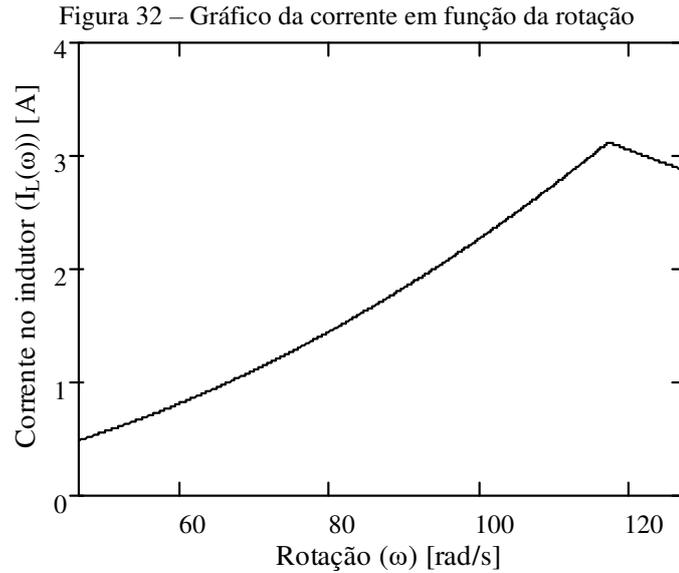
$$I_{Lmax} = \frac{0,5 \cdot \rho \cdot \pi \cdot R_{raio}^2 \left(\frac{\omega \cdot R_{raio}}{\lambda_{opt}} \right)^3 \cdot C_{pmax}}{K_{ret} \cdot K_e \cdot \omega} \quad (46)$$

Através da observação do gráfico apresentado na Figura 32, verifica-se uma corrente máxima de 3,1A, utiliza-se no projeto do indutor um valor de 4A.

Para definir o valor do indutor, observa-se o gráfico da variação de corrente em função da rotação, determina-se o ponto onde a variação é maior e o respectivo valor de rotação (ω_{maxALL}), então, calcula-se o indutor de acordo com (47).

O gráfico utilizado para obter o ponto de rotação para o qual a variação de corrente é máxima é apresentado na Figura 33.

O cálculo do capacitor de entrada é feito levando-se em conta a oscilação do sinal senoidal trifásico de saída do gerador síncrono de ímã permanente, que por sua vez varia proporcionalmente à velocidade de rotação da máquina.

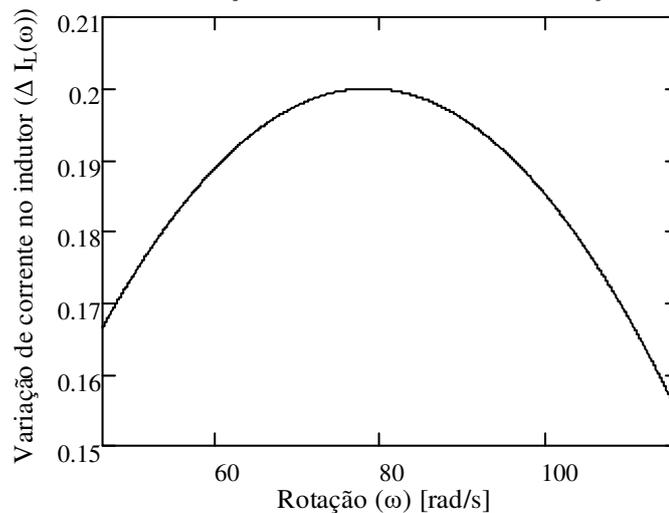


Fonte: desenvolvido pelo autor.

$$L = \frac{\left(1 - \frac{K_{ret} \cdot K_e \cdot \omega_{max \Delta L}}{V_o}\right) \cdot (K_{ret} \cdot K_e \cdot \omega_{max \Delta L})}{f_s \cdot \Delta I_L \cdot I_{Lmax}} = 3,024mH \quad (47)$$

A equação (48) apresenta o cálculo do valor do capacitor enquanto a equação (49) determina o valor máximo de resistência interna do capacitor de entrada. Com base em tais valores, adota-se um capacitor de 470μF, com uma resistência interna de 0,7Ω, fabricado pela EPCOS.

Figura 33 – Gráfico da variação de corrente do indutor em função da rotação



Fonte: desenvolvido pelo autor

$$C_i = \frac{P_o}{f_{ret} \cdot \Delta V_i \cdot V_i^2} = 281,48 \mu F \quad (48)$$

$$r_{Ci\max} = \frac{\Delta V_i \cdot V_i}{\Delta I_L \cdot I_L} = 1,2 \Omega \quad (49)$$

O capacitor de saída é determinado através da equação (50) e tem o objetivo de reduzir a oscilação de tensão em cima da carga. O valor da resistência interna máxima do capacitor é calculado pela equação (51). Adotou-se, portanto, a associação de dois capacitores de 100 μ F da EPCOS cuja resistência interna equivalente é de 0,85 Ω .

$$C_o = \frac{I_o \cdot D}{\Delta V_o \cdot f} = 65,778 \mu F \quad (50)$$

$$r_{Co\max} = \frac{\Delta V_o \cdot V_o}{\Delta I_L \cdot I_L} = 2,419 \Omega \quad (51)$$

O diodo que opera em bloqueio durante o carregamento do indutor apresenta uma corrente de condução calculada como em (52). A tensão reversa que o mesmo deve suportar é calculada na equação (53). Pela disponibilidade de componentes, adotou-se um diodo do tipo MBR20200, cuja máxima tensão reversa é de 600V e que suporta até 30A de corrente em condução direta.

$$I_{Dboost} = I_L \cdot (1 - D) = 1,04 A \quad (52)$$

$$V_{Dboost} = V_o = 150 V \quad (53)$$

Já a chave utilizada no carregamento do indutor tem sua corrente eficaz de condução calculada de acordo com (54), enquanto a tensão reversa que a mesma deve suportar é definido como na equação (55). Optou-se pela chave do tipo IRFP260N fabricada pela *International Rectifier*, devido a disponibilidade em laboratório. Essa chave apresenta uma capacidade de condução direta de 50A e uma tensão de bloqueio máxima de 200V

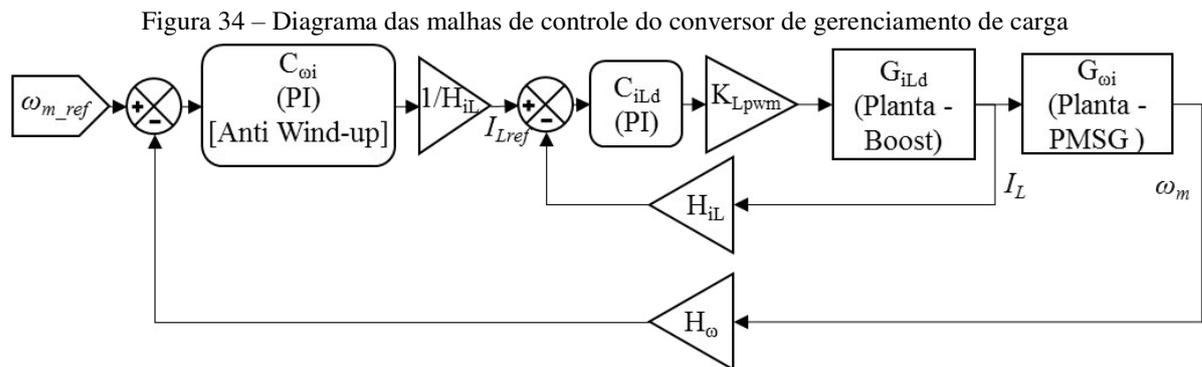
$$I_{Sboost} = I_L \cdot \sqrt{D} = 3,441 A \quad (54)$$

$$V_{Sboost} = V_o = 150 V \quad (55)$$

O sistema de acionamento das chaves é por feito por um microcontrolador do tipo dsPIC30F4011. O sinal enviado pelo microcontrolador passa por um driver do tipo MCC33152 da *On Semiconductor*. Os códigos utilizados na programação do microcontrolador são apresentados no Apêndice De esquemáticos do circuito no Apêndice E.

3.4.2 Projeto as malhas de controle do sistema de gerenciador de carga.

O sistema de controle do conversor Boost é composto por duas malhas cascadeadas: a malha mais interna, e mais rápida é a referente à corrente no indutor, já a mais externa, responsável por define o valor da referência de corrente, trata do controle de velocidade de rotação. O diagrama destas malhas é apresentado na Figura 34.



Fonte: desenvolvido pelo autor.

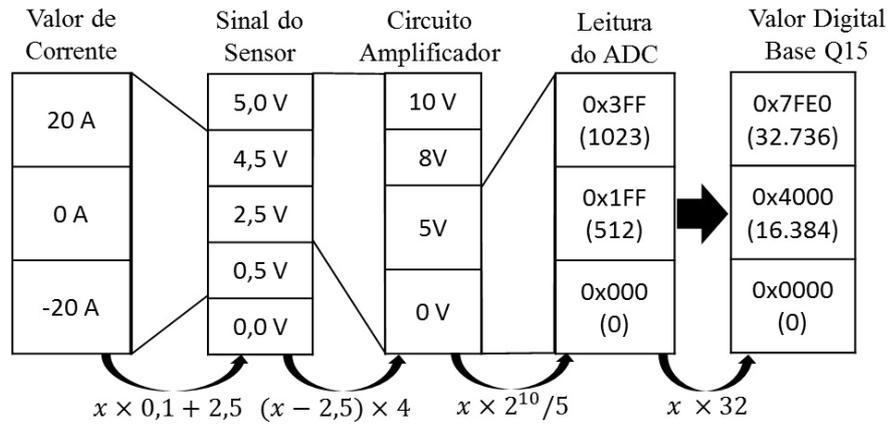
Na Figura 34 são apresentadas as duas malhas de controle, nela, C_{oi} é o controlador PI com anti wind-up da malha de rotação. O ganho H_{iL} ajusta a referência de corrente à escala do controlador de corrente definido como C_{iLd} . K_{Lpwm} é o ganho de modulação, G_{iLd} a planta de corrente, G_{oi} a planta de rotação e H_{iL} e H_{ω} os respectivos ganhos de realimentação.

Assim como no conversor de *chopper*, para controlar a corrente é necessário realizar o sensoriamento desta, novamente utiliza-se o sensor de corrente ACS712T-20^a da fabricante *Allegro MicroSystems*. Desenvolve-se, contudo, um circuito amplificador, que permite remover o offset de 2,5V do sinal, aplicando ainda um ganho de x4, tendo uma relação tensão/corrente de 0,4V/A. Utiliza-se um conversor analógico digital de 10-bits interno ao microcontrolador utilizado.

Ambos os controladores PI vistos na Figura 34 são implementados através da máquina DSP de um microcontrolador modelo dsPIC30F4011 fabricado pela Microchip. A escolha deste modelo se deu pela disponibilidade em laboratório. Novamente, faz-se uso da

base Q15. A Figura 35 apresenta a manipulação feita com o sinal de corrente, um pouco mais simples, devido ao circuito amplificador implementado.

Figura 35 - Manipulação do valor do sinal de corrente do indutor do conversor Boost



Fonte: desenvolvido pelo autor.

Com base na manipulação do sinal de corrente descrita, o valor de ganho de realimentação da corrente do indutor (H_{iL}) é apresentado em (56).

$$H_{iL} = 2621,44 \quad (56)$$

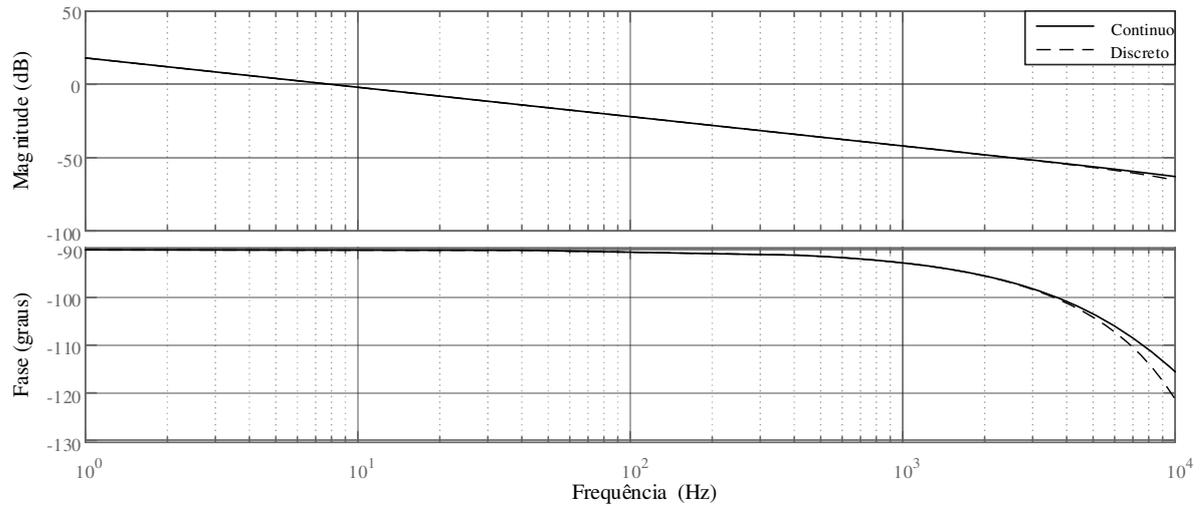
Já o ganho da modulação PWM (K_{Lpwm}) é dado em (57), onde é considerada uma portadora digital (V_{Ltri}) de com amplitude de 998.

$$K_{Lpwm} = \frac{1}{V_{Ltri}} = \frac{1}{998} = 1,002 \times 10^{-3} \quad (57)$$

A função de transferência que relaciona a corrente no indutor (I_L) com o ciclo de operação (d) do conversor Boost é denominada G_{iLd} e é definida em (58). Utilizou-se a função de transferência aproximada, obtida por meio do modelo da chave PWM (VORPERIAN, 1990). A Figura 36 apresenta o diagrama de Bode da função de transferência de malha aberta sem compensador da corrente no indutor pelo ciclo de trabalho.

$$G_{iLd} = \frac{V_o}{sL} = \frac{0,1503}{0,003s} \quad (58)$$

Figura 36 - Diagrama de Bode de malha aberta sem compensador da corrente no indutor do Boost.



Fonte: desenvolvido pelo autor.

Novamente utiliza-se a função *pidtune* do Matlab para definir a função de transferência do controlador C_{iLd} , baseado nos valores de K_{Lpwm} , H_{iL} , G_{iLd} , na frequência de cruzamento desejada e na margem de fase desejada.

Define-se uma margem de fase de 70° e uma frequência de cruzamento de 2kHz, de acordo com (BUSO; MATTAVELLI, 2006a). Dessa forma, obtém-se a função de transferência do compensador como apresentado em (59).

$$C_{iLd} = \frac{243,9s + 7,95e5}{s} \quad (59)$$

Discretizando C_{id} é possível encontrar a equação de controle a ser utilizada no microcontrolador. Utilizando uma taxa de amostragem com o dobro da frequência de chaveamento (Incluir alguma tabela com parâmetro assumidos e especificações do projeto de controle), tem-se a equação de controle, como dada em (60), na qual, são apresentadas as equações de controle na base decimal e na base Q15.

$$u(t) = (0,0945)e(t) - (0,0871)e(t-1) + u(t-1) \quad (60)$$

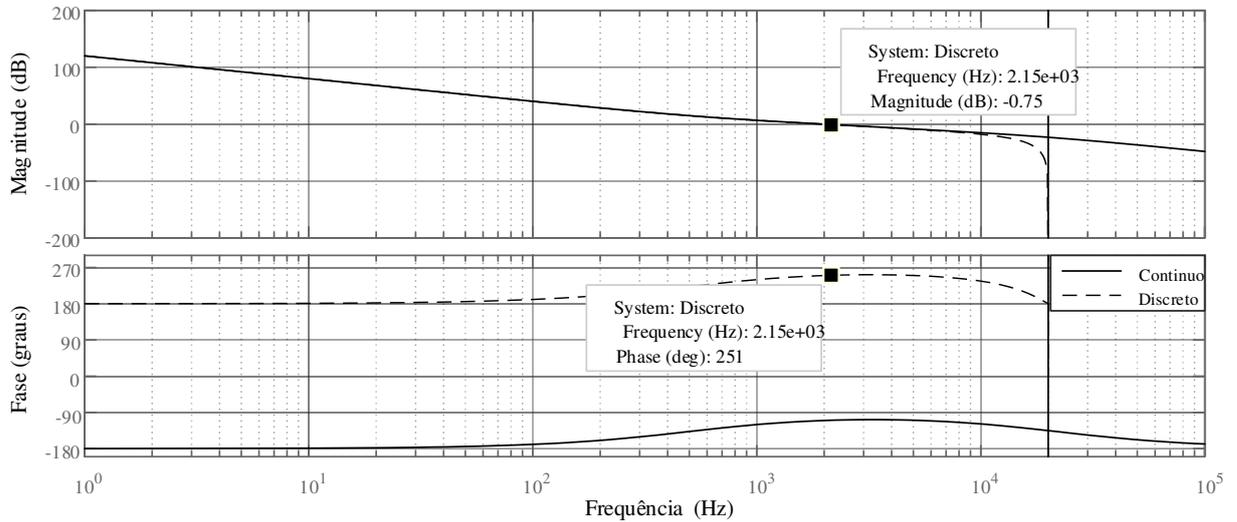
\downarrow
Q15

$$u(t) = (3095)e(t) - (2853)e(t-1) + (32767)u(t-1)$$

Na Figura 37 é apresentado o diagrama de bode de malha aberta da corrente no indutor do conversor Boost, já considerando o compensador. Observando o ponto onde a

magnitude atinge 0dB é possível notar que, após discretizar o controlador, obtém-se uma frequência de cruzamento de aproximadamente 2kHz e uma margem de fase de 71°, valores próximos aos desejados.

Figura 37 – Diagrama de Bode de malha de aberta compensada da corrente no indutor do Boost



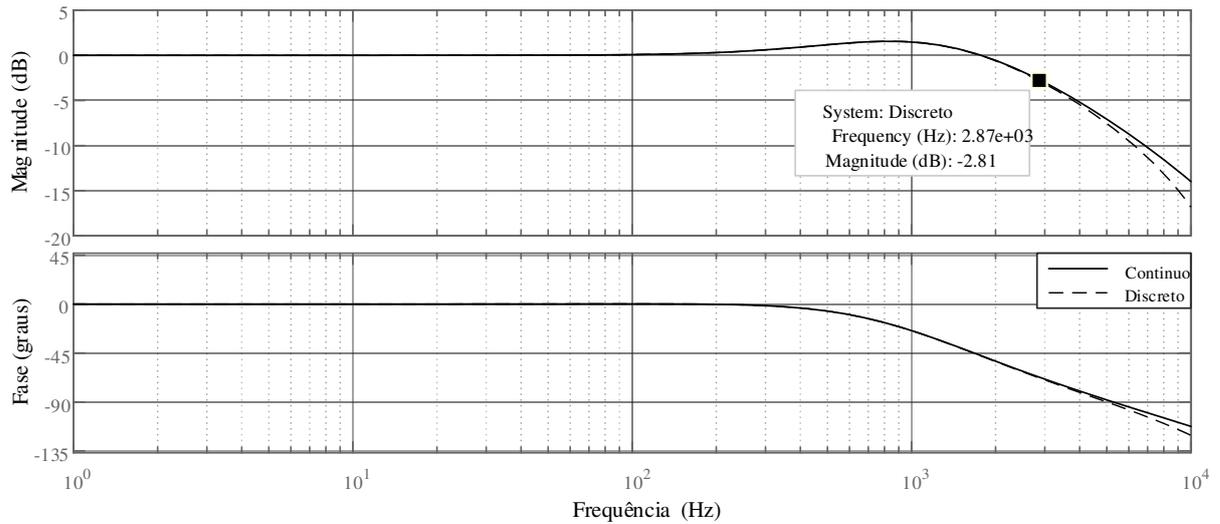
Fonte: desenvolvido pelo autor.

Analisando o diagrama de Bode de malha fechada da corrente no indutor do conversor Boost apresentado na Figura 38, é possível notar que a frequência de corte obtida se encontra acima da frequência de cruzamento, em aproximadamente 2,9kHz (-3dB).

Já a Figura 39 permite observar o comportamento da corrente no indutor do conversor Boost, quando um degrau unitário é aplicado como referência. Observa-se que o tempo de acomodação fica em aproximadamente 1,0 ms.

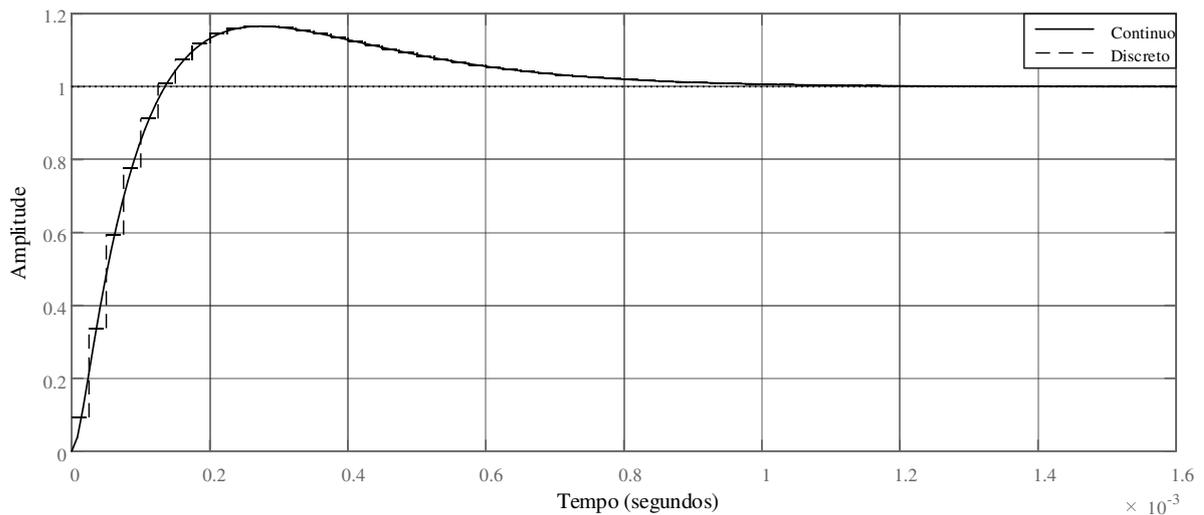
Já a malha de velocidade de rotação, responsável por definir a referência da corrente do indutor, utiliza o mesmo sistema de sensoriamento de rotação do emulador, apresentado no tópico 3.1. Contudo, interno ao microcontrolador, o sinal de referência de corrente recebe um ganho de 2^5 para adequar-se à escala da malha de corrente.

Figura 38 - Diagrama de Bode de malha fechada da corrente de armadura do MCC.



Fonte: Desenvolvido pelo autor.

Figura 39 – Resposta ao degrau unitário da malha de corrente no indutor do conversor Boost



Fonte: desenvolvido pelo autor.

Assim como as demais malhas de controle, a malha de controle de rotação é implementada através da máquina DSP do microcontrolador dsPIC30F4011, fazendo uso da base Q15. A leitura do encoder é feita em pulsos, que posteriormente é convertida em rotações por minuto (rpm). Uma vez que todos os parâmetros mecânicos se apresentam no sistema internacional (SI) de unidades, considera-se que o sensor de rotação insira um ganho de $(60/2\pi)$. Dessa forma, o ganho do sensor de rotação (H_ω) fica definido como em (61).

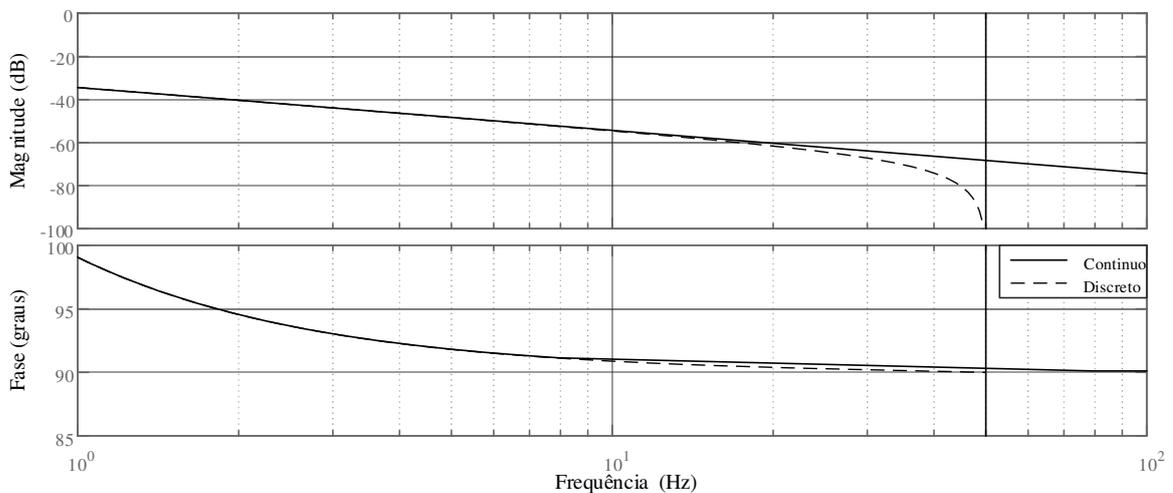
$$H_\omega = \frac{60}{2\pi} \cdot 2^5 = 305,58 \quad (61)$$

Como pode ser visto na Figura 34, o ganho de realimentação de corrente aparece como um ganho na saída do controlador de corrente, estando assim, presente na função de transferência de malha aberta do mesmo. Devido a diferença entre as frequências de cruzamento, as malhas podem ser consideradas desacopladas e a planta interna acaba se aproximando de uma constante unitária.

A função de transferência que relaciona a velocidade de rotação (ω_m) com o a corrente no indutor do conversor Boost é denominada $G_{\omega i}$ e é definida em (62), onde K_T refere-se a constante que relaciona torque e corrente e J e B são a inércia do sistema e o coeficiente de atrito do sistema MCC-PMSG. A função de transferência foi obtida como em (MOHAN, 2012), contudo, não foram desprezadas as perdas mecânicas. O diagrama de bode da malha de rotação do eixo é apresentado na Figura 40.

$$G_{\omega i} = \frac{1}{H_{iL}} \frac{K_T}{J.s + B} = \frac{-0.0807}{0,0416s + 0,0416} \quad (62)$$

Figura 40 - Diagrama de Bode de malha aberta sem compensador da rotação do eixo

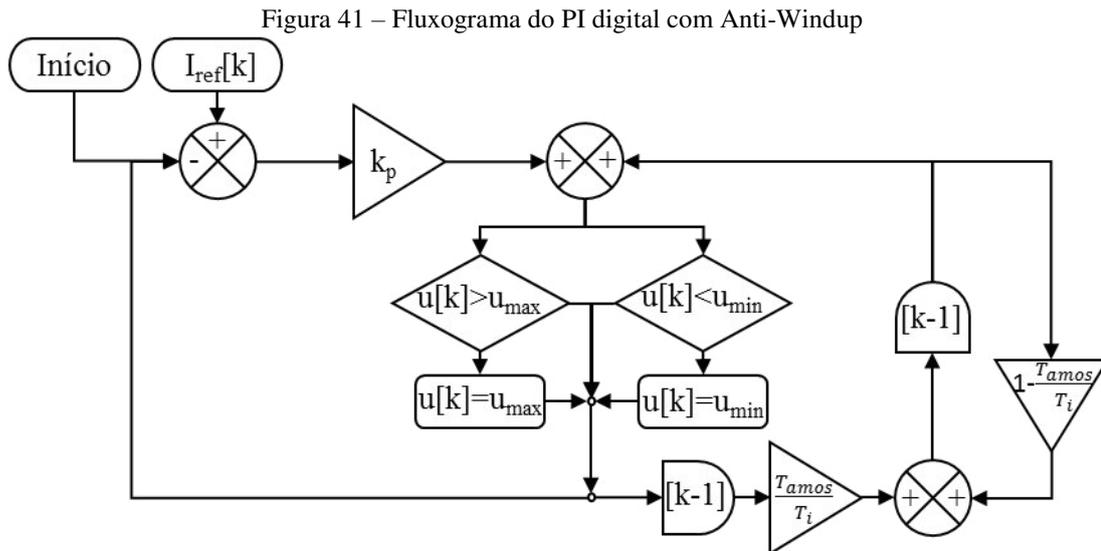


Fonte: desenvolvido pelo autor.

Utilizando novamente a função *pidtune* do Matlab, define-se a função de transferência do controlador $C_{\omega i}$, baseado nos valores de $H_{\omega i}$, H_{iL} e $G_{\omega i}$. Define-se uma margem de fase de 70° e uma frequência de cruzamento de 5Hz. Tal frequência de cruzamento é adotada como sendo 20 vezes menor que a frequência de amostragem de 100Hz. Dessa forma, obtém-se a função transferência do compensador como apresentado em (63).

$$C_{\omega i} = \frac{-240,7s - 3028}{s} \quad (63)$$

Discretizando C_{oi} é possível encontrar os parâmetros K_i e K_p discretos. A fim de evitar problemas relacionados a saturação do elemento integrador do controlador, utiliza-se um controlador PI com *Anti-Windup* na estrutura de defasagem do termo integrativo (NETO, 2012). O algoritmo apresentado na Figura 41 descreve a forma de implementação do controle digital.



Fonte: desenvolvido pelo autor

Utilizando uma taxa de amostragem de 100Hz, mantendo-se uma precisão de 0,6 rpm. Os coeficientes utilizados são apresentados em (64).

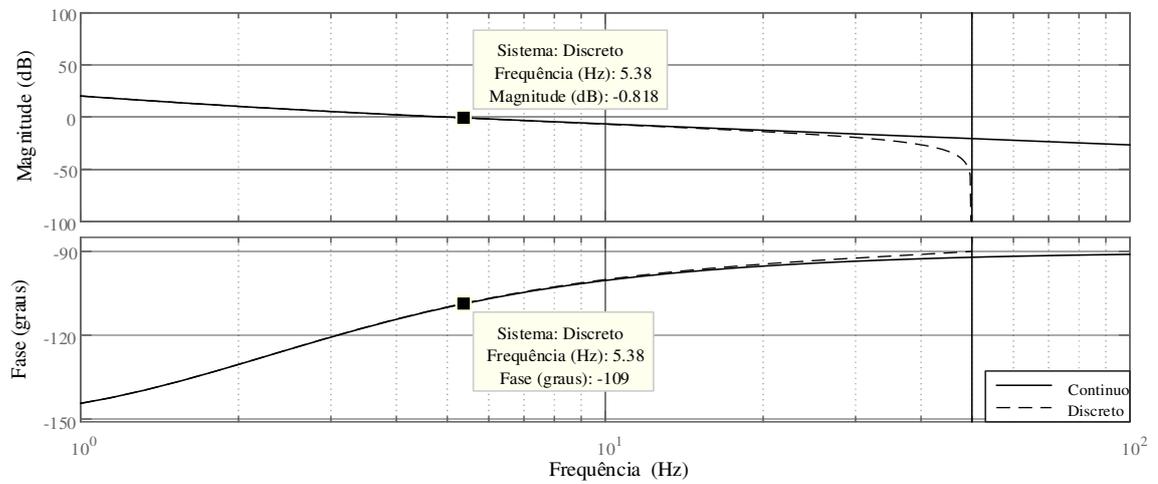
$$\begin{aligned} u(k) &= (-12904) \cdot e[k] + w[k] \\ w[k] &= (4122) \cdot u[k-1] + (28646) \cdot w[k-1] \end{aligned} \quad (64)$$

Na Figura 42 é apresentado o diagrama de bode de malha aberta da velocidade de rotação, já considerando o compensador. Observando o ponto onde a magnitude atinge 0dB é possível notar que, após discretizar o controlador, obtém-se uma frequência de cruzamento de aproximadamente 5Hz e uma margem de fase de 71°, valores próximos aos desejados.

Analisando o diagrama de Bode de malha fechada da velocidade de rotação apresentado na Figura 43, é possível notar que a frequência de corte obtida se encontra acima da frequência de cruzamento, em aproximadamente 6Hz (-3dB).

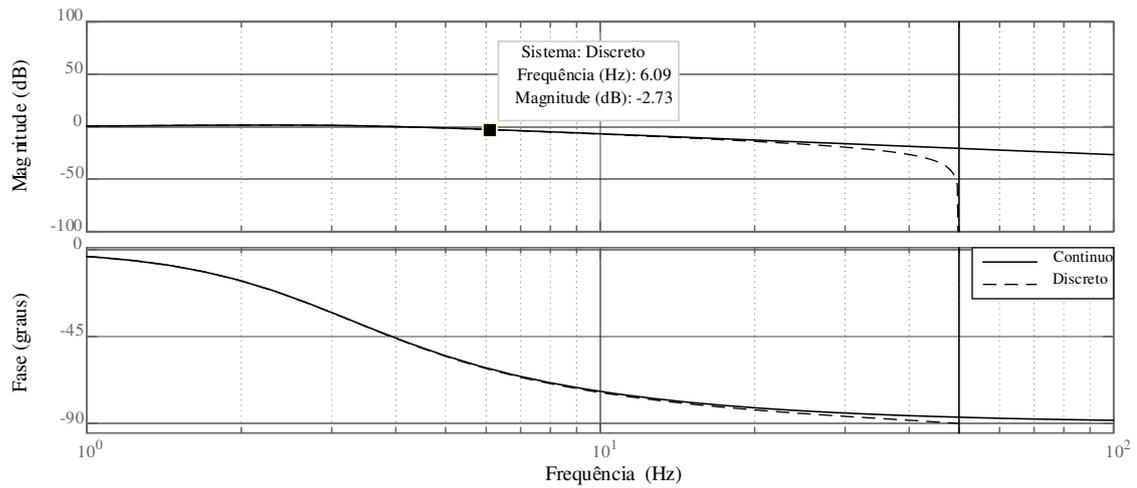
Já a Figura 44 permite observar o comportamento da corrente no indutor do conversor Boost, quando um degrau unitário é aplicado como referência. Observa-se que o tempo de acomodação fica em aproximadamente 0,3 s.

Figura 42 - Diagrama de Bode de malha de aberta compensada da velocidade de rotação



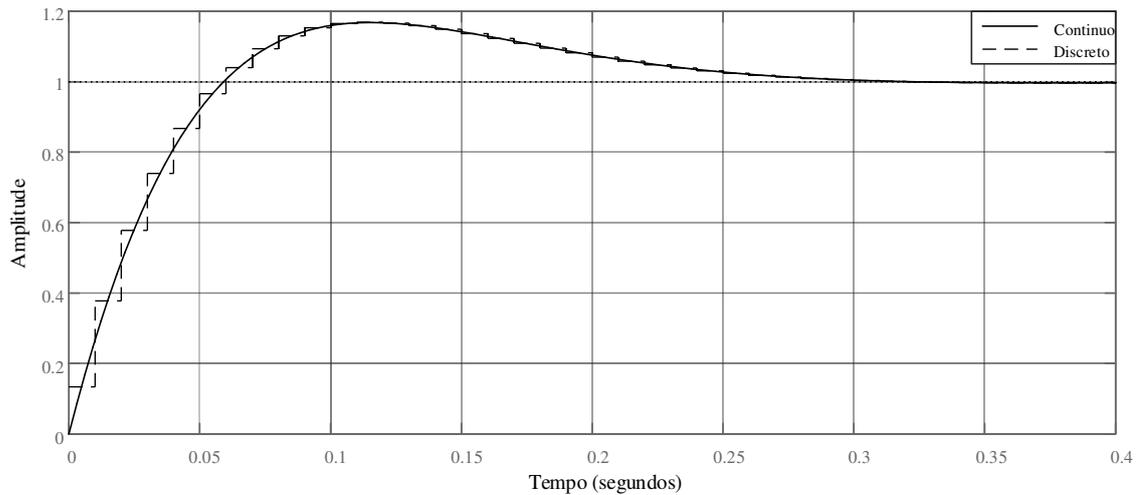
Fonte: desenvolvido pelo autor.

Figura 43 – Diagrama de Bode de malha fechada da velocidade de rotação do eixo



Fonte: desenvolvido pelo autor.

Figura 44 – Resposta ao degrau unitário da malha de velocidade de rotação no eixo



Fonte: desenvolvido pelo autor.

A referência de rotação vista na Figura 34 é obtida através da comunicação serial RS-232 do microcontrolador, e é enviada pelo algoritmo MPPT implementado em Matlab, em tempo real.

3.5 Sistema de controle via Matlab

Uma vez que o sistema de rastreamento de máxima potência proposto se utiliza de redes neurais artificiais, o esforço computacional necessário para que um microcontrolador realizar as operações dos neurônios é elevado. Desta forma, optou-se pela utilização de um sistema computacional para a realização de tais cálculos. Os dados lidos são também registrados, haja vista a necessidade de observar o comportamento do MPPT desenvolvido perante os perfis de vento fornecidos.

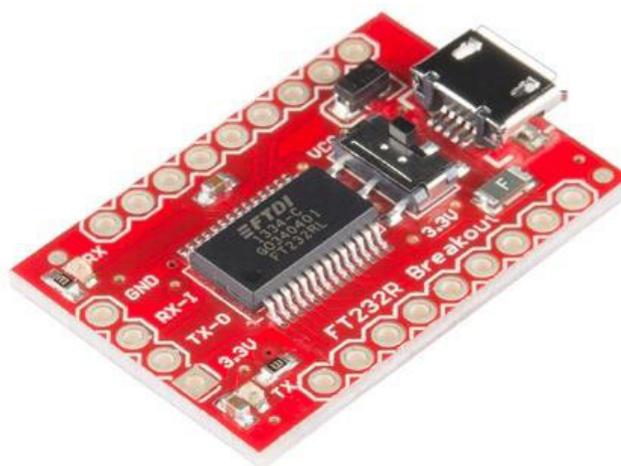
Diversos softwares são utilizados no desenvolvimento de redes neurais, contudo, optou-se pela utilização do software Matlab, tanto pela maior familiaridade do autor com a ferramenta, quanto por sua capacidade de desenvolvimento de códigos genéricos, permitindo modificações internas às redes neurais implementadas.

Os códigos utilizados na implementação da comunicação e interface através do *software* Matlab são divididos em três: *controle.m*, *communicate.m* e *plotter.m*. o primeiro é o código principal, responsável por chamar os demais, o segundo realiza o protocolo de comunicação e o terceiro realiza a plotagem dos dados para que seja possível acompanhar em tempo real o sistema. Os três códigos são apresentados no Apêndice F.

3.5.1 Sistema de aquisição e controle

A comunicação entre microcontrolador e computador é estabelecida através de um sistema de interface entre o protocolo USB e o protocolo RS-232. Para tal, utiliza-se um módulo com o circuito integrado FT232RL. A Figura 45 apresenta a foto de um módulo similar ao utilizado.

Figura 45 – Módulo de interface USB baseado em FT232RL

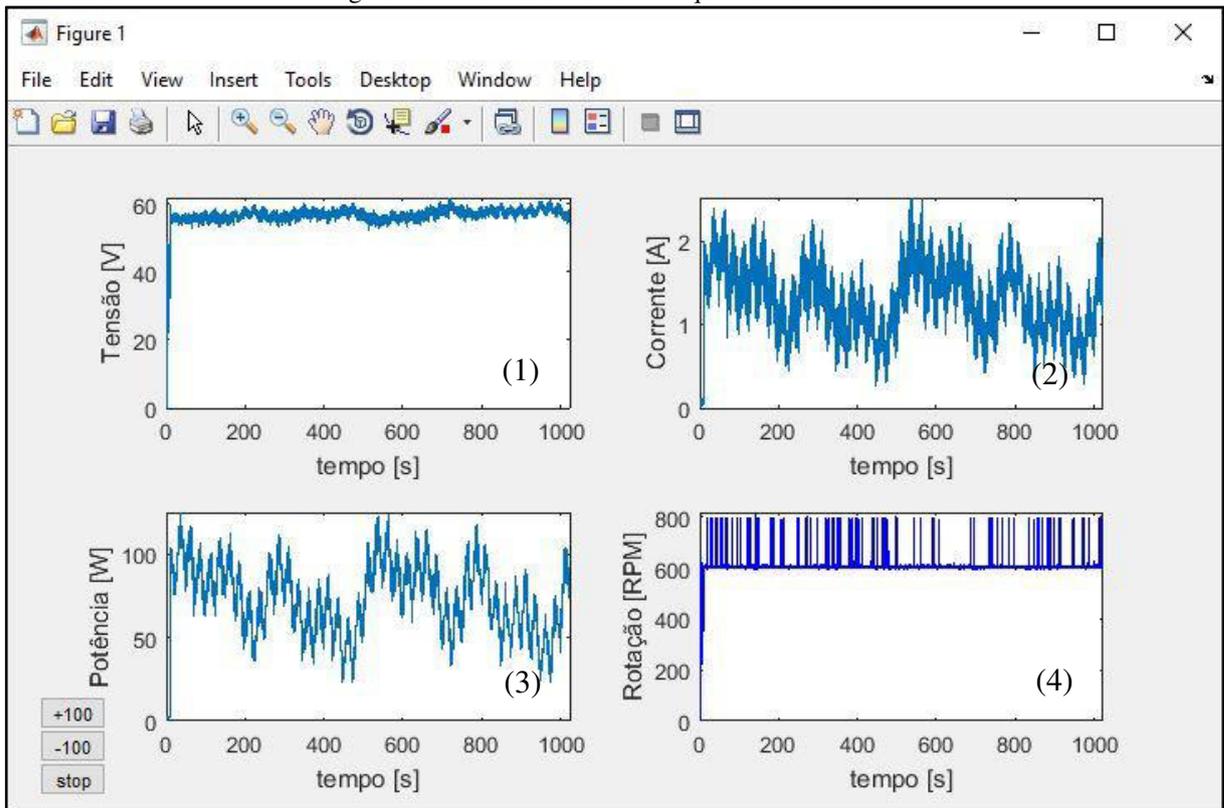


Fonte: site do revendedor *Sparkfun*

O código interno do microcontrolador define que o protocolo de envio e recepção de dados junto ao computador ocorra a cada 10ms. Utiliza uma taxa de comunicação de bits (*BaudRate*) de 250.000, sem *bit* de paridade, com *bit* de parada unitário e utilizando o protocolo de dados de 8 *bits*. O registrador de armazenamento é programado para apresentar apenas 1 posição, descartando valores antigos, quando um novo dado é recebido.

Buscando simplificar a observação dos dados coletados, e o comportamento dos algoritmos MPPT propostos, utiliza-se uma interface simples, através da plotagem de alguns dados. A Figura 46 mostra a interface implementada, nela, são apresentados a tensão (1), a corrente (2) e a potência (3) de entrada do controlador de carga, bem como a velocidade de rotação do eixo do sistema gerador (4) em rotações por minuto (rpm). São implementados ainda três botões que permitem variar a referência de rotação manualmente com passos de 100 rpm, além de um botão que realiza a frenagem do sistema (stop).

Figura 46 – Interface Homem Máquina desenvolvida



Fonte: desenvolvido pelo autor.

3.5.2 Metodologia de implementação do sistema de MPPT.

Os algoritmos de MPPT utilizados foram desenvolvidos em ambiente Matlab na forma de funções. Foram utilizados como parâmetros os dados previamente lidos e transmitidos do microcontrolador, sendo a referência de velocidade definida pelo algoritmo, enviada de volta ao microcontrolador.

No código de controle é possível definir o tempo de espera entre cada iteração, imprescindível para que o sistema chegue a um regime, permitindo uma correta avaliação por parte do MPPT. Os códigos foram feitos em linguagem Matlab, estruturados na forma de função, o que torna necessária a utilização de variáveis do tipo *persistent*, que não são apagadas entre chamadas de funções.

Os algoritmos desenvolvidos são apresentados no capítulo 4 e os códigos dos mesmos são fornecidos no Apêndice G.

3.6 Comentários parciais

O desenvolvimento de uma estrutura para realização de ensaios experimentais é de singular importância para a comprovação da eficácia dos algoritmos propostos. Através da bancada de testes, torna-se possível a implementação prática do sistema MPPT proposto, levando em consideração problemáticas reais relacionadas a ruídos de medições, utilização de simplificações na modelagem dos componentes, dentre outras características que possam vir a influenciar o funcionamento de um algoritmo MPPT.

4 ALGORITMOS MPPT PROPOSTOS

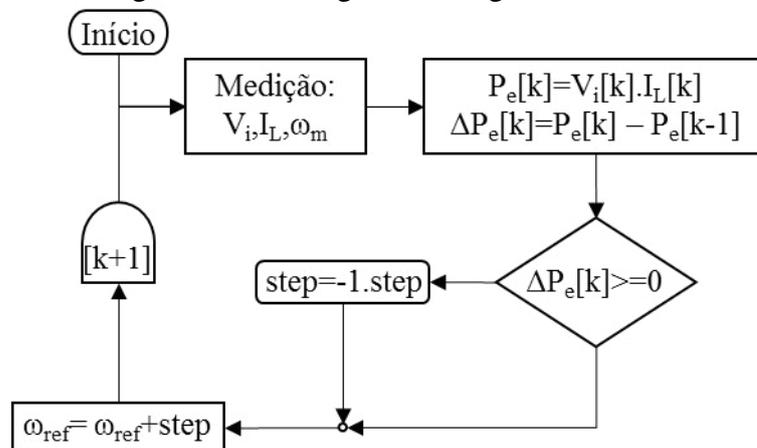
Buscando uma proposta de algoritmo de rastreamento de máxima potência, que apresente a vantagem de não utilizar sensores de velocidade de vento como um P&O, mas que possua capacidade de aprendizagem, de forma a melhorar a resposta obtida e reduzir a complexidade na escolha do tamanho da perturbação.

São apresentados neste capítulo, dois modelos de algoritmos baseados em redes neurais: o P&O com passo adaptativo neural (*Neural Adaptive Step Size Perturb and Observe*) definido nesse trabalho como NASPO (WATTES et al., 2015) e o MPPT baseado no *Continuous Actor Critic Learning Automaton* denominado neste trabalho como CACLA (WATTES et al., 2016).

4.1 Algoritmo Perturba e Observa de referência

No ambiente de simulação, tem-se o conhecimento de todos os parâmetros do sistema de geração eólico e, portanto, é possível realizar um comparativo do valor de potência extraída através da utilização dos algoritmos propostos com o valor máximo de potência extraível pela turbina. Contudo, nos ensaios experimentais existem parâmetros não ideais que impossibilitam definir com precisão o valor máximo de potência extraível. Dessa forma, utiliza-se o algoritmo Perturba e Observa (P&O) como parâmetro de comparação ante os algoritmos propostos. A Figura 47 apresenta um fluxograma explicativo do funcionamento do algoritmo P&O. A implementação realizada foi simplificada através da escolha do valor de passo de forma fixa. Os parâmetros do MPPT P&O implementado são apresentados na Tabela 8.

Figura 47 – Fluxograma do algoritmo P&O



Fonte: desenvolvido pelo autor.

Tabela 8 – Parametros do MPPT P&O

Parâmetro	Descrição	Valor
t_{ite}	Tempo entre iterações	1 s
Step	Valor do passo de perturbação	10 rpm
$\omega_{m_ref_min}$	Valor mínimo da referência de rotação	200 rpm
$\omega_{m_ref_max}$	Valor máximo da referência de rotação	1000 rpm

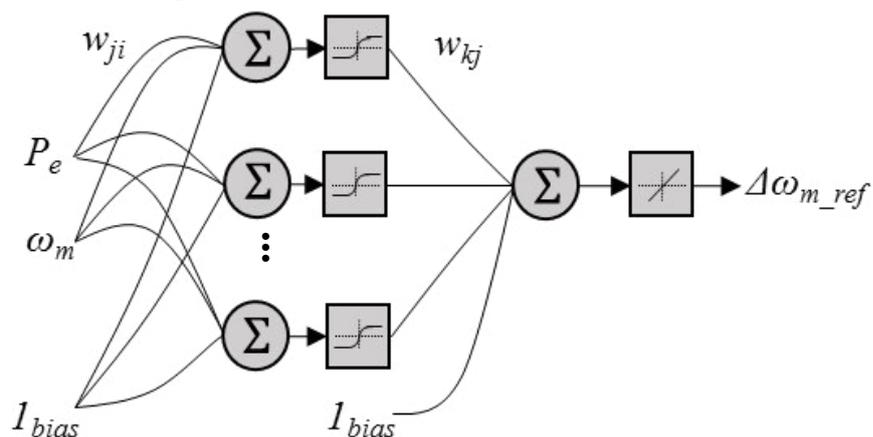
Fonte: Desenvolvido pelo autor.

4.2 Algoritmo MPPT NASPO

Partindo do algoritmo básico de P&O, propõe-se um tamanho de passo de perturbação variável, que permita o algoritmo manter uma elevada dinâmica, mas que ainda assim possua um erro de regime suficientemente pequeno. Busca-se também, dotar o algoritmo MPPT de inteligência, de forma que ao operar novamente em determinado estado, ele o faça com uma melhor dinâmica.

Dessa forma, a perturbação aplicada pelo algoritmo é obtida através de uma rede neural artificial, cujos pesos são ajustados de acordo com os resultados obtidos a partir das perturbações anteriormente aplicadas. O algoritmo desenvolvido segue o mesmo modelo apresentado em (WATTES et al., 2015).

Figura 48 – Rede neural utilizada no NASPO



Fonte: desenvolvido pelo autor.

A Figura 48 apresenta a estrutura da rede neural artificial desenvolvida. Nela, $\Delta\omega_{m_ref}$ indica o passo de perturbação a ser aplicado na referência de rotação (ω_{m_ref}). As entradas da rede neural são o valor de potência elétrica (P_e) instantânea e a velocidade de rotação do eixo (ω_m) instantânea.

A rede neural proposta utiliza funções de ativação do tipo tangente hiperbólica na camada oculta e linear na camada de saída, sendo a camada oculta formada por 10 neurônios do tipo Perceptron.

Tabela 9 – Parametros do MPPT NASPO

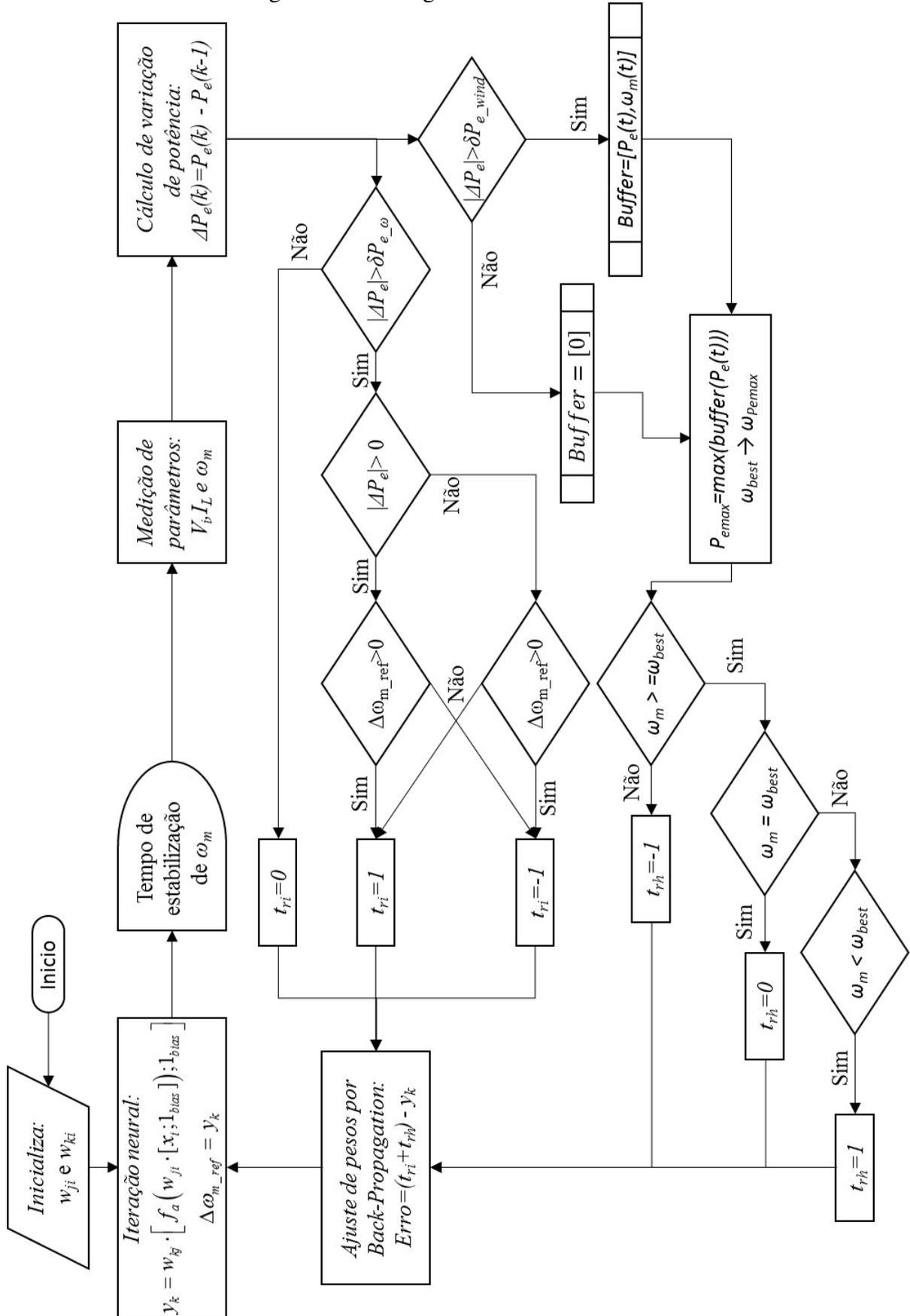
Parâmetro	Descrição	Valor
t_{ite}	Tempo entre iterações	1 s
n	Número de neurônios na camada oculta	15
η	Coefficiente de aprendizagem	0,5
$\delta P_{e_ \omega m}$	Varição mínima de potência considerada melhoria, ante a variação de referência de rotação ωm .	5 W
$\delta P_{e_ wind}$	Varição mínima de potência considerada indicativo de variação do vento.	40 W
Tam_Buffer	Número de elementos do buffer circular.	5
Step	Ganho aplicado no passo de perturbação unitário.	50

Fonte: Desenvolvido pelo autor.

O ajuste dos pesos sinápticos da rede neural são feitos baseados na variação de potência elétrica de acordo com o que é mostrado no fluxograma exibido na Figura 49.

Pelo fluxograma verifica-se o seguinte modo de operação do algoritmo: (I) Inicia-se aleatoriamente os valores dos pesos das camadas oculta (w_{ji}) e de saída (w_{kj}) da rede neural e define-se a referência de rotação no valor mínimo; (II) Calcula-se através da rede neural a perturbação a referência de rotação ($\Delta\omega_{m_ref}$) que é então somada ao valor de referência de rotação (ω_{m_ref}); (III) Aguarda-se o tempo de estabilização da velocidade de rotação cuja dinâmica é mais lenta que a dinâmica de potência; (IV) A variação de potência é então analisada, a fim de definir se houve melhoria significativa da mesma e se houve uma variação na velocidade do vento; (IV.a) Se a variação de potência foi baixa ($<\delta P_{e_wind}$), estima-se que não houve variação de velocidade do vento significativa e mantêm-se o estado anteriormente visitado registrado em um buffer circular de cinco posições, caso a variação tenha sido elevada, considera-se mudança na velocidade do vento e portanto, limpa-se as informações anteriores gravadas no buffer circular. Os valores registrados no buffer são então utilizados no cálculo de t_{rh} baseado no conhecimento prévio de um ponto de potência superior ou não; (IV.b) Paralelamente, observa-se se há uma variação de potência superior a um valor mínimo ($\delta P_{e_ \omega}$), o que evita ajustes incorreto devido a ruídos. De acordo com a variação de potência calcula-se o valor de t_{ri} que indica aumento ou redução da potência; (V) A variável Erro é então calculada com base nos valores t_{rh} , t_{ri} , e na saída anterior da rede neural; (IV) A partir do valor do erro são ajustados os pesos da rede neural levando-se em conta o coeficiente de aprendizagem e o algoritmo retorna ao passo (II).

Figura 49 – Fluxograma do MPPT NASPO



Fonte: desenvolvido pelo autor.

O termo t_{ri} tem o objetivo de indicar a rede neural se o ultimo passo dado teve resultado satisfatório ou não, contudo devido a variação de vento, adiciona-se o termo t_{rh} , com o objetivo de reduzir o número de avaliações erroneas. A forma de cálculo da variável *Erro* é desenvolvida com o objetivo de permitir a utilização do algoritmo de *Back-Propagation* para o ajuste dos pesos, uma vez que seria necessário um valor de alvo, do qual não se tem conhecimento.

4.3 Algoritmo MPPT CACLA

O algoritmo CACLA foi proposto por Hasselt e Wiering (2007) baseado nos conceitos de TD e sistemas *actor-critics*, mas levando em consideração a premissa de que no mundo real, poucas variáveis encontram-se de forma discreta, de forma que o algoritmo tenha capacidade de operar num espaço de estados contínuo.

Sistemas *actor-critic* são apresentados por Sutton e Barto (2012). Tais algoritmos consistem de dois sistemas, denominados atuador (*actor*) e crítico (*critic*). O atuador, é presente na maioria dos algoritmos de aprendizagem por reforço, ele é o responsável pela decisão de qual ação deve ser tomada, já o crítico é responsável por analisar a recompensa obtida, informando ao atuador se a ação tomada foi de fato boa ou não.

O algoritmo CACLA, tem como objetivo realizar o controle *actor-critic* aplicado a sistemas contínuos, para tal, faz-se uso de redes neurais MLP, uma para o crítico e outra para o atuador. O trabalho realizado é apresentado também em Watters (2016).

Para utilizar o algoritmo CACLA na aplicação de MPPT (WATTERS et al., 2016) alguns parâmetros devem ser definidos. O estado s_t é definido pelo conjunto velocidade de rotação e potência elétrica $[\omega_m, P_e]$. A ação obtida pelo atuador é a referência de rotação $[\omega_{m_ref}]$, enquanto a saída do crítico define a avaliação do estado s_t $[V(s_t)]$. A recompensa obtida é calculada de acordo com (65). Salienta-se a assimetria da recompensa feita com o objetivo de reduzir oscilação em torno do ponto de máximo.

$$\begin{cases} |P_e(t) - P_e(t-1)| < dP_{e_min} \rightarrow r = 0 \\ P_e(t) - P_e(t-1) > dP_{e_min} \rightarrow r = 0.5 \\ P_e(t) - P_e(t-1) < -dP_{e_min} \rightarrow r = -1 \end{cases} \quad (65)$$

Tabela 10 – Parametros do MPPT CACLA

Parâmetro	Descrição	Valor
t_{ite}	Tempo entre iterações	1 s
n_{crit}	Número de neurônios na camada oculta do crítico	15
n_{act}	Número de neurônios na camada oculta do atuador	15
α	Coefficiente de aprendizagem do atuador	0,01
β	Coefficiente de aprendizagem do crítico	0,95
γ	Coefficiente de amortização das avaliações futuras do crítico	0,9
ε	Coefficiente de seleção de ação aleatória	0,3
σ	Coefficiente de ajuste da ação aleatória	0,025
δP_{e_min}	Varição mínima de potência considerada melhoria, ante a variação de referência de rotação ω_m .	5 W

Fonte: Desenvolvido pelo autor.

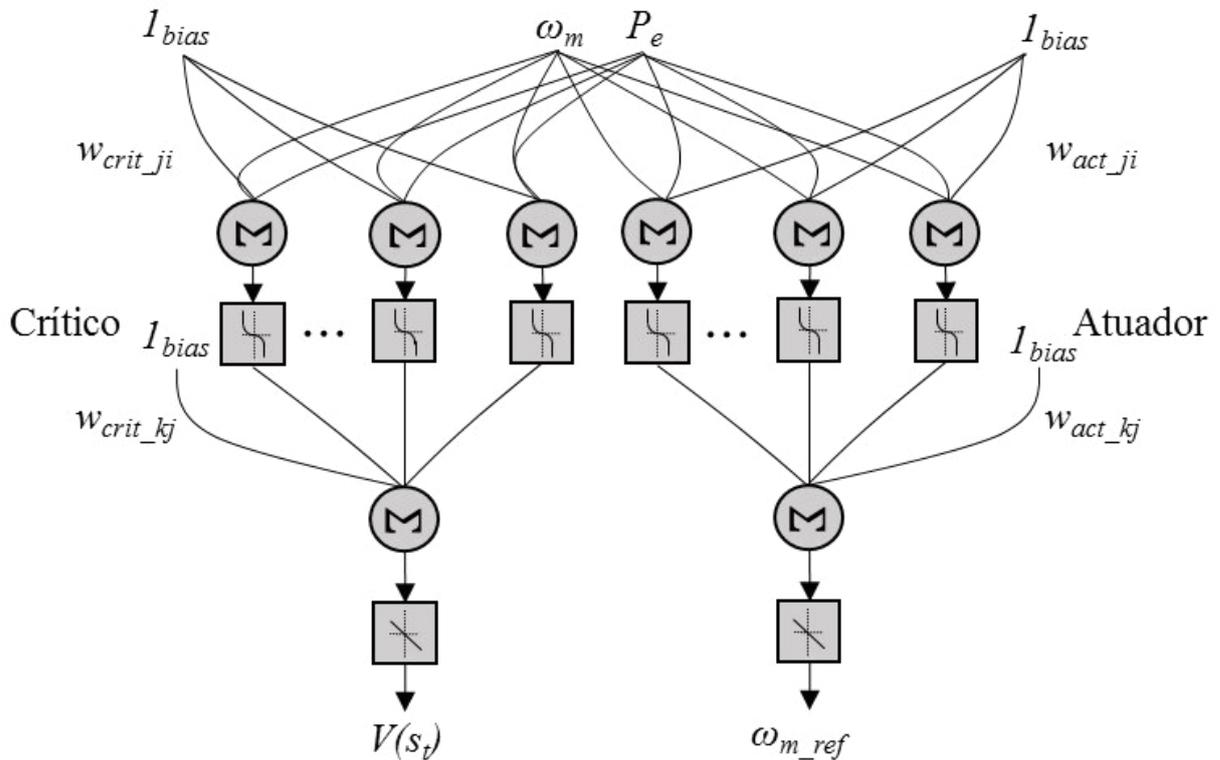
Definindo os parâmetros da CACLA é possível estabelecer um diagrama das redes neurais desenvolvidas, tal diagrama é apresentado na Figura 50. A arquitetura utilizada é a *Multi-Layer Perceptron* (MLP), com funções de ativação sigmoide nos neurônios das camadas ocultas e linear nas camadas de saída, para ambos, Crítico e Atuador. O número de neurônios utilizados foi definido através de simulações preliminares, sendo estabelecido para ambos os sistemas 15 neurônios ocultos.

Há ainda os parâmetros de aprendizagem do Atuador (α) e do Crítico (β), constantes que ponderam o ajuste dos pesos destas redes, os valores estabelecidos para α e β foram 0,01 e 0,95, respectivamente. Outro parâmetro importante para algoritmos de aprendizagem por reforço é a ponderação de avaliações futuras (γ). Essa variável é responsável pelo decaimento do cumulado histórico de avaliações, adotando-se um valor de 0,65.

Uma vez que se faz necessário explorar as possibilidades de ações tomadas, faz-se imperativo estabelecer os parâmetros de aleatoriedade. O primeiro é referente ao percentual de vezes que uma ação aleatória é tomada (ε), para o qual adota-se o valor de 5%. O segundo parâmetro é relacionado ao valor de ação aleatório (σ), adota-se para tal, um valor aleatório que varia de 10% da ação esperada, para mais ou para menos.

O algoritmo MPPT baseado no CACLA é desenvolvido em ambiente Matlab através de codificação sequencial, o código utilizado para isso é apresentado no Apêndice G: Códigos de MPPT MATLAB.

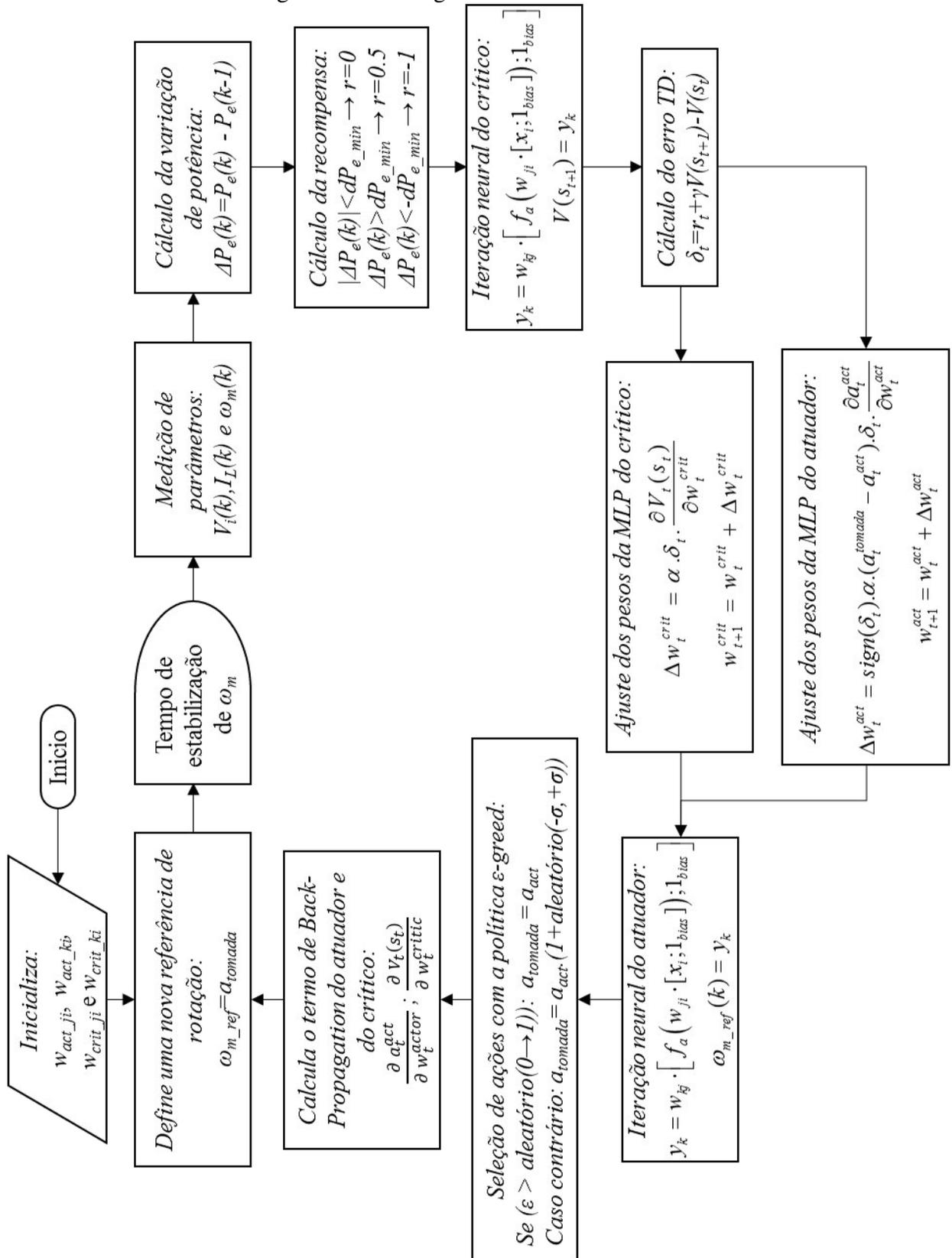
Figura 50 – Rede neural utilizada no MPPT CACLA



Fonte: desenvolvido pelo autor.

A fim de esclarecer a forma de implementação, um fluxograma completo dos processos internos do algoritmo MPPT são apresentados na Figura 51. Nele observam-se os seguintes passos: (I) a inicialização dos pesos das camadas ocultas e de entrada do MLP Crítico e do MLP Atuador, seguido pelo (II) estabelecimento de uma referência de velocidade de rotação e posterior espera do tempo necessário para estabilização da malha de velocidade de rotação; (III) após a estabilização do sistema, medem-se as variáveis tensão de entrada (V_i) e corrente do indutor (I_L) do conversor boost e a velocidade de rotação do eixo (ω_m); (IV) Calcula-se então a variação de potência (ΔP_e) a partir da qual define-se a recompensa (r) obtida; (V) Realiza-se então a avaliação do novo ponto de operação ($V_{(s+1)}$) através do crítico, calculando em seguida o erro de diferença temporal (δ_t); (VI) Utilizando δ_t realiza-se o ajuste do pesos do Crítico e do Atuador, com base no gradiente anterior; (VII) A iteração do Atuador define então o valor de referência de rotação; (VIII) Realiza-se decisão de ação aleatório ou não; (IX) calcula-se o gradiente do Atuador e do Crítico, retornando então ao passo (II).

Figura 51 – Fluxograma do MPPT CACLA



Fonte: desenvolvido pelo autor.

4.4 Comentários parciais

Embora os algoritmos propostos apresentem-se como uma metodologia diferenciada para a solução da problemática de rastreamento de máxima potência, ambos se fundamentam em algoritmos já utilizados com essa finalidade (P&O), ou que já tenham sido utilizados de forma eficaz em outras aplicações (CACLA).

A fim de observar a eficácia em solucionar a problemática, realizam-se simulações computacionais através dos softwares PSIM e Matlab e ensaios experimentais através da bancada emuladora baseada em MCC, projetada e montada com essa finalidade. O capítulo 5 apresenta esses resultados.

5 RESULTADOS

Visando observar o funcionamento dos algoritmos MPPT propostos, experimentos práticos são necessários, contudo, análises computacionais prévias são essenciais.

As simulações computacionais desenvolvidas foram feitas de forma independente. Devido às constantes de tempo da corrente dos conversores ser muito menor que a constante de tempo das máquinas, as simulações de circuitos realizadas em ambiente PSIM são divididas em malhas de correntes e malhas de rotação. Além disso, as simulações de validação do algoritmo MPPT proposto são realizadas independentemente em ambiente Matlab. Nas seguintes seções serão apresentados os resultados dessas simulações.

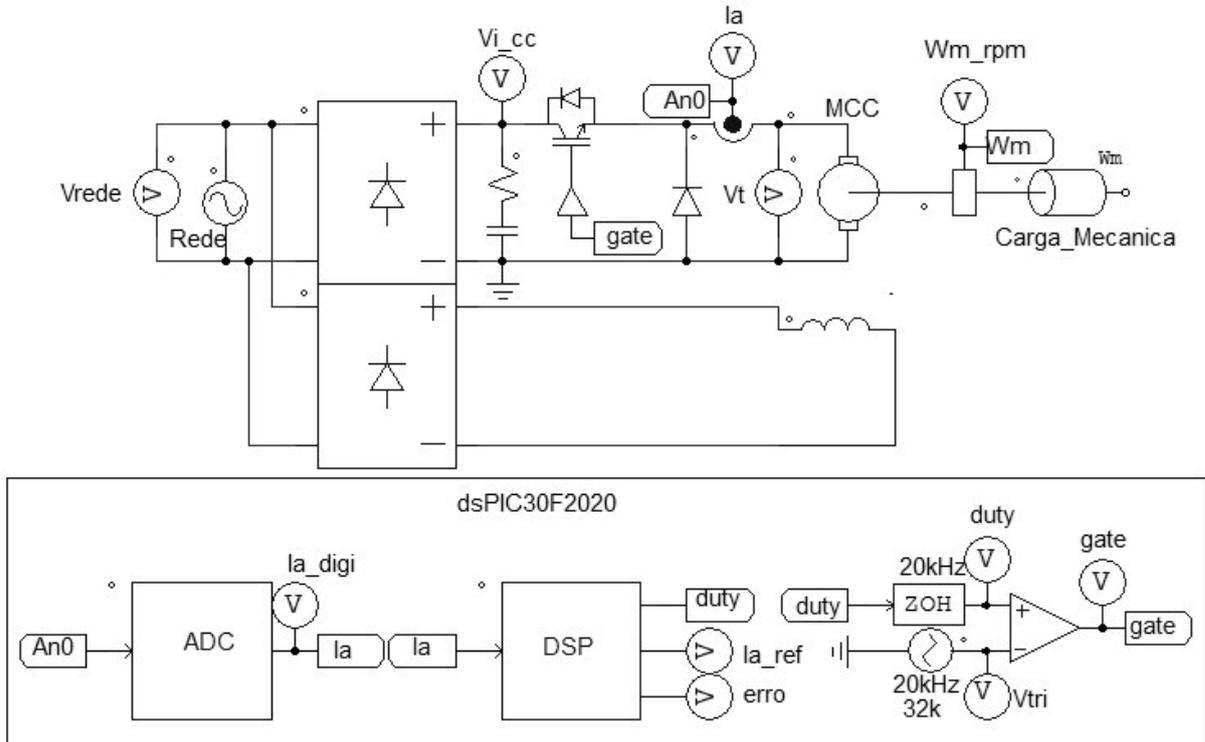
5.1 Simulações do emulador de turbina eólica

Uma vez que o sistema emulador de turbina eólica busca reproduzir o torque mecânico imposto por um determinado valor de velocidade de vento à uma turbina fictícia, faz-se necessária a implementação de uma malha de torque, que em motores de corrente contínua com excitação independente, corresponde à uma malha de corrente, onde é levada em conta a constante K_T que relaciona ambos.

5.1.1 Simulações da malha de corrente de armadura do MCC

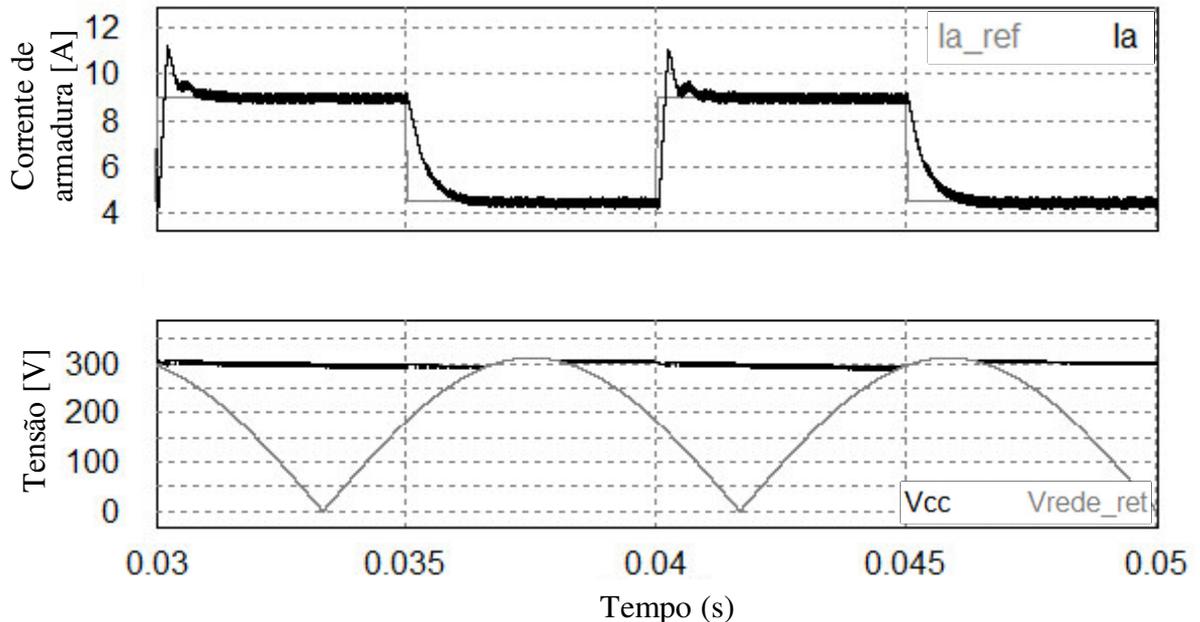
Busca-se realizar ensaios computacionais o mais próximo possível das condições reais do experimento. Dessa forma, a simulação desenvolvida faz uso de blocos que simulam um motor de corrente contínua com campo independente, semicondutores eletrônicos, bem como alguns blocos computacionais que reproduzem o mais fielmente possível, a programação implementada nos microcontroladores do sistema real. A Figura 52 apresenta o diagrama do sistema simulador.

Figura 52 – Diagrama de simulação da malha de corrente de armadura do MCC



Fonte: Esquemático desenvolvido no software PSIM.

Figura 53 – Gráfico da corrente de armadura do MCC com degraus na referência

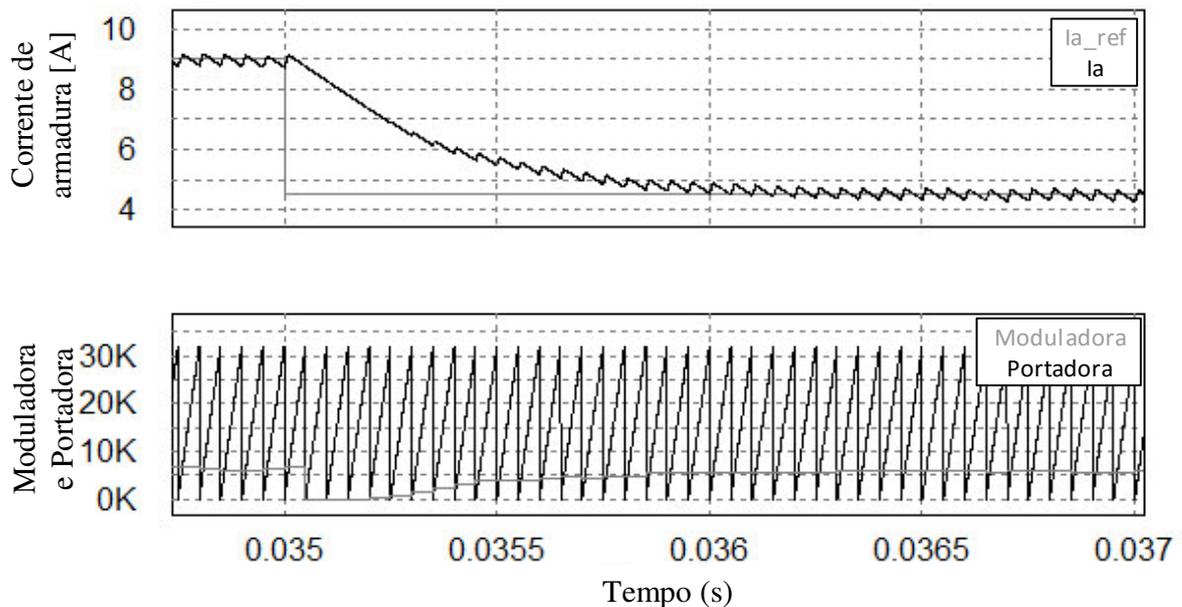


Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

Através dessa simulação é possível observar o comportamento da corrente (I_a), quando submetida a degraus de variação de referência (I_{a_ref}). A Figura 53 apresenta a corrente de armadura (I_a) juntamente com sua referência, além da tensão no barramento CC (V_{cc}) e um sinal referente ao sinal da rede retificado (V_{rede_ret}).

A Figura 54 apresenta com maior precisão a oscilação de chaveamento da corrente de armadura (I_a) do MCC, apresentando como referência os sinais de portadora (V_{tri}) e moduladora ($duty$), utilizadas para formar o sinal pwm de controle. Percebe-se o controle atuando de forma que a corrente do indutor oscile em torno da referência estabelecida, bem como sua rápida resposta, onde observa-se um tempo de acomodação inferior a 1ms.

Figura 54 – Detalhe de chaveamento da corrente de armadura do MCC



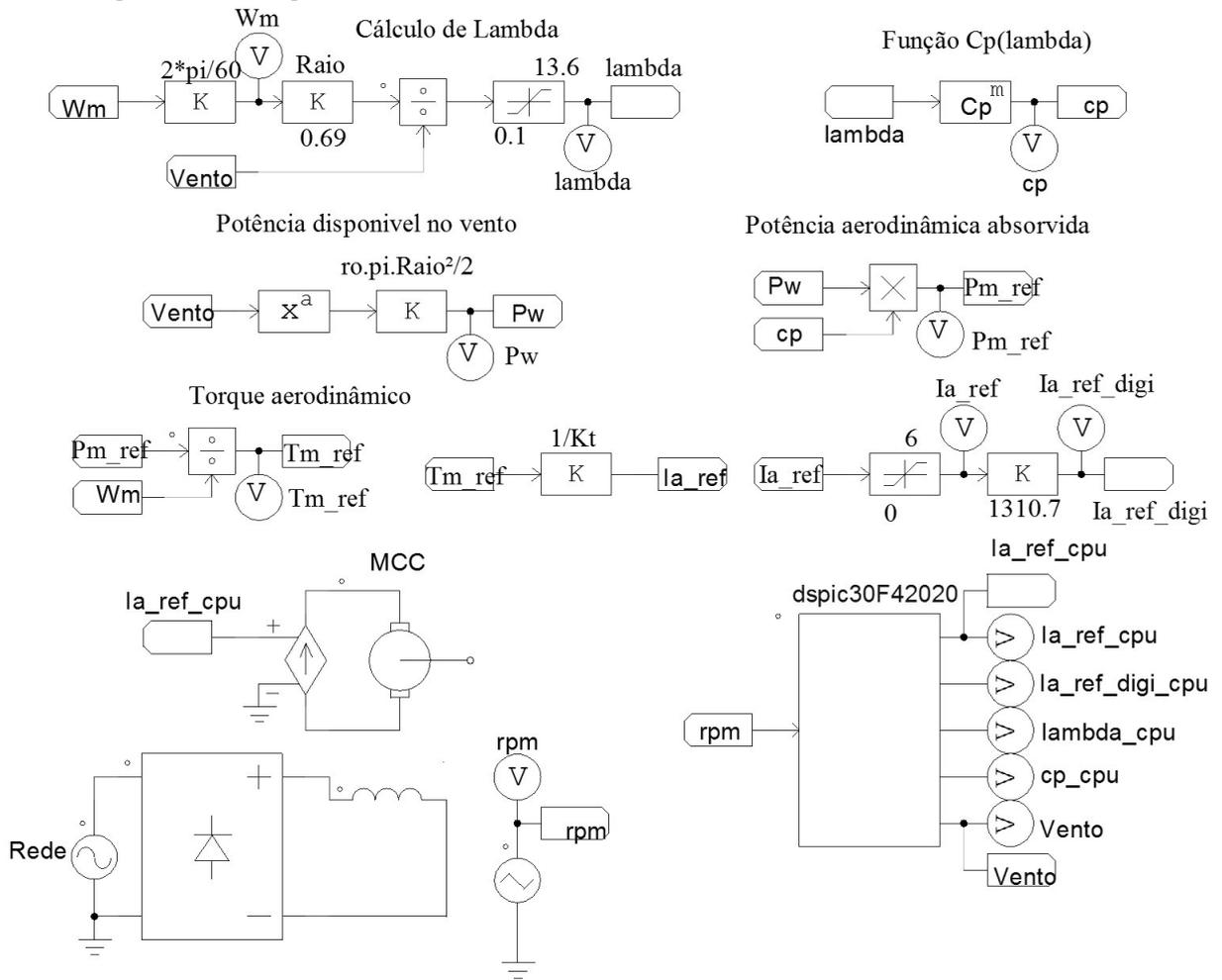
Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

Após estabelecer a funcionalidade da malha de controle de corrente de armadura do MCC, faz-se necessário validar o algoritmo de cálculo da referência dessa corrente.

5.1.2 Simulações do cálculo da referência de corrente do conversor chopper

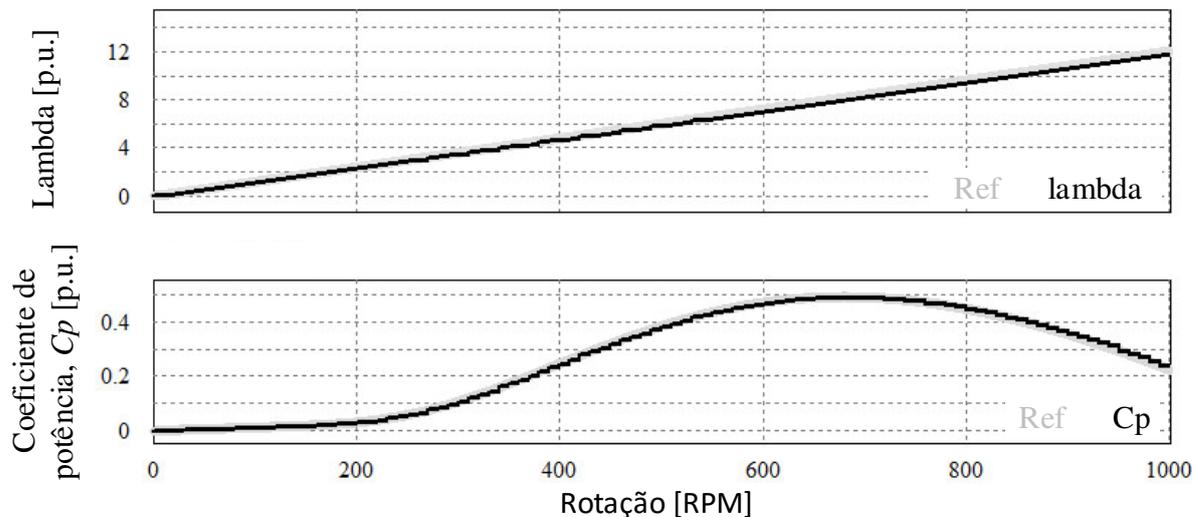
Como descrito no capítulo 3, a referência de corrente no enrolamento de armadura do MCC é obtida através de um equacionamento aerodinâmico. A fim de validar o código desenvolvido com essa finalidade, uma simulação para fins de comparação foi realizada. Nas simulações desenvolvidas, optou-se por utilizar um valor constante de vento de 6 m/s. A Figura 55 apresenta os diagramas na forma de blocos e o bloco de código C com o referido código.

Figura 55 – Diagrama do cálculo da referência de corrente de armadura do MCC



Fonte: Esquemático desenvolvido no software PSIM.

Figura 56 – Gráfico comparativo dos valores de c_p e Λ , teóricos e calculados

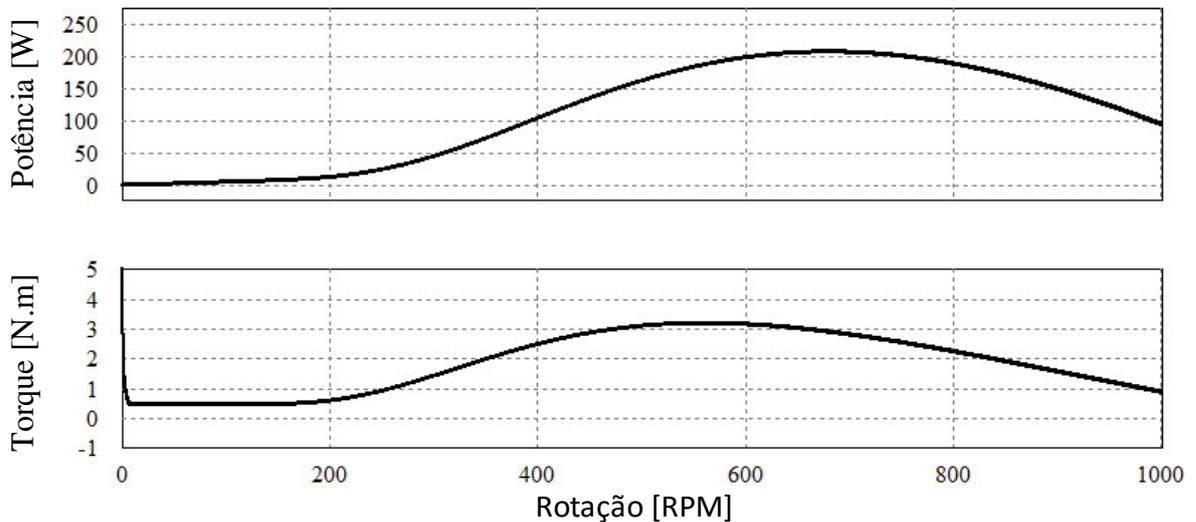


Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

Como é possível visualizar na Figura 56, os valores calculados de λ e c_p , através do diagrama de blocos, coaduna com o valor obtido através do bloco de código C (c_{pu}). Vale ressaltar, que apenas ao final do código em C, um ganho em ponto flutuante é aplicado nos valores de λ e c_p , para fins de comparação, sendo todo o cálculo dos mesmos realizado com números inteiros.

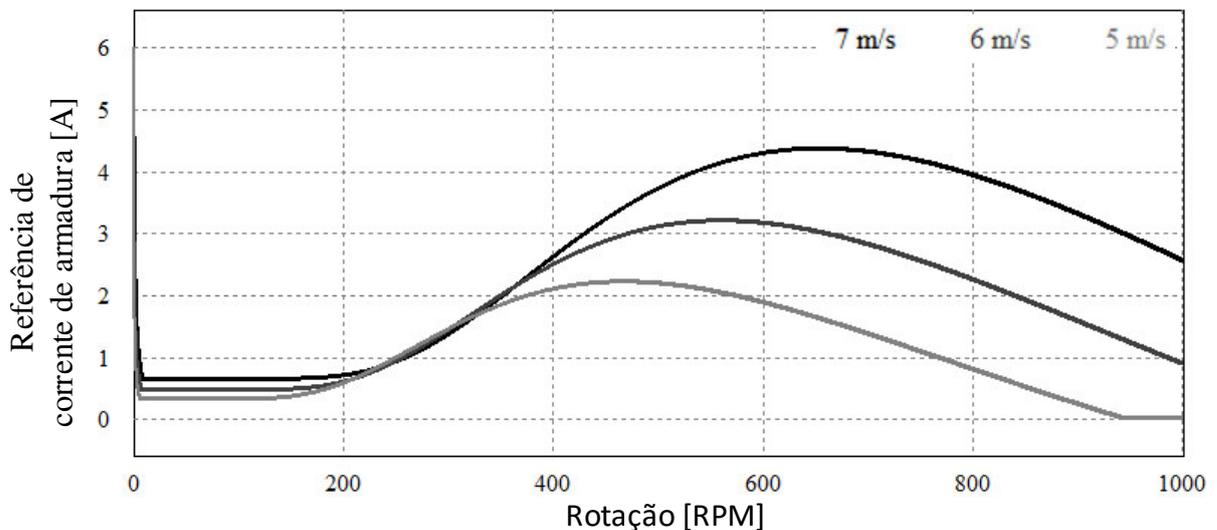
Na Figura 57 os valores de potência mecânica no eixo do MCC (P_{m_ref}) e o torque mecânico no mesmo (T_{m_ref}) são apresentados em função da velocidade de rotação da máquina, tendo como base uma velocidade de vento de 6 m/s.

Figura 57 – Gráficos de potência e torque da simulação de referência de corrente do MCC



Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

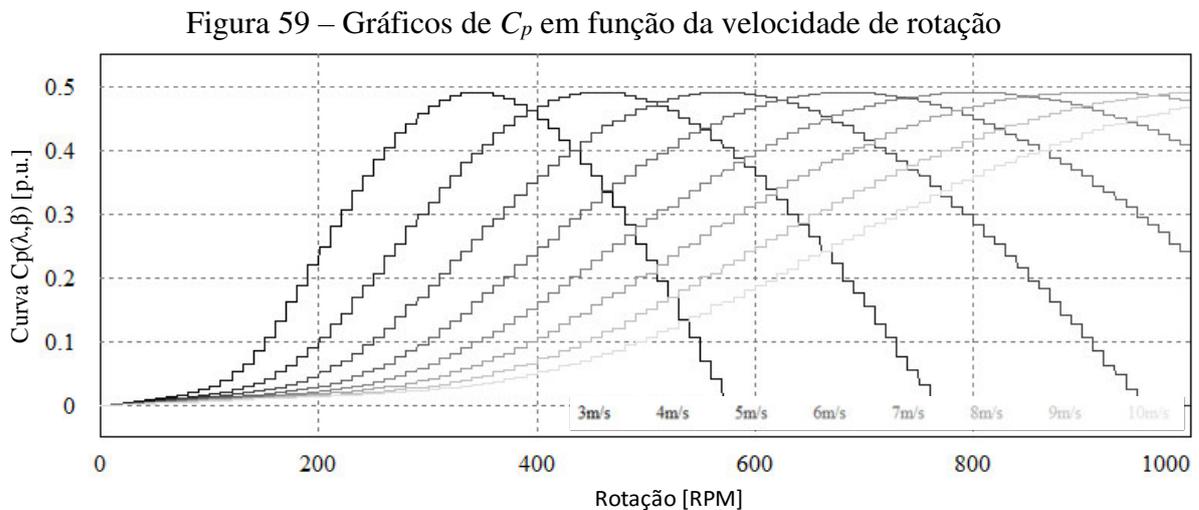
Figura 58 – Gráfico do valor de corrente de armadura de referência do MCC em função da rotação



Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

Observa-se ainda o valor de corrente obtida em função da velocidade de rotação do motor. A Figura 58 apresenta o gráfico que relaciona o valor de corrente de referência em função da velocidade de rotação do eixo em RPM, considerando velocidades de vento de 5 m/s, 6 m/s e 7 m/s.

Com o objetivo de ilustrar a variação do ponto de máxima potência, a Figura 59 apresenta a função C_p relacionada com a velocidade de rotação em rpm para diferentes valores de velocidade do vento.



Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

Tem-se dessa forma uma comprovação computacional da correta forma de operação do sistema emulador, passando-se então para a avaliação do sistema controlador de carga.

5.2 Simulações do controlador de carga do gerador

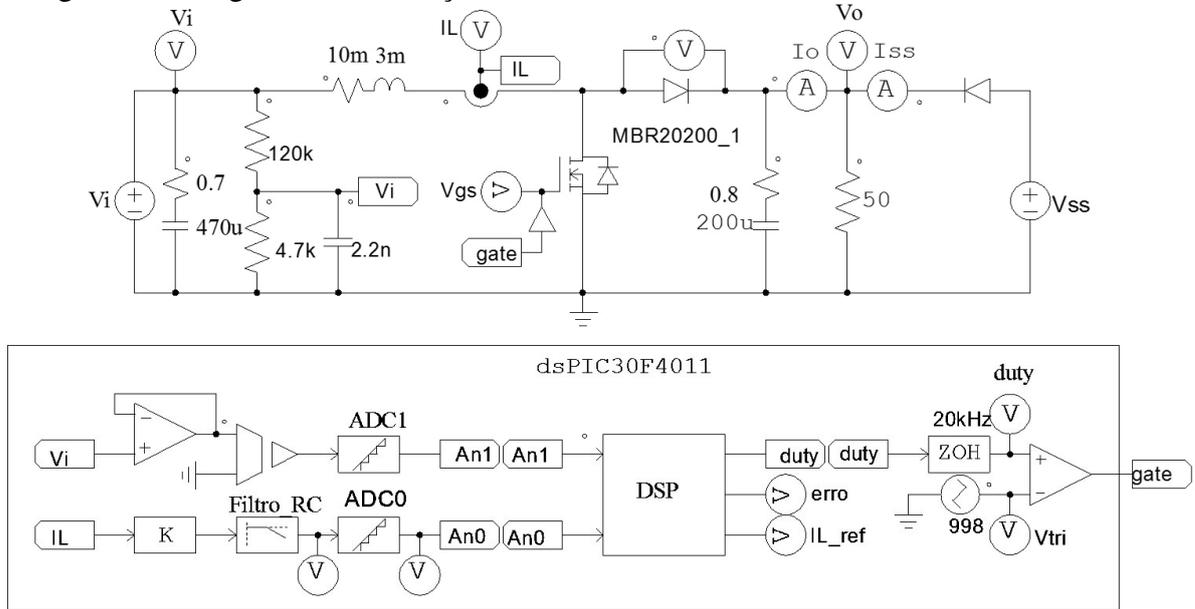
O conversor Boost, responsável por ajustar a carga do gerador, o faz por meio do controle de corrente do indutor (I_L), que é feito de forma a estabelecer a velocidade de rotação do eixo ótima, de acordo com o algoritmo MPPT. Dessa forma, simula-se inicialmente a malha de controle de corrente no indutor, seguida da simulação da malha de rotação.

5.2.1 Simulação da malha de corrente do conversor Boost

A Figura 60 apresenta o diagrama do circuito do conversor Boost simulado em PSIM. Para a validação da malha de corrente, dispensa-se o modelo do gerador de imã

permanente, uma vez que a malha de corrente é projetada rápida o suficiente para desprezar variações de velocidade de rotação do gerador.

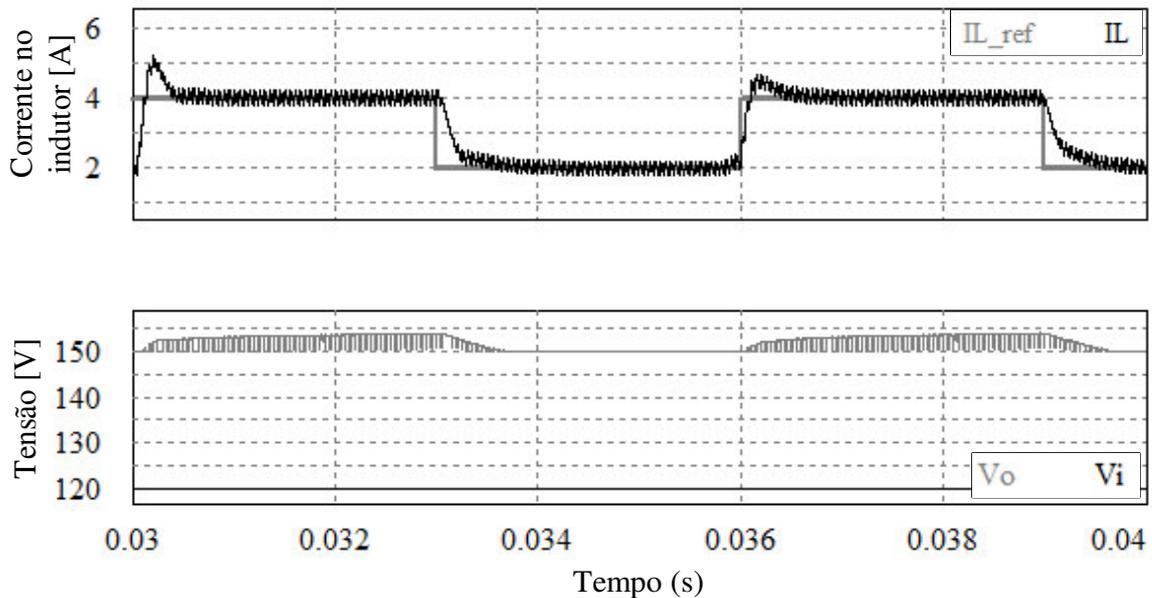
Figura 60 - Diagrama de simulação da malha de corrente do indutor do conversor Boost



Fonte: Esquemático desenvolvido no software PSIM

Opta-se por utilizar uma fonte de corrente contínua de 120 V substituindo a saída do retificador trifásico. Buscando viabilizar o controle da corrente uma fonte de 150 V é conectada em paralelo à carga, visando manter a tensão na mesma estabilizada.

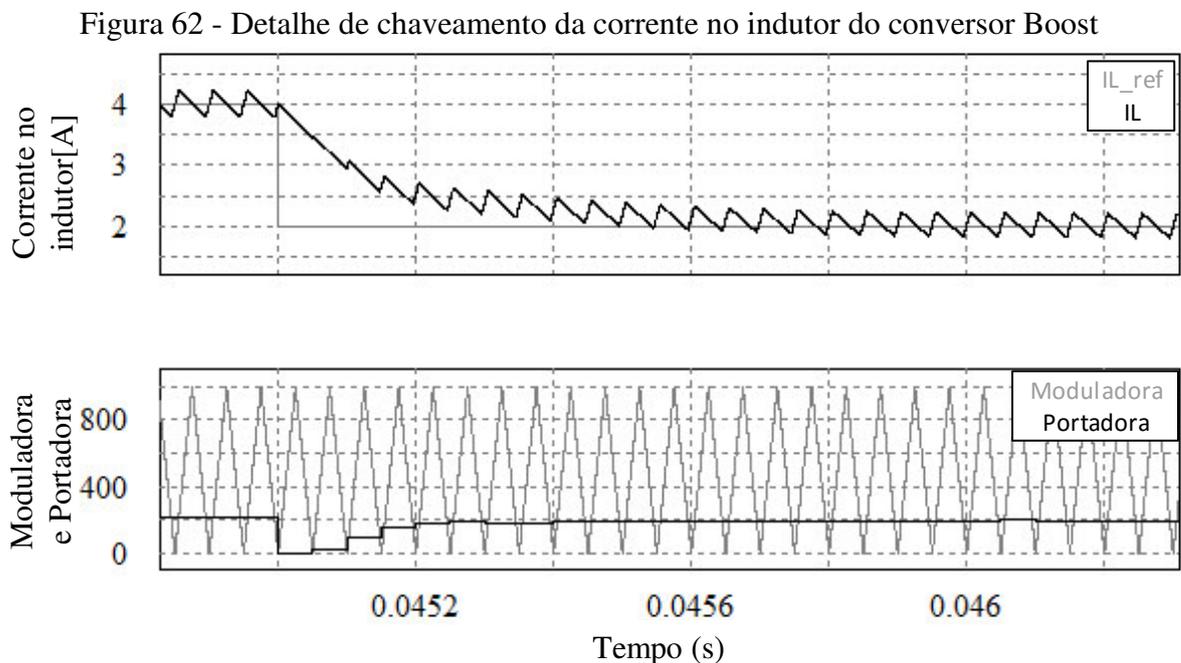
Figura 61 – Gráfico da corrente no indutor e das tensões de entrada e saída do controlador de carga



Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

A Figura 61 apresenta o gráfico da corrente no indutor do conversor Boost junto com a respectiva referência. Além disso são apresentadas as formas de onda da tensão de entrada (V_i) e em cima da carga (V_o). É possível notar a estabilização do controlador, bem como o nível aceitável de sobressinal do mesmo.

Já na Figura 62, é apresentado o detalhamento do comportamento transitório da malha de corrente no indutor do conversor Boost. São apresentados também os sinais de portadora (V_{tri}) e moduladora ($duty$) do sinal *pwm* de controle. Com o detalhe do chaveamento é possível ver a correta operação do controlador, uma vez que a corrente do indutor oscila em torno da referência fornecida, além disso, observa-se um tempo de acomodação inferior a 1ms.



Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

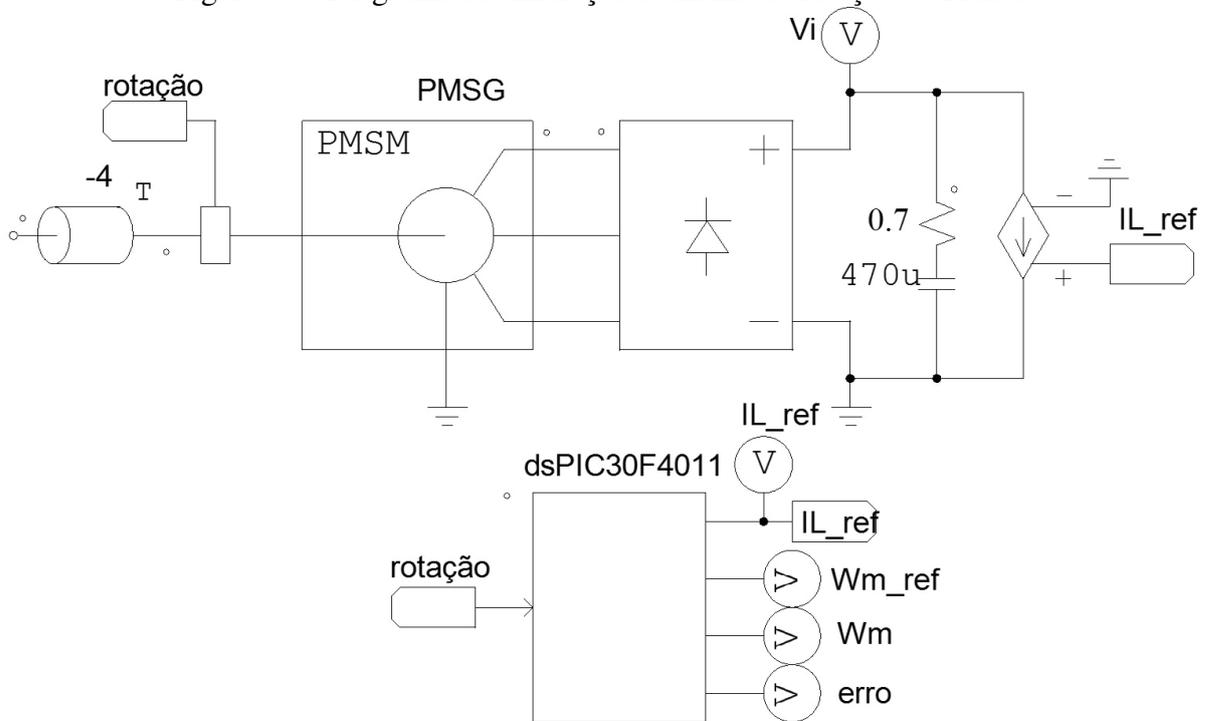
Nota-se que os resultados obtidos através de simulação, predizem uma boa qualidade no ajuste dos controladores, haja vista a velocidade de resposta da malha, bem como a verificação apenas da oscilação de chaveamento em regime permanente.

5.2.2 Simulação da malha de rotação do sistema gerador

A referência da malha de corrente é obtida através de uma malha mais externa de rotação. A validação da malha de rotação foi também feita no software PSIM, sendo utilizado o bloco de máquinas síncronas de ímã permanente (*PMSM*) para simular o comportamento do gerador. Além disso, utiliza-se no lugar do conversor Boost uma fonte de corrente controlada,

uma vez que a dinâmica da malha de corrente é muito mais lenta. A Figura 63 apresenta o diagrama da simulação desenvolvida.

Figura 63 - Diagrama de simulação da malha de rotação do PMSG

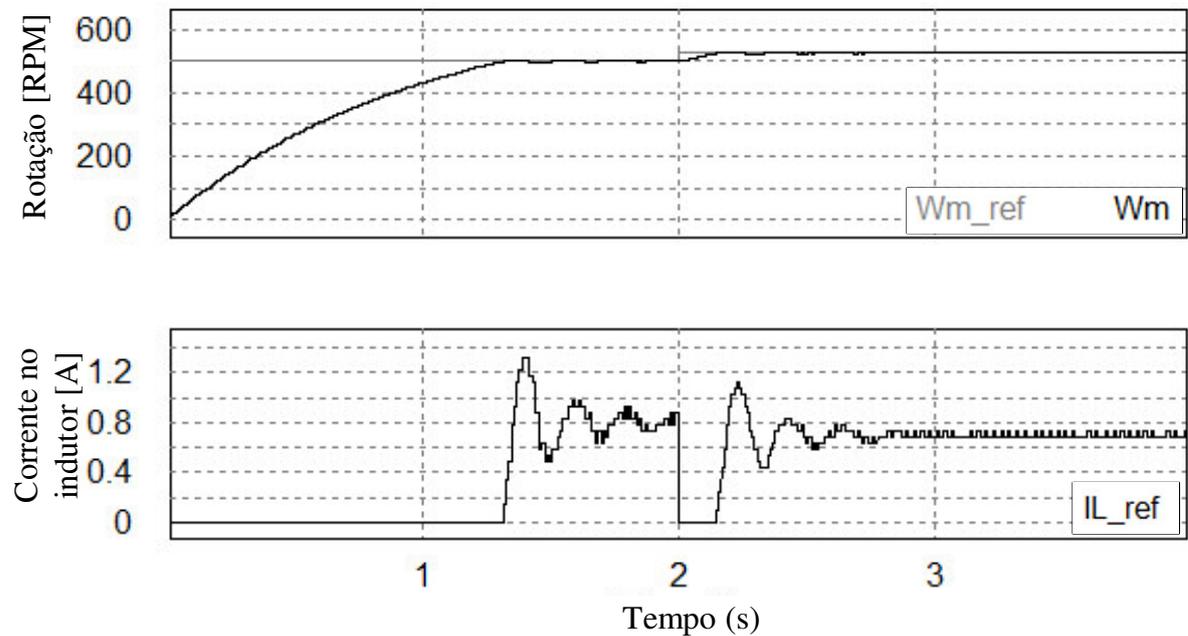


Fonte: Esquemático desenvolvido no software PSIM

Os resultados do comportamento da rotação durante a partida do sistema MCC-PMSG são apresentados na Figura 64. Nota-se que a utilização do *Anti-Windup* evita sobressinal de rotação, mesmo durante a partida.

Já a Figura 65 apresenta mais detalhadamente o comportamento dinâmico da malha de rotação. Neste gráfico é possível verificar que um degrau faz com que a referência de rotação passe de 550 RPM para 575 RPM. Observa-se um tempo de assentamento inferior a 0,5 s, e a estabilização da referência de corrente do indutor do conversor *Boost*.

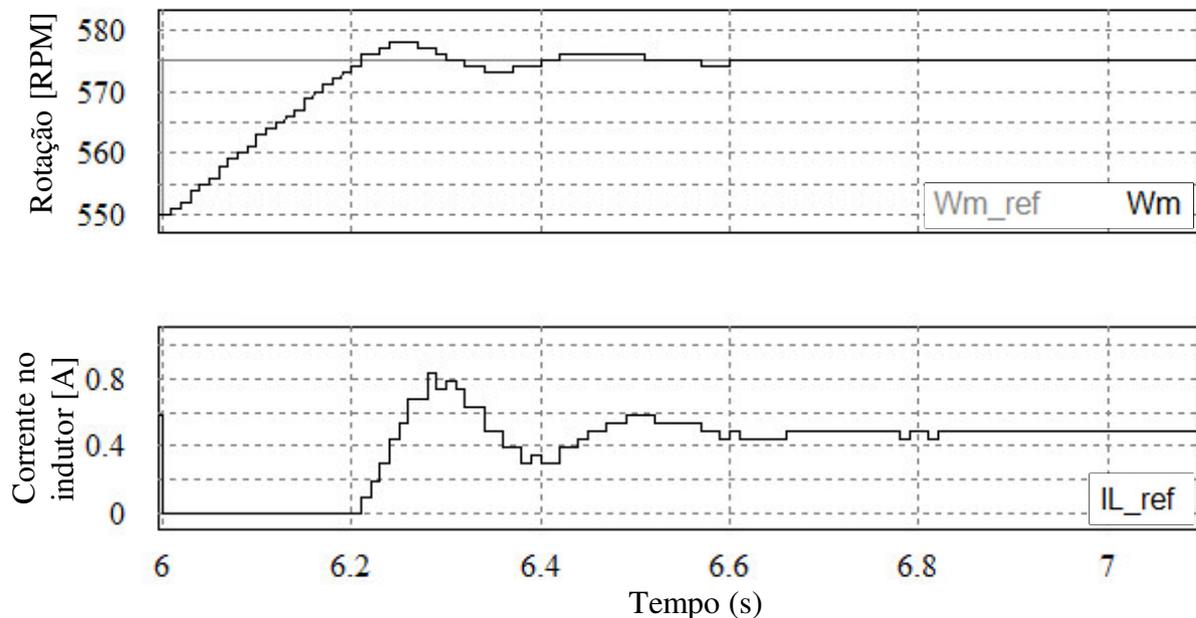
Figura 64 – Gráfico de atuação do *Ant-Windup* na malha de rotação do controlador de carga



Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

Desta forma, o sistema de físico é validado em sua totalidade por meio de simulações computacionais, passando-se a fase de validação experimental.

Figura 65 – Detalhe do comportamento dinâmico da malha de rotação



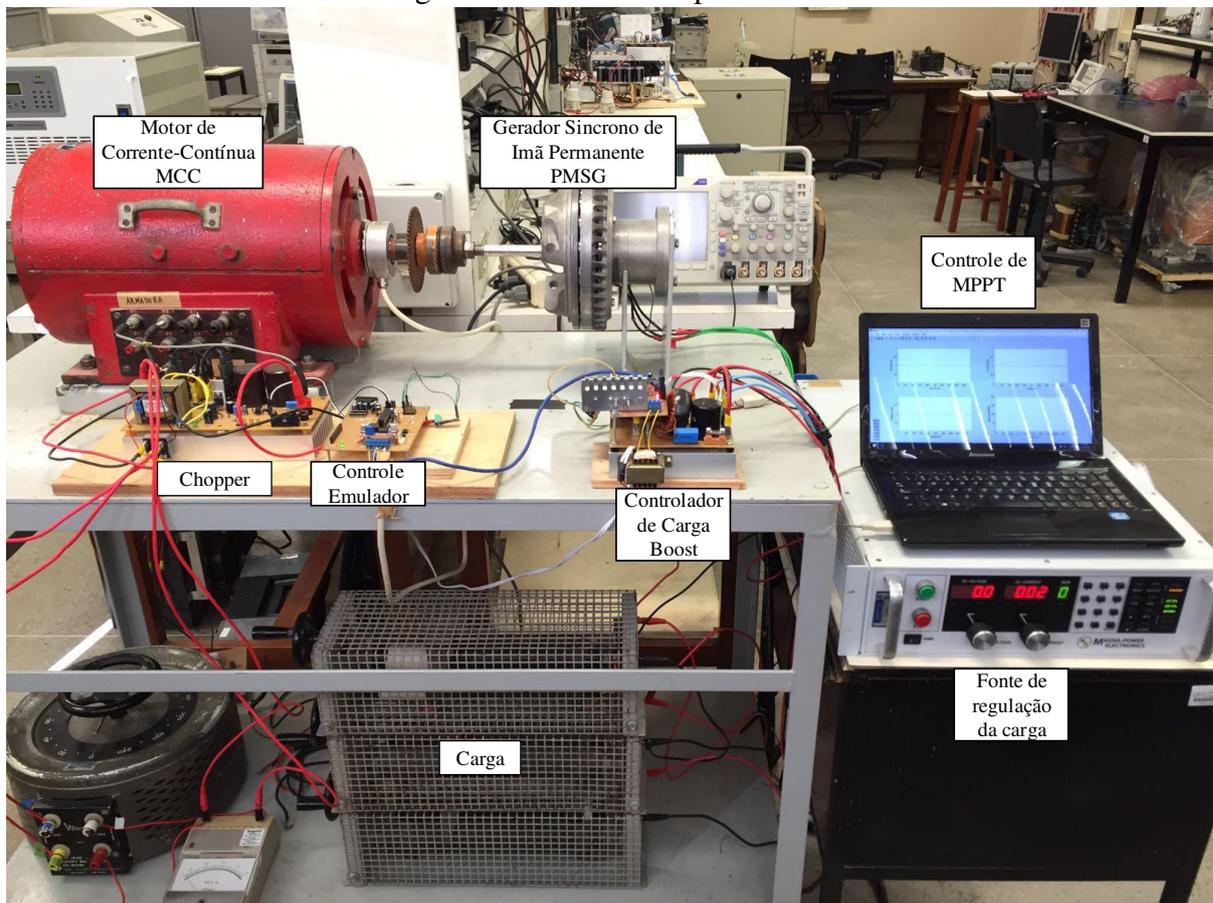
Fonte: Obtido no aplicativo SIMVIEW do software PSIM.

5.3 Validação da bancada experimental

A bancada experimental é montada baseada nos parâmetros já apresentados. Uma fotografia da mesma é apresentada na Figura 66.

A validação do sistema experimental é feita primeiramente através de ensaios individuais das malhas, contudo, há uma ordem a ser seguida, para que seja possível validar corretamente todo o sistema.

Figura 66 – Bancada experimental



Fonte: desenvolvido pelo autor.

- A primeira malha testada é a de corrente no indutor do conversor Boost responsável pelo controle de carga. Esse teste é feito utilizando-se uma fonte CC de 120V na entrada e outra, unidirecional, de 150V conectada em paralelo a carga. Aplicando-se degraus de referência, observa-se a correta operação da malha.
- Testa-se então a malha de corrente do sistema emulador, no qual se faz necessário aplicar um valor de frenagem eletromecânica por meio do PMSG

(malha de corrente do indutor do conversor Boost). Utilizam-se degraus de referência para validar esta malha.

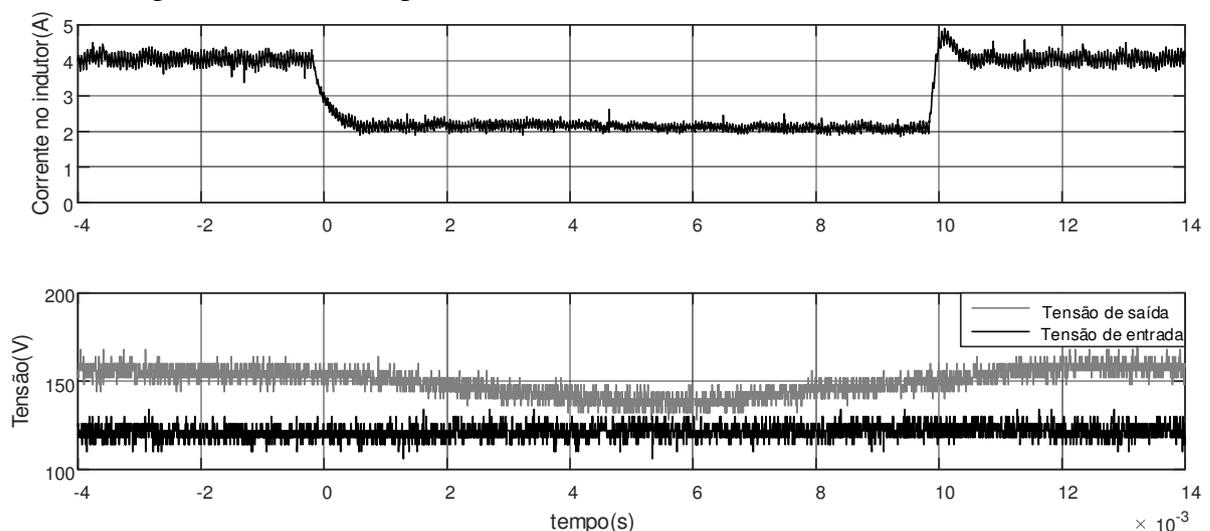
- c) Novamente no lado do PMSG, realiza-se a validação da malha de rotação feita pelo conversor Boost. Para tal, faz-se necessário que uma corrente de armadura seja aplicada no MCC. Os testes são realizados através de degraus de referência de rotação, e com variações na corrente de armadura do MCC.
- d) Por fim, analisa-se o sinal de referência de corrente que o algoritmo emulador está enviando ao controlador, através da observação em regime permanente da corrente de armadura do MCC, dado uma velocidade de vento e rotação constante.

Dessa forma, cada procedimento de validação da bancada experimental é descrito nas seções seguintes.

5.3.1 Validação da malha de corrente do indutor do conversor Boost

Utilizando duas fontes de tensão CC unidirecionais, conecta-se uma de 120V na entrada do conversor Boost, e uma de 150V em paralelo com a carga do mesmo, a fim de manter a regulação da tensão na mesma. A Figura 67 apresenta a estabilidade de regime permanente obtida para referências de 2A e 4A. Embora seja possível notar uma oscilação na tensão de saída, a mesma ainda é estável o suficiente para não interferir na dinâmica da corrente.

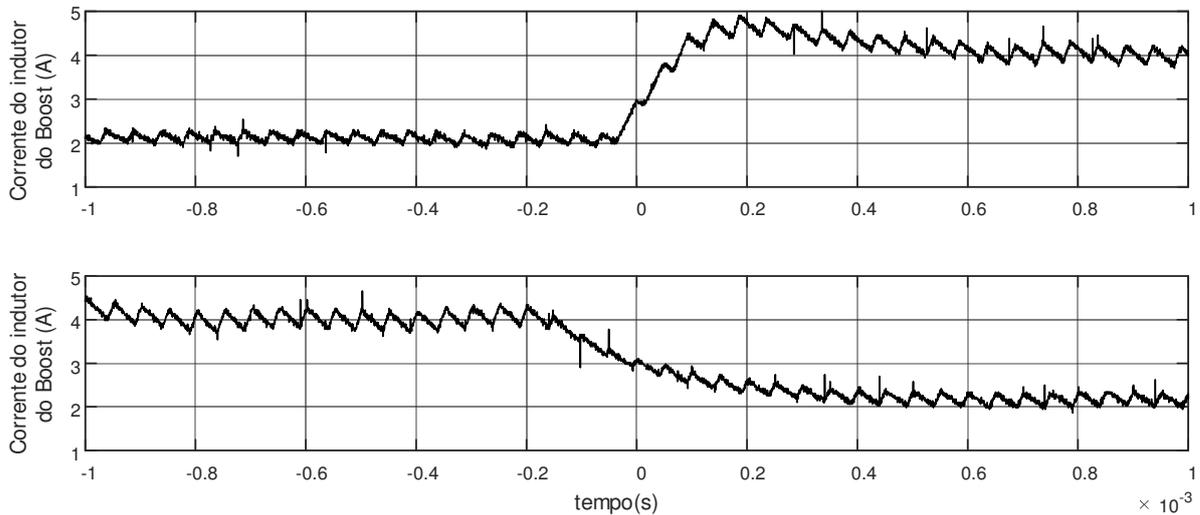
Figura 67 – Malha experimental de corrente no indutor do conversor Boost



Fonte: desenvolvido pelo autor.

A Figura 68 apresenta o detalhe da dinâmica da corrente quando submetida a um degrau de referência de 50%, de subida e descida. Nota-se uma boa resposta, uma vez que em ambos os casos o tempo de assentamento fica em torno de 0,5 ms.

Figura 68 – Detalhe da dinâmica experimental da corrente no indutor do conversor Boost



Fonte: desenvolvido pelo autor.

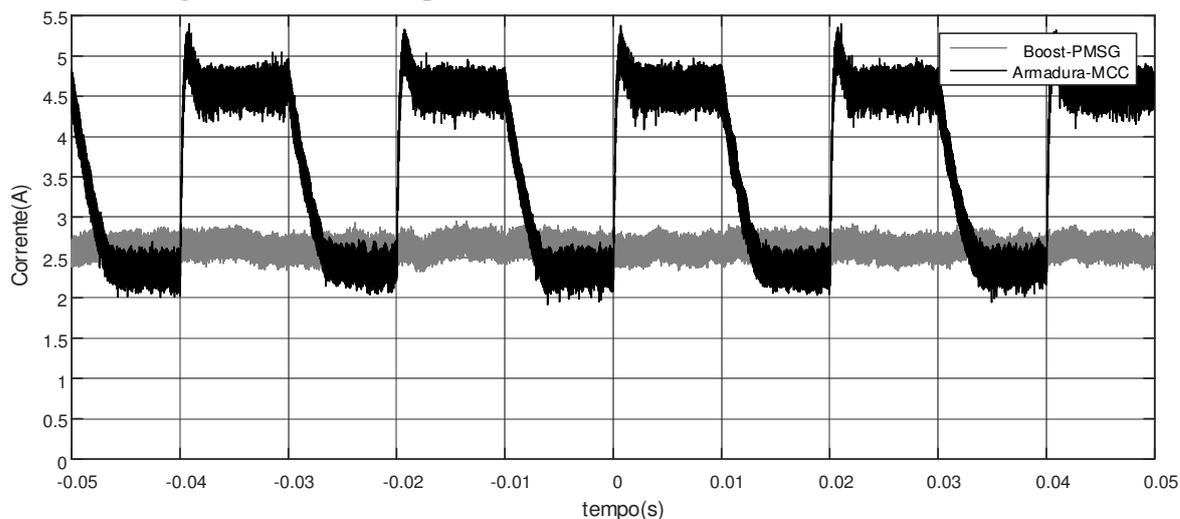
Dessa forma, comprova-se a correta operação do controlador de corrente do conversor *Boost*.

5.3.2 Validação da malha de corrente de armadura do MCC

Para realizar a validação experimental da malha de corrente de armadura do MCC, faz-se necessário aplicar uma frenagem eletromecânica por meio do PMSG, uma vez que operando a vazio, a velocidade de rotação do MCC, irá alcançar níveis onde a tensão induzida nos enrolamentos seja próxima a tensão nos terminais de armadura, impossibilitando o controle de corrente nos enrolamentos, devido à baixa tensão disponível. Dessa forma, aplica-se uma corrente de referência de 2,5A no indutor do conversor *Boost* e aplica-se uma referência de corrente de armadura com perfil de onda quadrada, com patamares de 4,5A e 2,25A. A Figura 69 apresenta os valores obtidos experimentalmente.

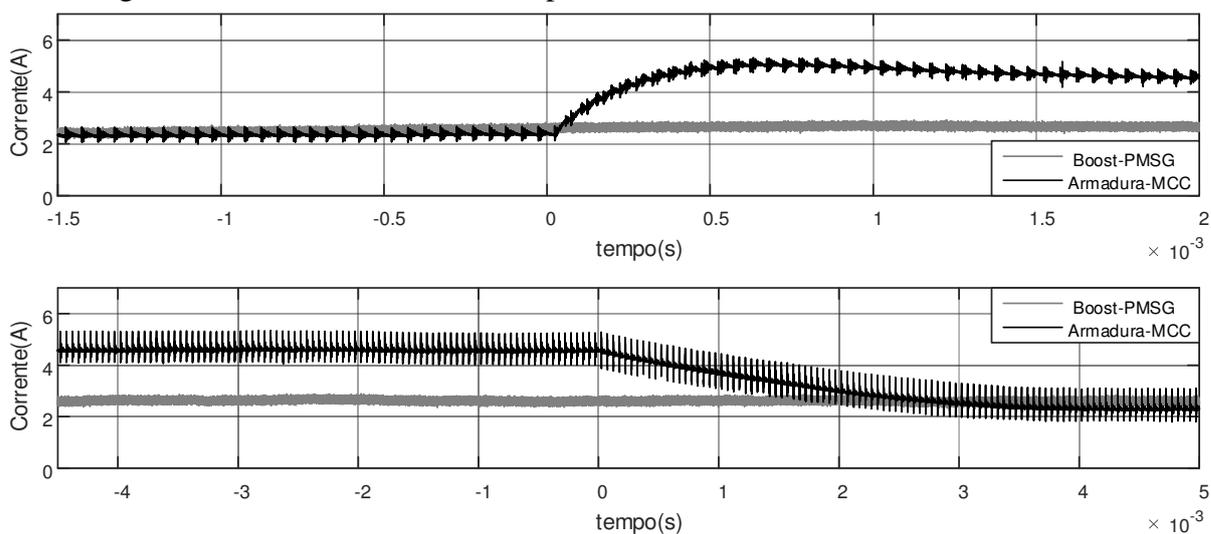
É possível observar a capacidade da malha em de fato estabilizar o valor da corrente de armadura do MCC em regime permanente. Já na Figura 70 o detalhe da dinâmica do controle fica mais clara, é possível notar uma resposta mais rápida no degrau de subida, sendo o tempo de assentamento em ambos os casos inferior a 4ms.

Figura 69 - Malha experimental de corrente no de armadura do MCC



Fonte: desenvolvido pelo autor.

Figura 70 - Detalhe da dinâmica experimental da corrente de armadura do MCC



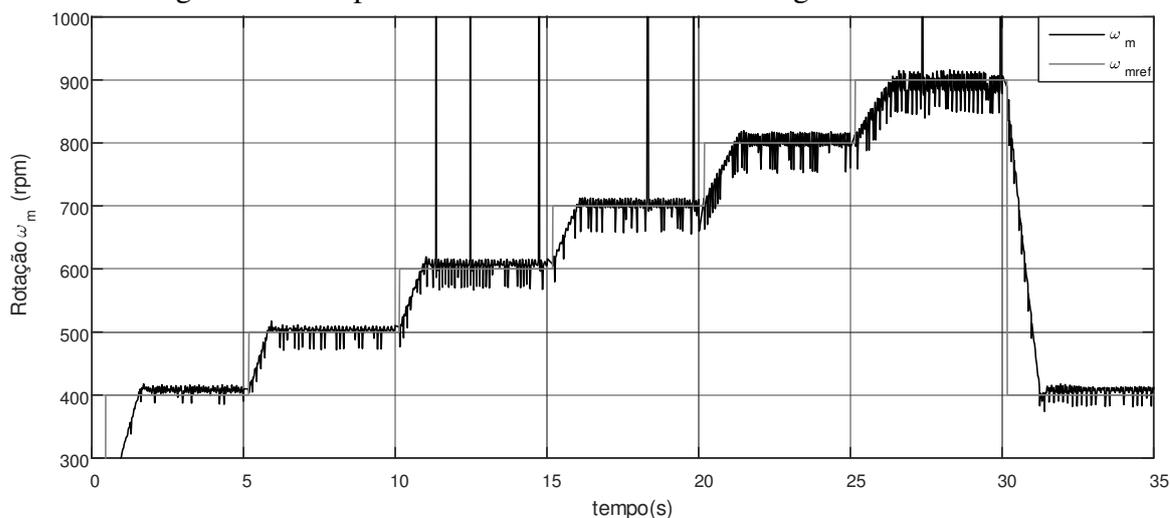
Fonte: desenvolvido pelo autor.

Dessa forma, considera-se validada a malha de corrente de armadura do MCC.

5.3.3 Validação da malha de rotação feita pelo controlador de carga

A validação da malha de velocidade é feita com base nos valores medidos e enviados ao computador através da interface serial. Dessa forma, definiu-se um perfil de referência de rotação, estabilizou-se a corrente de armadura do MCC em 2,25A, obtendo-se assim o comportamento apresentado na Figura 71.

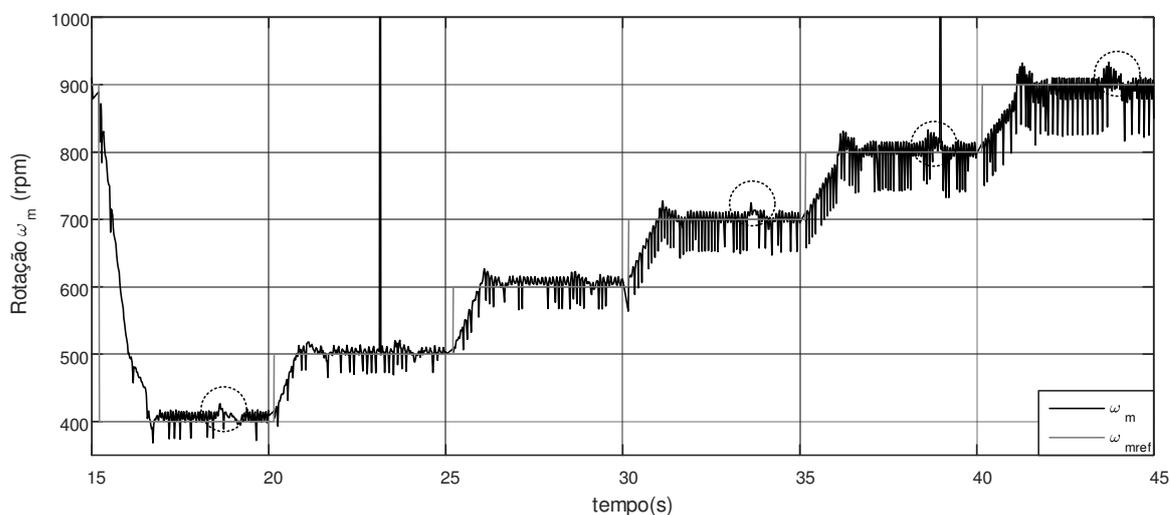
Figura 71 – Resposta da malha de velocidade a degraus de referência



Fonte: desenvolvido pelo autor.

Buscando observar a robustez da malha de rotação, inseriu-se no valor de corrente de armadura, degraus de 4,5 A com duração de 0,5 segundos, a cada 2 segundos. O resultado obtido é mostrado na Figura 72. Na figura, a referência é apresentada em cinza.

Figura 72 - Resposta da malha de velocidade a degraus de referência e variações do torque de carga



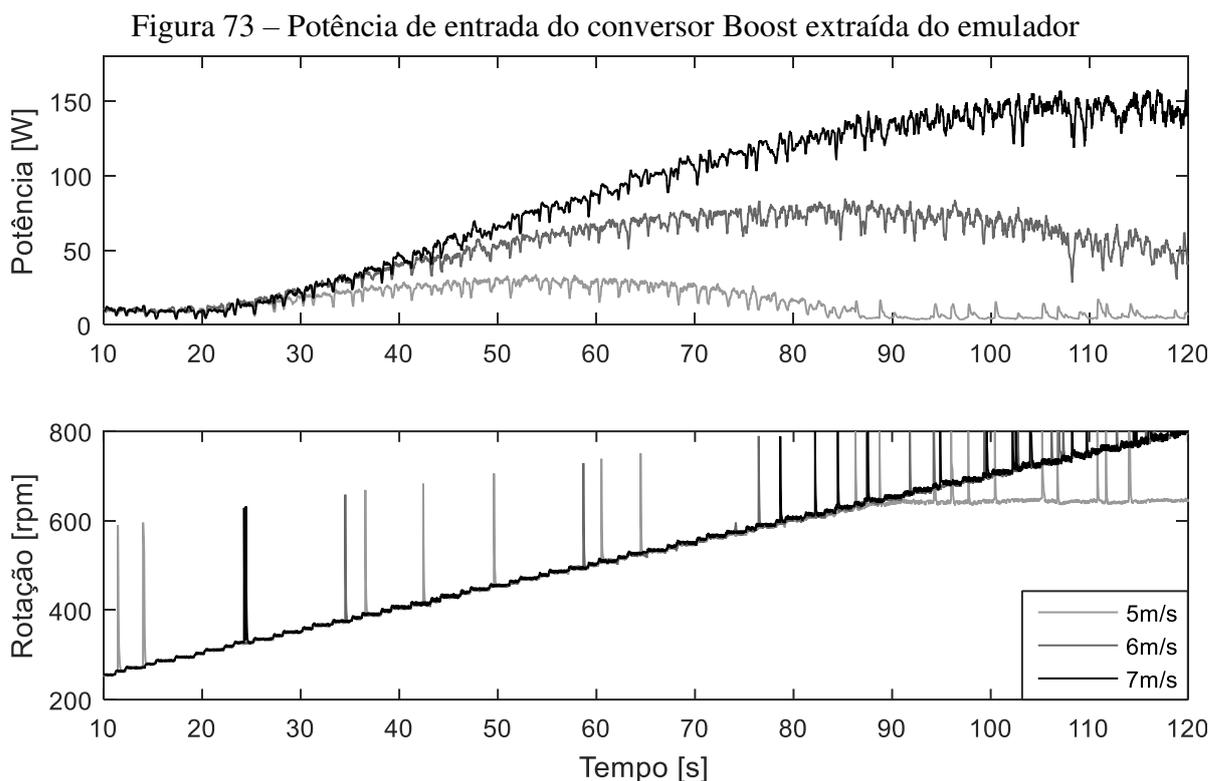
Fonte: desenvolvido pelo autor.

Na Figura 72 é possível observar em certos momentos a perturbação na velocidade advinda da variação do torque imposto pelo MCC, contudo, a rápida dinâmica de ajuste, permite afirmar que a malha de controle de rotação feita pelo controlador de carga funciona como esperado.

5.3.4 Validação do sistema de emulação baseado em MCC

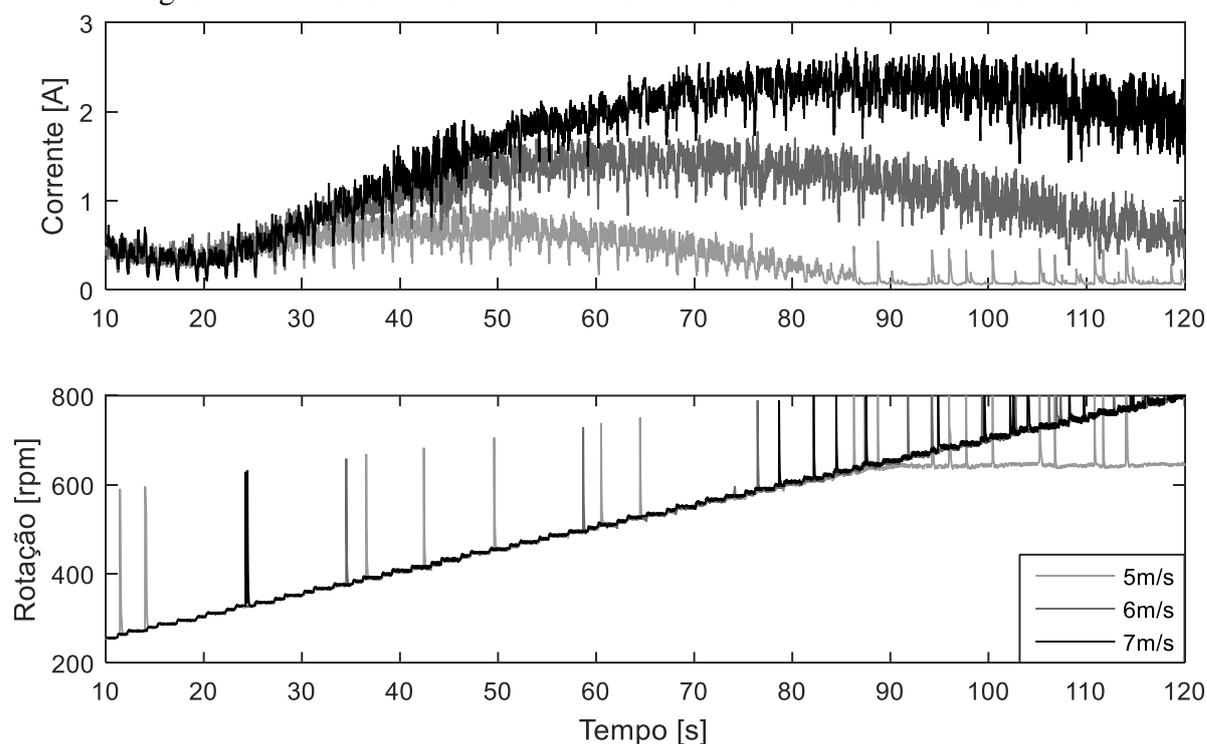
O sistema de emulação, como já citado, tem como função aplicar um torque mecânico no eixo equivalente ao que seria imposto por pás de um aero gerador que estejam submetidas a valores mínimos de velocidade de vento. Dessa forma, é possível determinar através de equações fictícias de comportamento da função $C_p(\lambda, \beta)$, o valor de torque e consequentemente de corrente deve ser imposto no MCC.

Uma vez que a malha de corrente de armadura do motor de corrente contínua já tenha sido validada, para admitir que o emulador esteja operando de forma correta, realizam-se medições de corrente em regime, quando aplicado uma rampa na referência de velocidade de rotação do eixo. O ensaio é repetido para velocidades de vento de 5, 6 e 7 m/s, e os resultados obtidos são expressos através da potência e da corrente vista na entrada do conversor boost controlador de carga, como visto na Figura 73 e na Figura 74



Fonte: desenvolvido pelo autor.

Figura 74 – Corrente do indutor do conversor Boost extraída do emulador



Fonte: desenvolvido pelo autor.

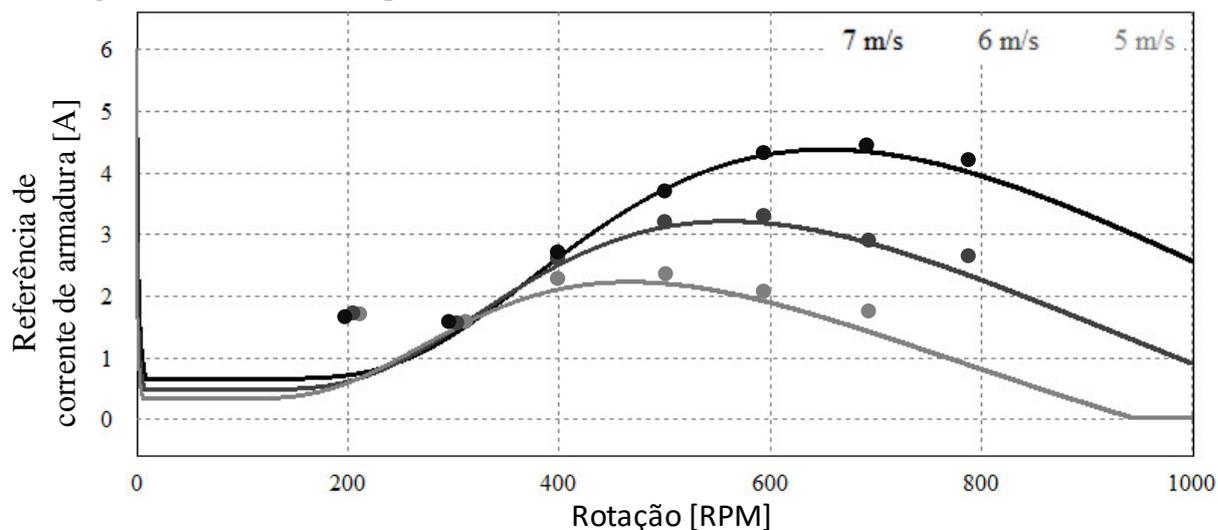
Com o objetivo de observar diretamente o valor da corrente de armadura do MCC ante à referência proveniente do equacionamento, realiza-se medições de corrente para sete valores distintos de velocidade de rotação e os valores de vento de 5, 6 e 7 m/s. Os valores medidos são apresentados na Tabela 11 e posicionados frente aos resultados de simulação previamente elaborados na Figura 75. Ressaltam-se dois pontos importantes nessa figura: a falta do valor de corrente para velocidade de rotação de 800 rpm, com vento de 5m/s devido a incapacidade do emulador de acelerar o gerador, uma vez que a energia extraída nessa configuração é muito baixa; Além disso, nota-se que para um baixo valor de rotação, o erro entre a corrente medida e a referência é grande devido à elevada diferença de tensão nos terminais de armadura e a tensão induzida nos enrolamentos da MCC.

Tabela 11 – Dados de ensaio da corrente de armadura do emulador.

Velocidade do vento	Velocidade de rotação do eixo						
	200 rpm	300 rpm	400 rpm	500 rpm	600 rpm	700 rpm	800 rpm
5 m/s	1,75 A	1,57 A	2,10 A	2,20 A	2,00 A	1,85 A	-- A
6 m/s	1,75 A	1,60 A	2,53 A	3,05 A	3,15 A	2,90 A	2,60 A
7 m/s	1,70 A	1,60 A	2,63 A	3,62 A	4,20 A	4,35 A	4,20 A

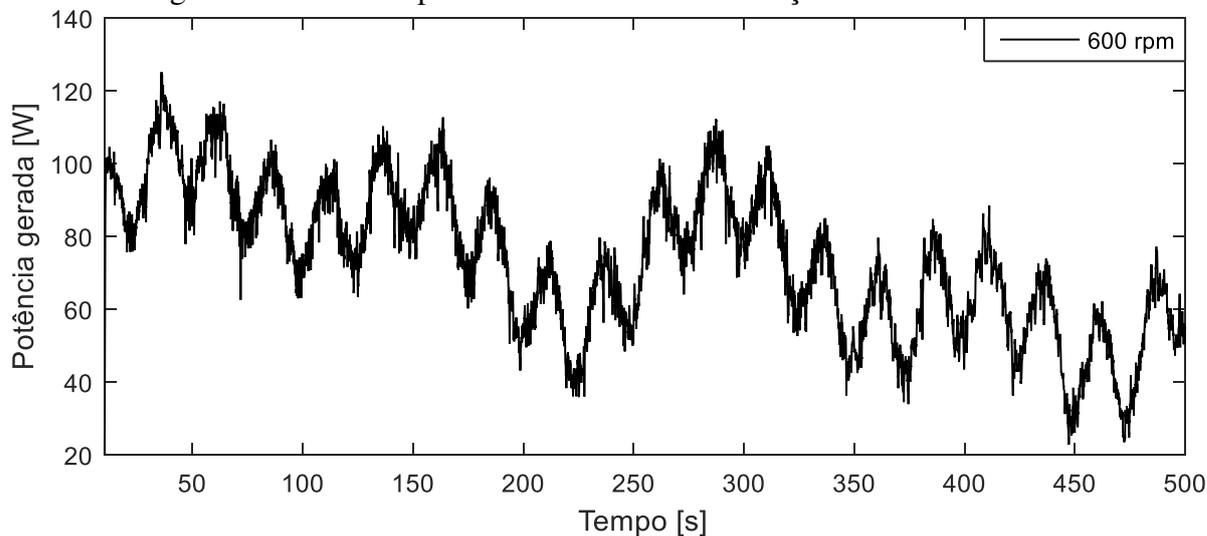
Fonte: desenvolvido pelo autor.

Figura 75 – Gráfico comparativo da corrente de armadura do MCC e sua referência.



É também realizada a análise do emulador operando a uma velocidade de rotação constante de 600 rpm, variando-se a velocidade do vento. A Figura 76 apresenta os resultados obtidos nesse ensaio.

Figura 76 – Perfil de potência extraída com emulação de vento variável.



Dessa forma, conclui-se a validação da bancada experimental, o que permite então o desenvolvimento dos testes de algoritmos de rastreamento de máxima potência.

5.4 Simulações do algoritmo rastreador de máxima potência

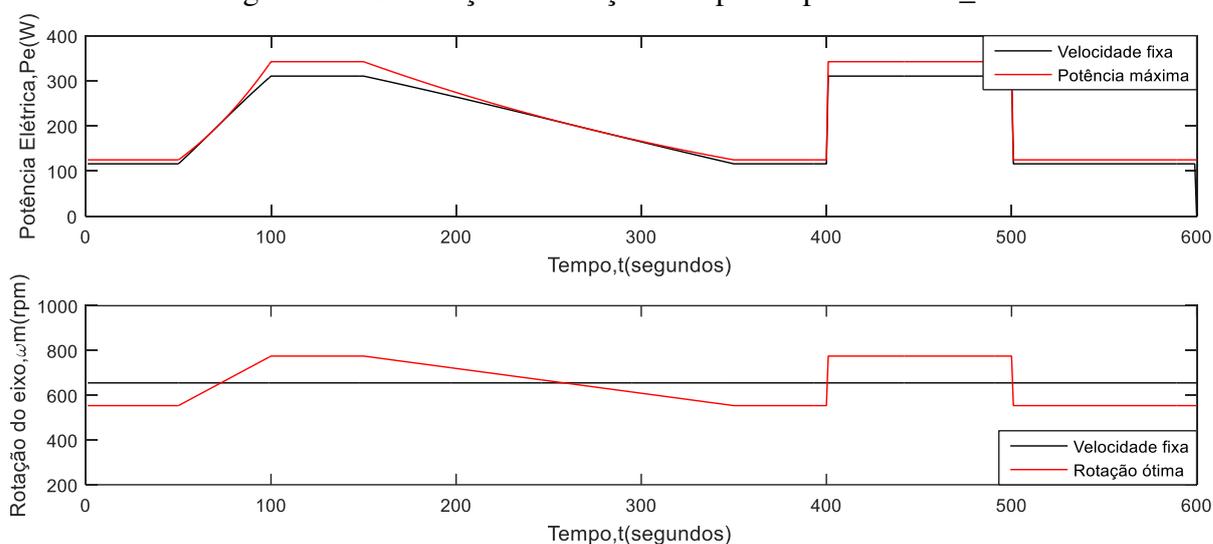
Com o objetivo de observar o comportamento dos algoritmos propostos diante da aplicação de um gerador eólico, utiliza-se o software computacional Matlab para realização de simulações. Uma vez que a malha de velocidade de rotação do eixo já tenha sido validada, busca-se reduzir o esforço computacional da simulação, desprezando-se a dinâmica mecânica e elétrica do sistema. Para a simulação, utilizou-se uma função cujo comportamento mimetiza o de uma turbina, e na qual são aplicadas variações de velocidade de rotação e vento, a fim de observar a dinâmica dos algoritmos de MPPT propostos. Os códigos utilizados nesta simulação são apresentados no Apêndice G: Códigos de MPPT MATLAB. Os resultados obtidos são apresentados nos tópicos seguintes.

As simulações apresentam além dos algoritmos propostos, também métodos convencionais, como a operação da turbina em velocidade constante e através da utilização do MPPT P&O.

5.4.1 Simulações com velocidade de rotação fixa

Como em turbinas de grande porte é comum a operação sob velocidade de rotação fixa, optou-se por realizar um comparativo dos algoritmos propostos com essa metodologia de controle. A Figura 77 apresenta os valores de potência extraída ao operar em velocidade fixa e a potência máxima extraível, bem como a velocidade de rotação ótima para obtenção dessa potência máxima.

Figura 77 – Simulação de rotação fixa para o perfil Vento_II

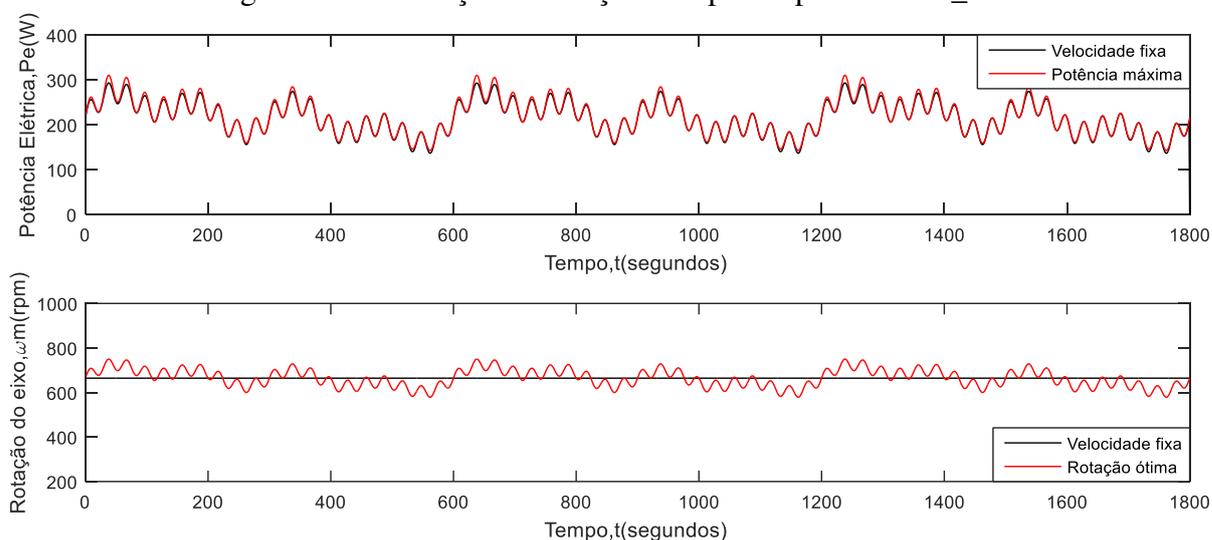


Fonte: desenvolvido pelo autor.

Em projetos práticos, faz-se necessário ter um conhecimento prévio do valor de vento médio da localização, para assim, definir com base na curva aerodinâmica do aerogerador o valor de rotação que permite maior extração de energia. Nas simulações desenvolvidas, utiliza-se o valor médio do perfil de rotação ótima. No caso do perfil Vento_I, tem-se 100% de potência absorvida, uma vez que a velocidade de rotação ótima é fixo, e portanto, numericamente igual à sua média.

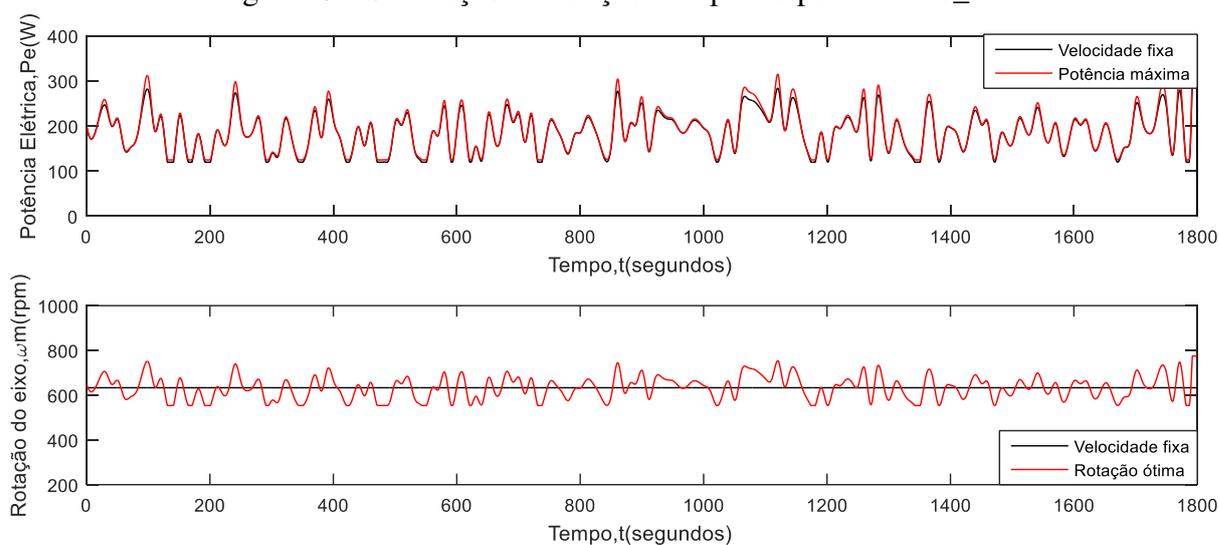
Quando utilizado o perfil Vento_II há uma queda da potência de geração como visto na Figura 77.

Figura 78 - Simulação de rotação fixa para o perfil Vento_III



Fonte: desenvolvido pelo autor.

Figura 79 - Simulação de rotação fixa para o perfil Vento_IV



Fonte: desenvolvido pelo autor.

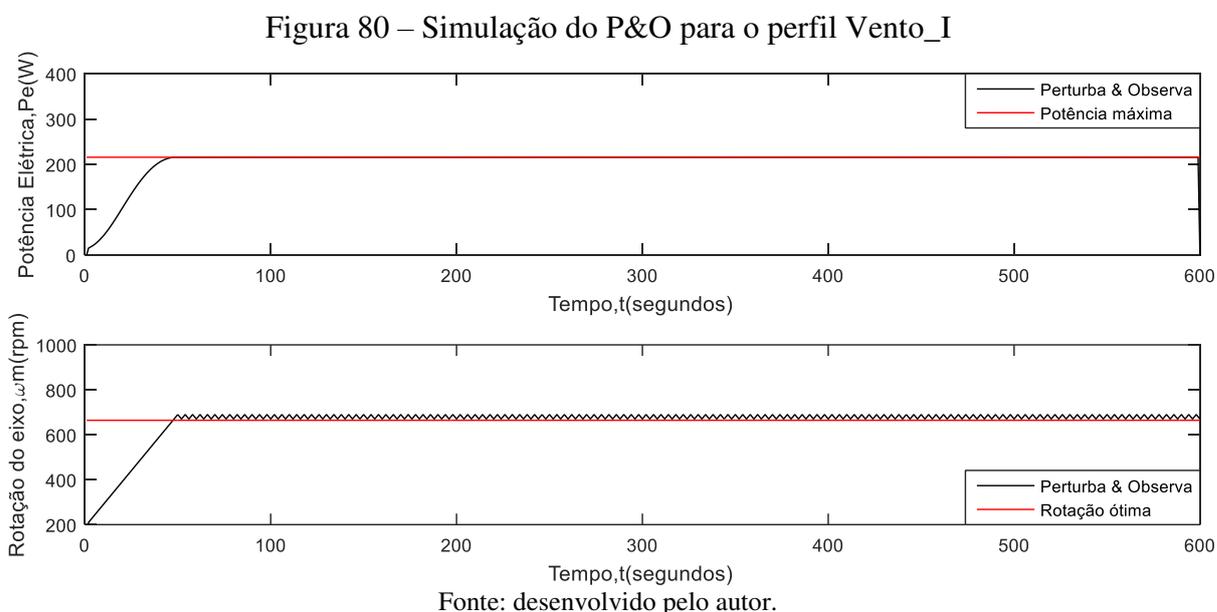
Para os demais perfis de vento a oscilação novamente reduz a potência extraída, mantendo, contudo, um valor de potência consideravelmente alto, quando comparado com os demais algoritmos, como é mostrado mais a frente. A Figura 78 apresenta os resultados para o perfil Vento_III e a Figura 79 os resultados para Vento_IV.

5.4.2 Simulações do MPPT P&O de referência

Com o objetivo de comparar os algoritmos propostos com um modelo de MPPT já difundido, optou-se pelo P&O de passo fixo, cuja implementação é extremamente simples, com baixo esforço computacional e baixa necessidade de memória. Os parâmetros do algoritmo foram: passo de perturbação de 10 rpm e tempo entre iterações de 1 segundo.

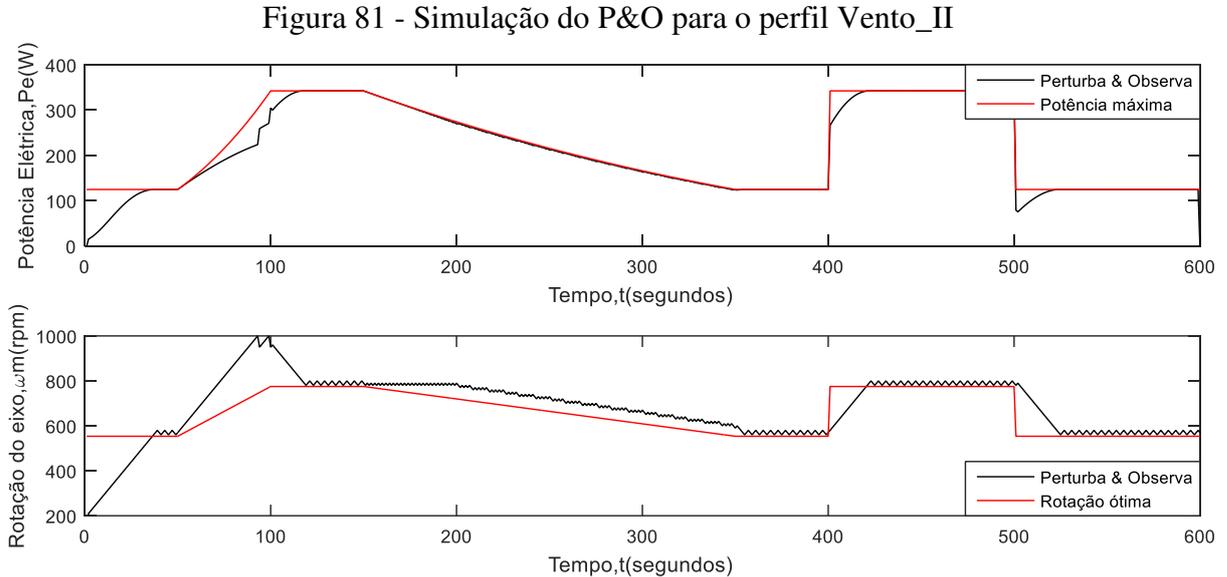
Inicialmente observa-se o comportamento do P&O para o valor fixo do perfil Vento_I. Na Figura 80 observa-se que de fato o algoritmo faz com que a velocidade de rotação do eixo fique oscilante em torno do ponto ótimo, de forma que a energia extraída seja próxima da máxima.

Quando submetido ao perfil Vento_II, o comportamento dinâmico do algoritmo passa a ser observado. A Figura 81 apresenta os resultados de simulação do mesmo.

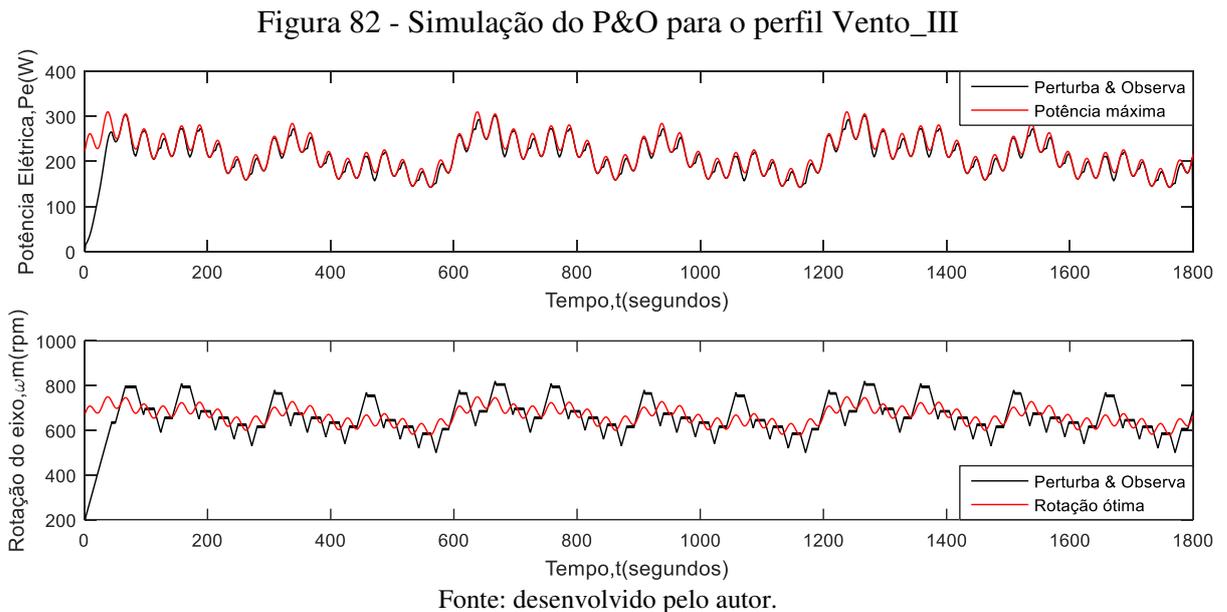


Já para o perfil de vento Vento_III, observa-se um pouco mais de dificuldade do algoritmo manter-se próximo ao valor ótimo de velocidade de rotação, devido à continua variação de vento. Salienta-se a não melhoria de resposta, mesmo com a periodicidade do perfil.

A Figura 82 apresenta os valores de potência absorvida e de velocidade de rotação, do algoritmo P&O quando submetido ao perfil Vento_III.

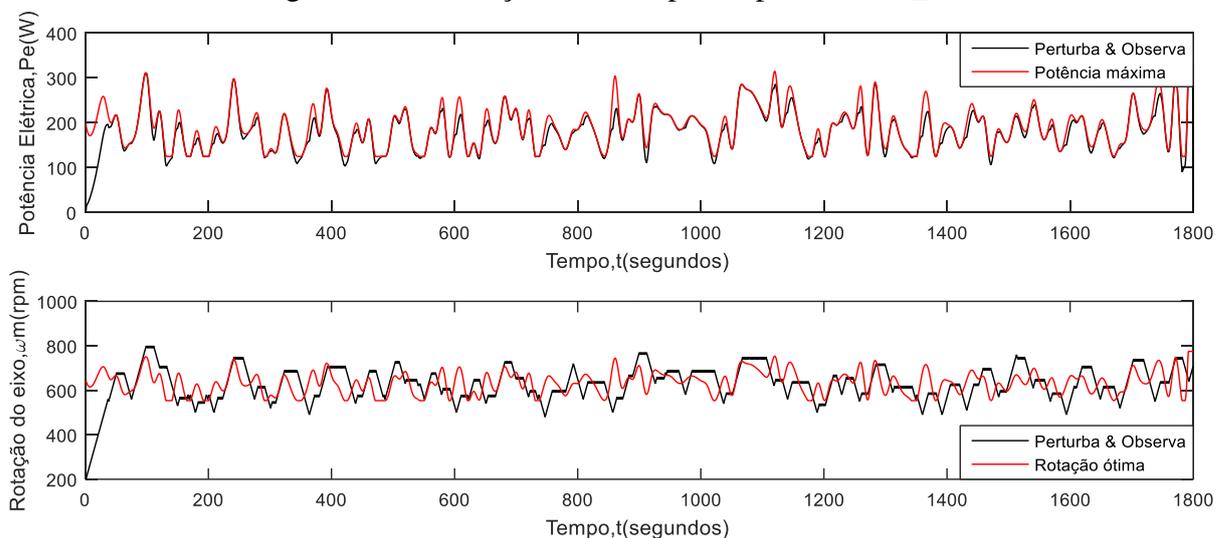


O perfil Vento_IV apresenta elevada variação do vento, por ser formado através de funções aleatórias, isto faz com que o algoritmo P&O tenha certa dificuldade em seguir a referência de rotação ótima. A Figura 83 apresenta os resultados de simulação de comprovam isto.



É válido salientar que a definição do passo de perturbação pode ser trabalhosa, uma vez que deve ser estabelecido de forma a equilibrar o tempo de resposta do algoritmo e o erro de regime obtido pelo mesmo.

Figura 83 - Simulação do P&O para o perfil Vento_IV



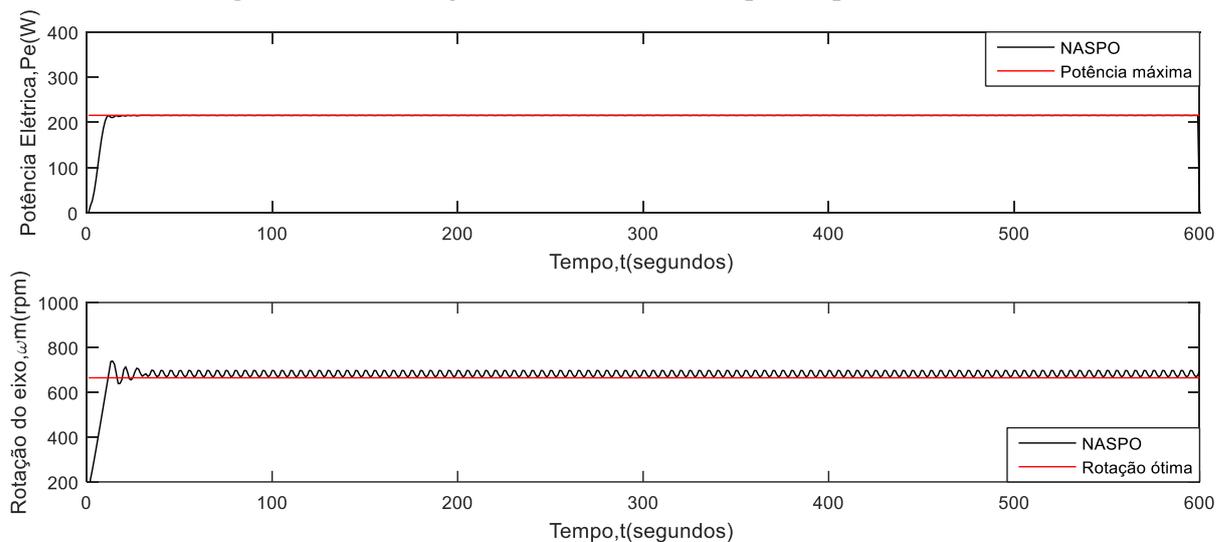
Fonte: desenvolvido pelo autor.

5.4.3 Simulações do MPPT NASPO

O primeiro algoritmo proposto a ser analisado por meio das simulações computacionais é o NASPO. A fim de comprovar a capacidade do algoritmo de atingir o ponto de máxima potência, realiza-se a simulação utilizando o perfil Vento_I. O resultado obtido é apresentado na Figura 84. Verifica-se que de fato o algoritmo atinge o ponto de máxima potência, atingindo uma baixa oscilação após a estabilização.

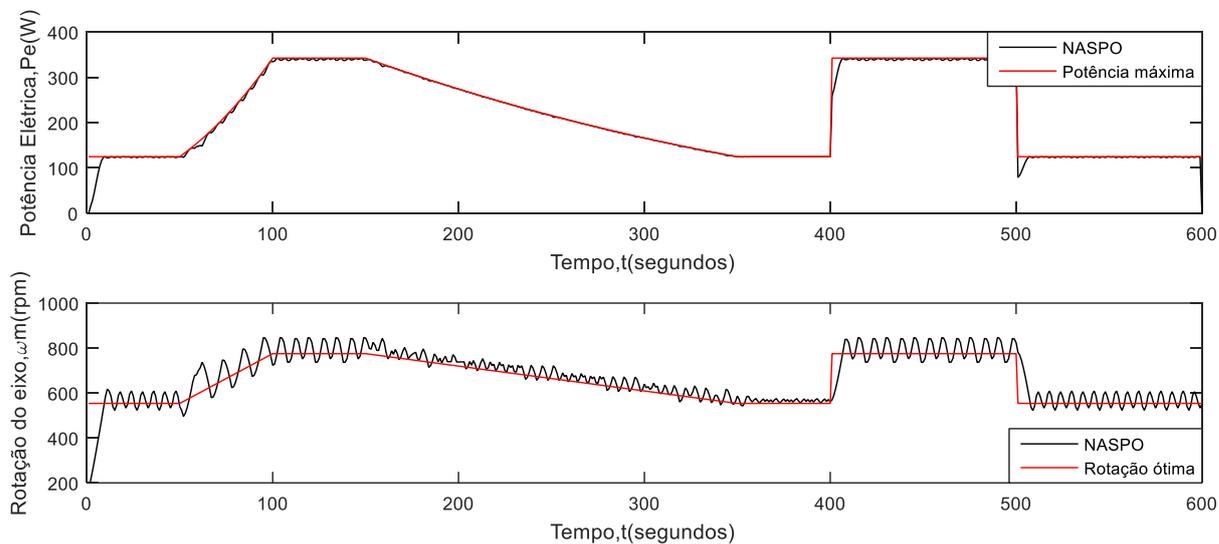
Para observar o comportamento dinâmico do algoritmo NASPO, realiza-se então a simulação para o perfil de vento Vento_II. A Figura 85 apresenta o resultado obtido. Quando comparado ao algoritmo P&O, o algoritmo NASPO apresenta uma melhor resposta à dinâmica dos ventos. A Figura 86 e a Figura 87 apresentam os resultados obtidos para os perfis Vento_III e Vento_IV respectivamente.

Figura 84 - Simulação do MPPT NASPO para o perfil Vento_I



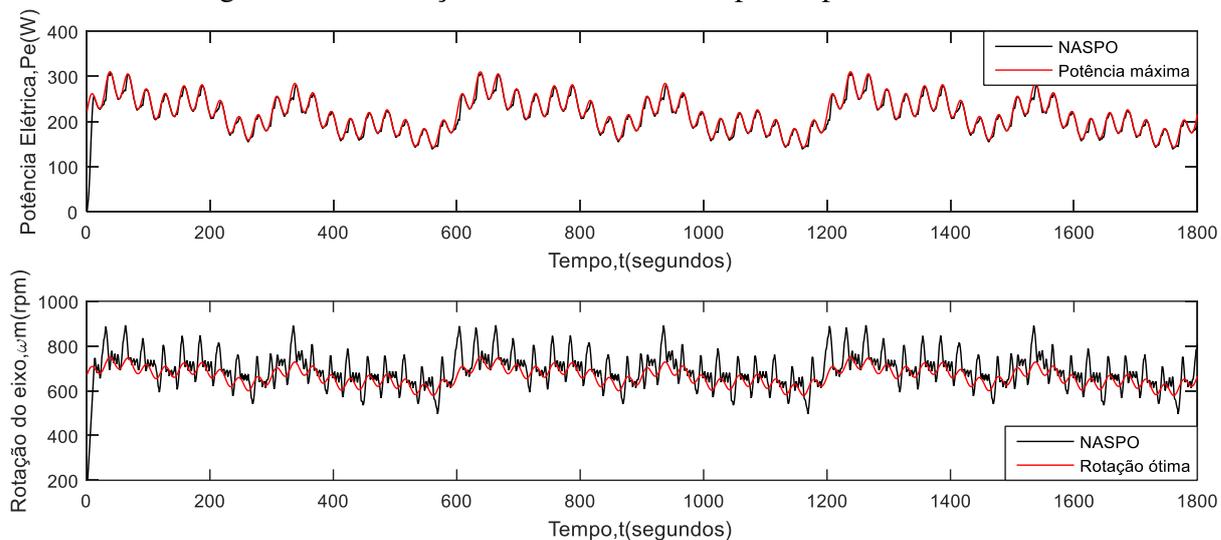
Percebe-se através da análise dos resultados que o algoritmo NASPO mostra-se mais eficiente ao perfil periódico. Isso ocorre devido a característica de aprendizagem do mesmo, o que resulta em uma melhor resposta, quando um perfil se repete.

Figura 85 - Simulação do MPPT NASPO para o perfil Vento_II



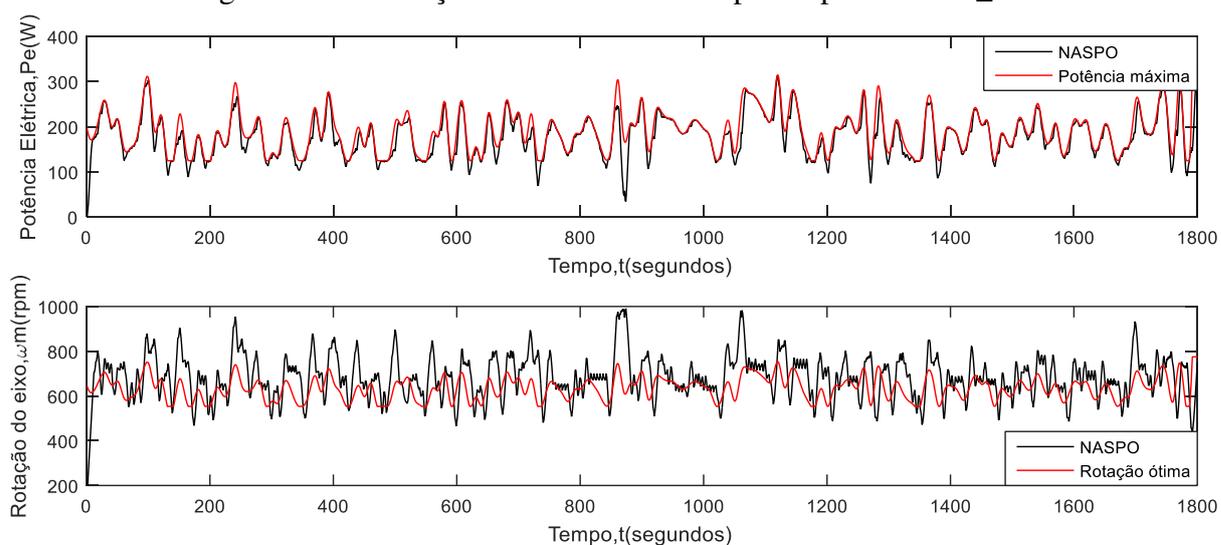
Os resultados apresentados mostram que o algoritmo NASPO tem a capacidade de realizar o rastreamento do ponto de máxima potência, contudo, não é suficiente ainda, para definir sua superioridade ante aos demais algoritmos. Tal análise será feita mais à frente.

Figura 86 - Simulação do MPPT NASPO para o perfil Vento_III



Fonte: desenvolvido pelo autor.

Figura 87 - Simulação do MPPT NASPO para o perfil Vento_IV



Fonte: desenvolvido pelo autor.

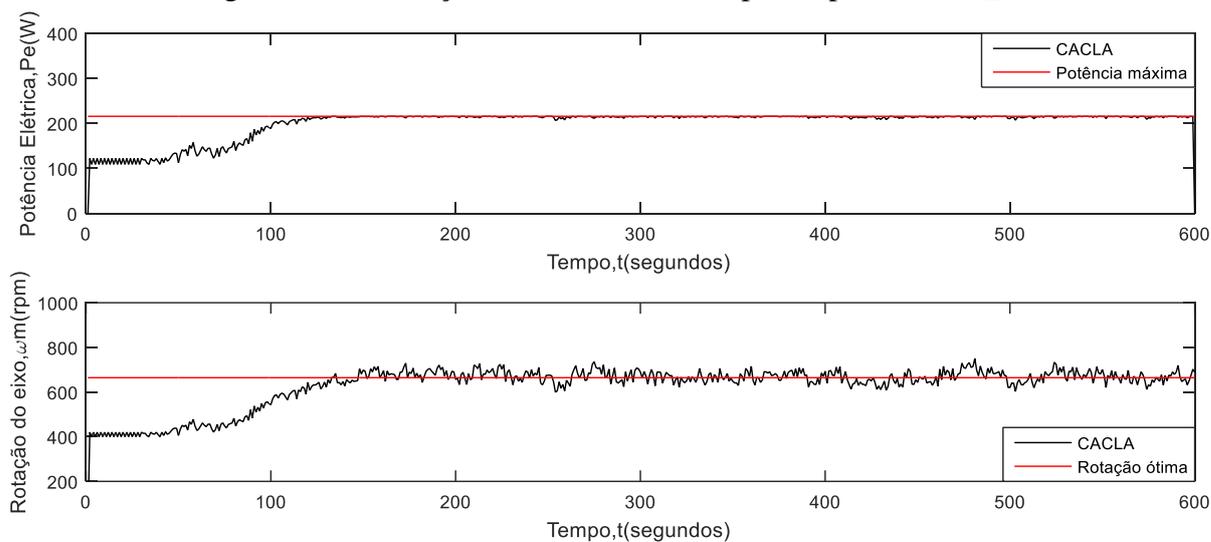
5.4.4 Simulações do MPPT CACLA

Assim como para os métodos anteriores, realiza-se também as simulações do algoritmo CACLA através do software Matlab, a fim de observar o comportamento da referência de rotação do eixo e a potência gerada quando submetida aos quatro perfis de vento.

Testa-se inicialmente o comportamento do algoritmo frente ao perfil de vento Vento_I. A Figura 88 apresenta o resultado obtido e percebe-se, então, que sem nenhum aprendizado prévio, o algoritmo leva cerca de 120 segundos para alcançar a valores de referência de rotação próximo ao valor de máxima potência.

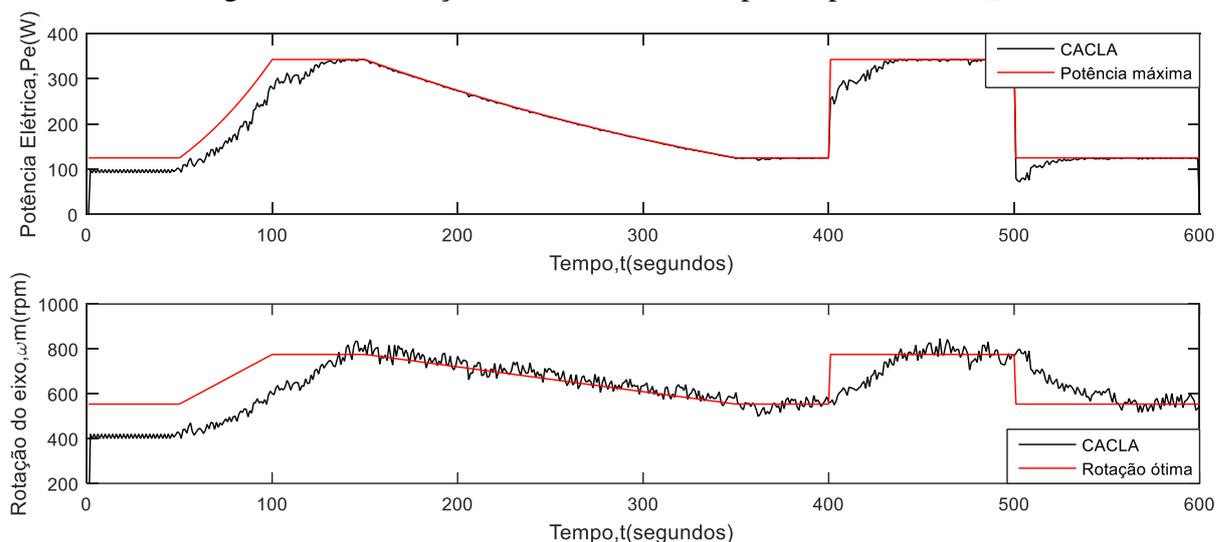
Em seguida, submete-se o algoritmo a uma simulação com o perfil Vento_II, buscando verificar a resposta do mesmo diante de dinâmicas na velocidade do vento. A Figura 89 apresenta esses resultados. Fica claro uma maior inércia do sistema, o que é visto devido ao fator de aprendizagem do mesmo que acaba por retardar a resposta, quando um padrão de vento é apresentado pela primeira vez.

Figura 88 - Simulação do MPPT CACLA para o perfil Vento_I



Fonte: desenvolvido pelo autor.

Figura 89 - Simulação do MPPT CACLA para o perfil Vento_II

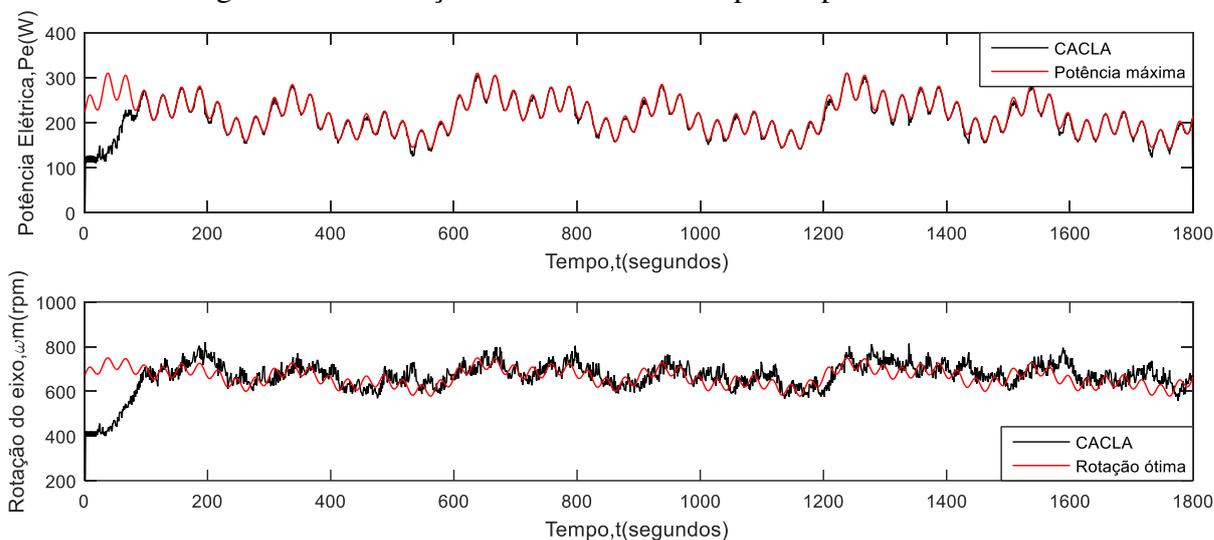


Fonte: desenvolvido pelo autor.

Buscando avaliar a resposta do CACLA frente a perfis com maior variação, apresenta-se os perfis Vento_III e Vento_IV ao mesmo. Avaliando visualmente, nota-se uma melhor resposta ao perfil periódico, o que reflete a capacidade de aprendizagem do algoritmo,

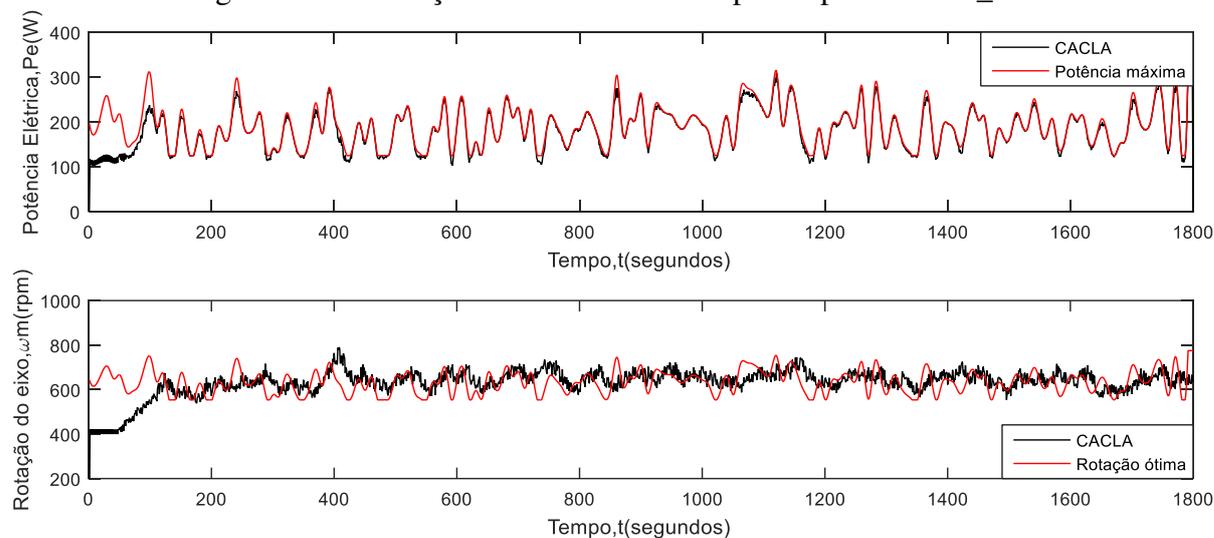
contudo, mesmo no perfil aleatório é possível notar a busca do mesmo pelo ponto de máxima potência.

Figura 90 - Simulação do MPPT CACLA para o perfil Vento_III



Fonte: desenvolvido pelo autor.

Figura 91 - Simulação do MPPT CACLA para o perfil Vento_IV



Fonte: desenvolvido pelo autor.

A fim de realizar uma análise comparativa mais consistente entre os algoritmos, utiliza-se a potência média de operação do sistema, para cada um dos algoritmos e cada um dos quatro perfis de vento propostos. Uma vez que os algoritmos NASPO e CACLA apresentam componentes aleatórias, realizam-se cinco simulações a fim de obter uma estimativa média. Os valores apresentados são percentuais relacionados à potência média máxima extraível do vento.

A Tabela 12 apresenta os dados comparativos entre os algoritmos obtidos via simulação computacional através do software Matlab.

Tabela 12 – Dados comparativos de simulação dos algoritmos implementados

Perfil de vento	Algoritmo	Percentual de Potência elétrica extraída					
		1º Teste	2º Teste	3º Teste	4º Teste	5º Teste	Médio
Vento_I (215,64W)	ω fixo	99,69 %	99,69 %	99,69 %	99,69 %	99,69 %	99,69%
	P&O	97,91 %	97,91 %	97,91 %	97,91 %	97,91 %	97,91%
	NASPO	98,82 %	98,74 %	98,79 %	98,78 %	98,86 %	98,79%
	CACLA	94,76 %	96,55 %	91,70 %	94,36 %	93,89 %	94,36%
Vento_II (219,54W)	ω fixo	93,54 %	93,54 %	93,54 %	93,54 %	93,54 %	93,54%
	P&O	98,19 %	98,19 %	98,19 %	98,19 %	98,19 %	98,19%
	NASPO	98,89 %	98,81 %	98,70 %	98,37 %	98,51 %	98,70%
Vento_III (217,86W)	CACLA	89,56 %	93,83 %	88,22 %	96,93 %	95,07 %	93,83%
	ω fixo	98,60 %	98,60 %	98,60 %	98,60 %	98,60 %	98,60%
	P&O	97,50 %	97,50 %	97,50 %	97,50 %	97,50 %	97,50%
	NASPO	96,80 %	96,92 %	96,93 %	96,67 %	96,93 %	96,92%
Vento_IV (190,14W)	CACLA	92,75 %	91,12 %	91,67 %	90,08 %	93,44 %	91,67%
	ω fixo	97,86 %	97,86 %	97,86 %	97,86 %	97,86 %	97,86%
	P&O	97,11 %	97,11 %	97,11 %	97,11 %	97,11 %	97,11%
	NASPO	93,92 %	93,85 %	93,66 %	93,68 %	93,96 %	93,85%
	CACLA	95,91 %	96,28 %	95,48 %	95,60 %	95,42 %	95,60%

Fonte: desenvolvido pelo autor.

Analisando os resultados apresentados na Tabela 12 faz-se necessário ressaltar certos pontos. Para o perfil Vento_I, o valor de velocidade fixa é calculado de forma a extrair 100% da energia, a queda de energia observada está relacionada com o curto tempo de aceleração do gerador.

Nos perfis de vento Vento_I e Vento_II, com baixa dinâmica de velocidade de vento, o algoritmo NASPO superou os demais, com exceção da operação sob velocidade fixas. Já quando o perfil de vento apresenta uma dinâmica considerável, o algoritmo P&O mostrou-se mais eficaz.

5.5 Validação experimental do algoritmo rastreador de máxima potência

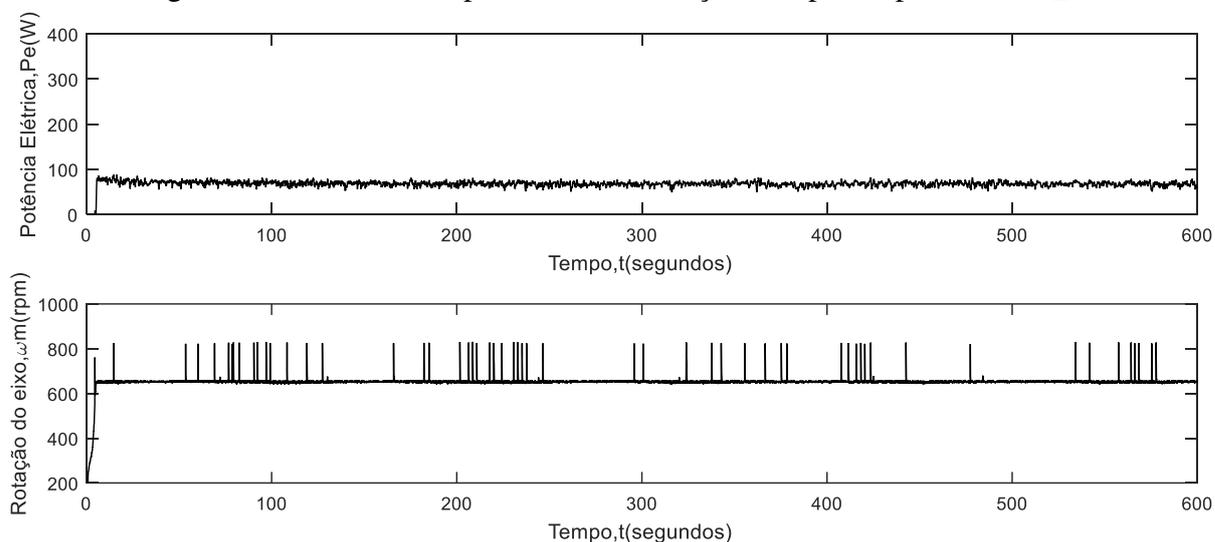
Após realizar a análise dos algoritmos através das simulações computacionais, busca-se obter resultados experimentais que corroborem com os resultados vistos em simulação. Nos subitens seguintes, são apresentados os resultados experimentais dos algoritmos para os diferentes perfis de ventos estabelecidos.

5.5.1 Resultados experimentais para velocidade de rotação fixa

Os ensaios experimentais seguem-se os mesmos passos realizados na etapa de simulação, sendo iniciado pelo algoritmo cuja velocidade de rotação é fixa. Uma vez que a

velocidade de vento emulada é conhecida, mas outros elementos da dinâmica do gerador não, optou-se por definir arbitrariamente, um valor de rotação fixa de 650 rpm.

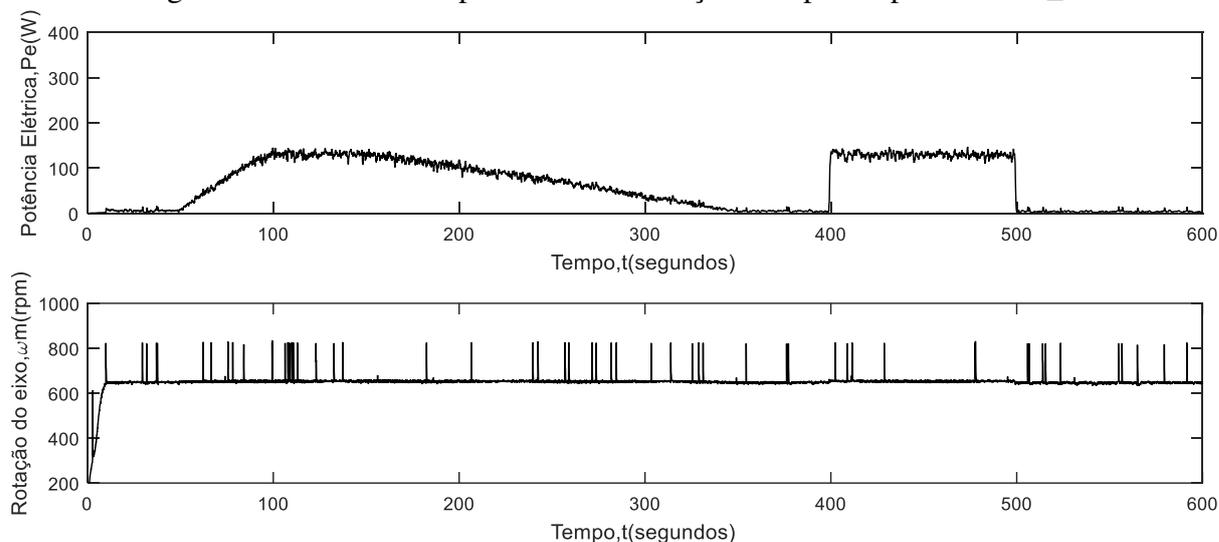
Figura 92 – Resultado experimental de rotação fixa para o perfil Vento_I



Fonte: desenvolvido pelo autor.

A Figura 92 apresenta o valor da potência extraída e da velocidade de rotação do sistema, quando é utilizada a técnica de velocidade de rotação fixa, sendo apresentado o perfil Vento_I. Nota-se diferença percentual de cerca de 52% entre a potência média extraída no experimento e a obtida via simulação, tal diferença se dá pelas perdas mecânicas e elétricas que não são previstas em simulação.

Figura 93 - Resultado experimental de rotação fixa para o perfil Vento_II

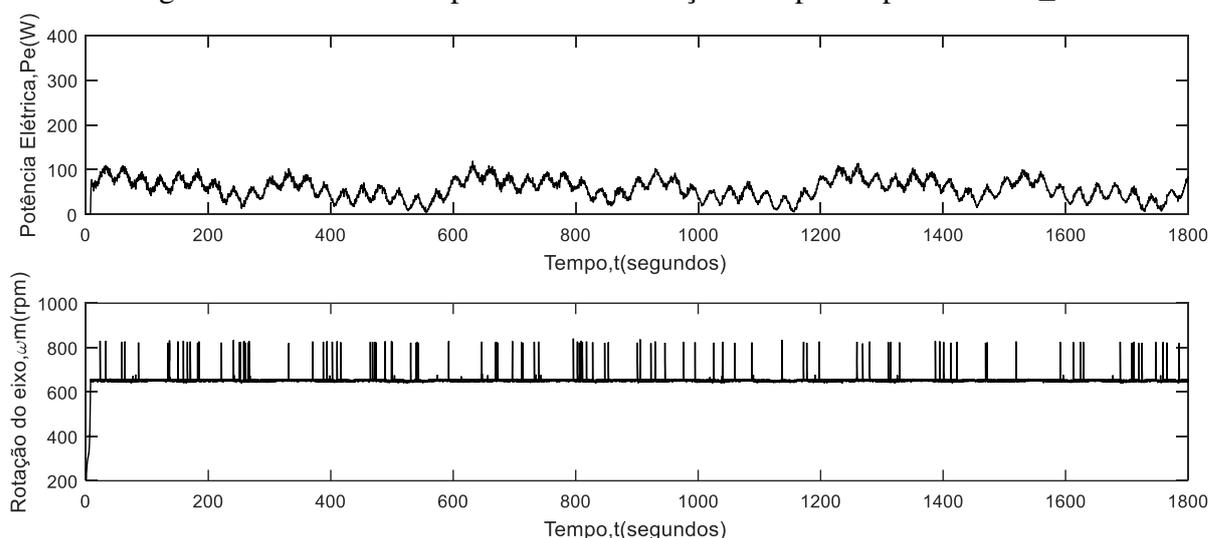


Fonte: desenvolvido pelo autor.

No experimento realizado com velocidade de rotação fixa e o perfil de vento Vento_II, nota-se que a potência extraída apresenta redução percentual, em especial quando há velocidades de vento superiores à média. A Figura 93 apresenta os resultados deste ensaio experimental.

Na Figura 94 e na Figura 95 são apresentados os resultados obtidos quando utilizados os perfis de vento Vento_III e Vento_IV, respectivamente. Comparando tais resultados com o perfil de potência máxima das simulações, nota-se quedas do percentual de potência extraída, o que é problemático especialmente quando há velocidade de vento elevadas, para a qual a potência total é maior, e qualquer percentual de perda, resulta em um valor considerável de energia desperdiçada.

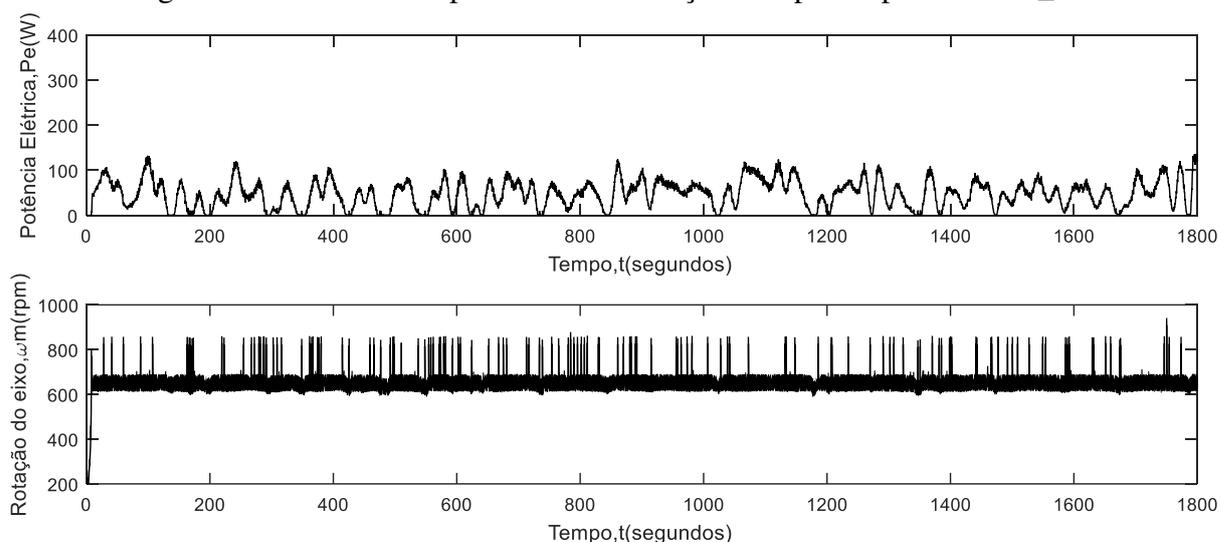
Figura 94 - Resultado experimental de rotação fixa para o perfil Vento_III



Fonte: desenvolvido pelo autor.

Conclui-se que a utilização de velocidade de rotação fixa é uma maneira simples de implementar o sistema de geração, contudo, é possível ver que para diversas situações, em especial aquelas onde há grandes oscilações na velocidade do vento, há uma grande quantidade de energia sendo desperdiçada.

Figura 95 - Resultado experimental de rotação fixa para o perfil Vento_IV



Fonte: desenvolvido pelo autor.

5.5.2 Resultados experimentais do MPPT P&O de referência

Buscando observar a o comportamento de um algoritmo de MPPT clássico, para obter um parâmetro de comparação, realiza-se a implementação experimental do P&O. O tamanho de passo definido anteriormente é adotado com base em experimentos preliminares. A Figura 96 apresenta a resposta deste algoritmo quando apresentado ao perfil de vento Vento_I, que define um valor de vento fixo para o emulador. No resultado experimental, nota-se uma maior oscilação em torno do ponto de máxima potência quando comparado com simulações, isto ocorre devido a ruídos de medições de potência que acarretam em observações errôneas no sentido do gradiente da potência.

Quando submetido ao perfil Vento_II, onde são apresentadas variações discretas e suaves na velocidade do vento, o algoritmo P&O mostra uma resposta dinâmica mais favorável que a utilização de uma velocidade de vento fixa.

Figura 96 - Resultado experimental do P&O para o perfil Vento_I

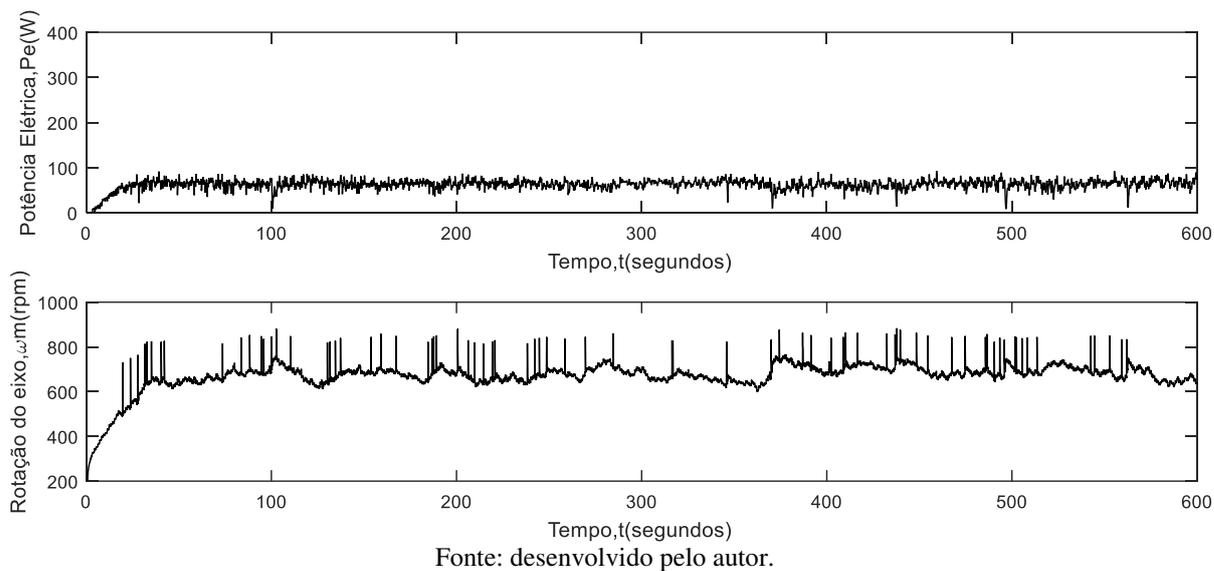
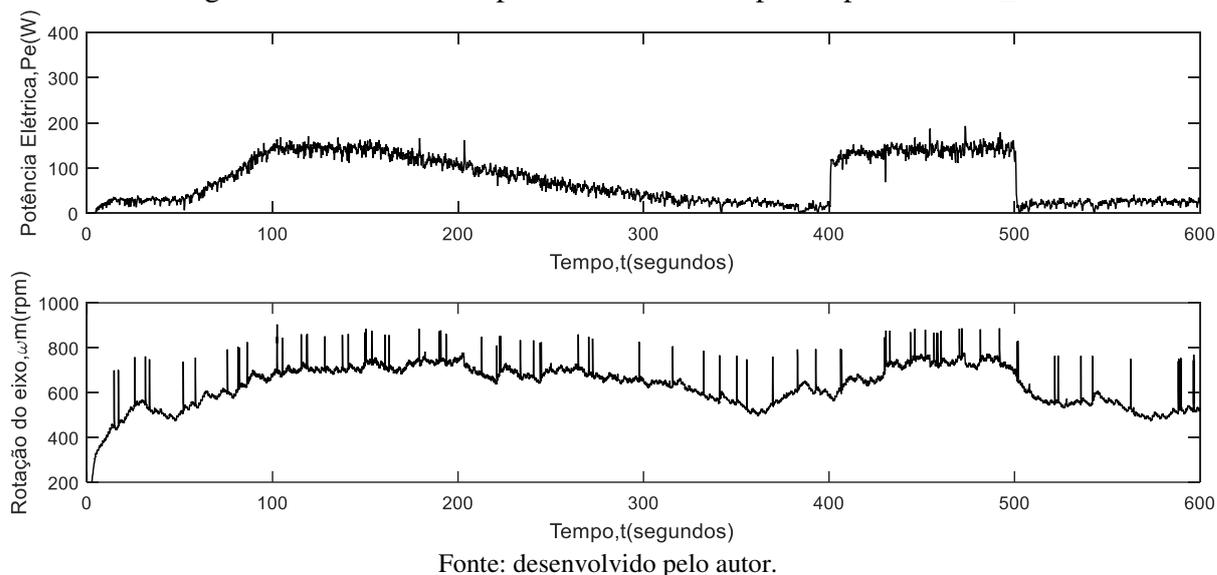
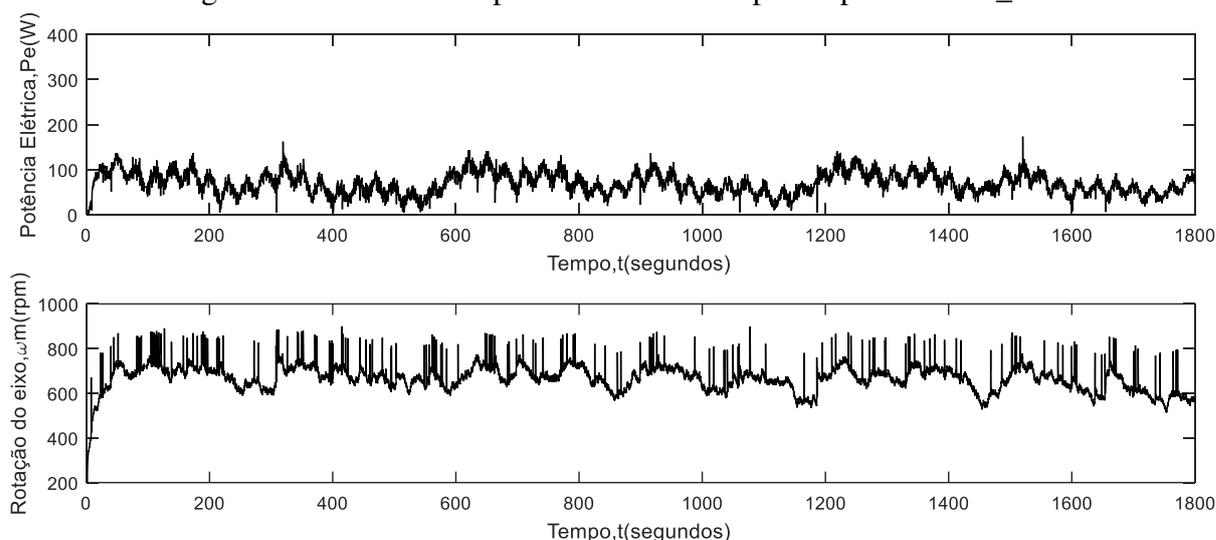


Figura 97 - Resultado experimental do P&O para o perfil Vento_II



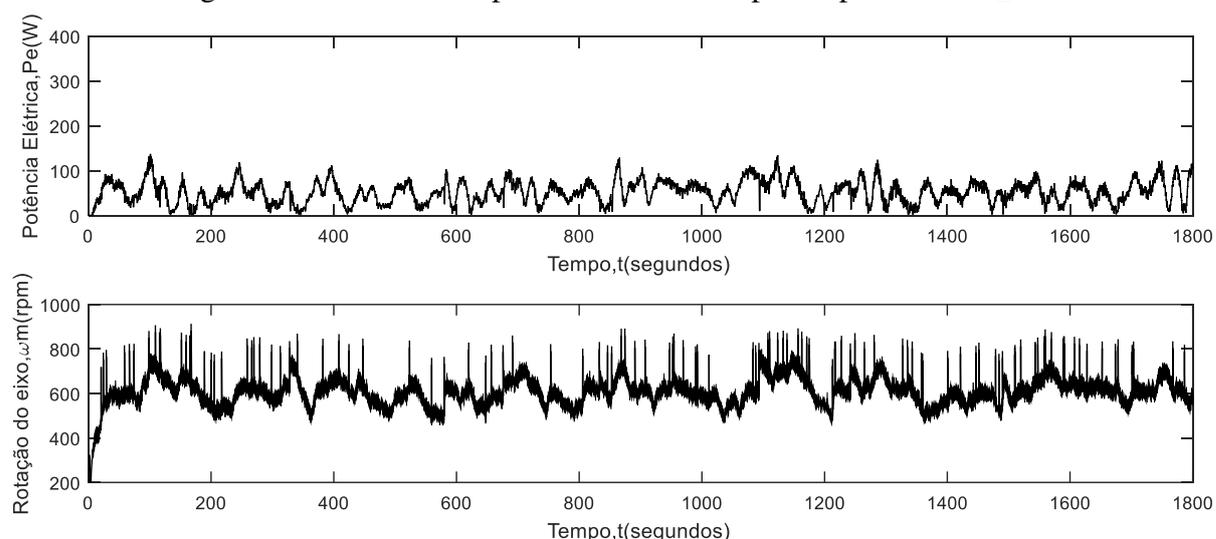
Na Figura 98 e na Figura 99 são apresentados os resultados obtidos pelo P&O quando submetido aos perfis Vento_III e Vento_IV. Visualmente não é possível estabelecer grandes vantagens ante ao modelo de velocidade de rotação fixa. Dessa forma, este e os demais algoritmos tem sua eficiência quantizadas pela potência média extraída para cada perfil de vento.

Figura 98 - Resultado experimental do P&O para o perfil Vento_III



Fonte: desenvolvido pelo autor.

Figura 99 - Resultado experimental do P&O para o perfil Vento_IV

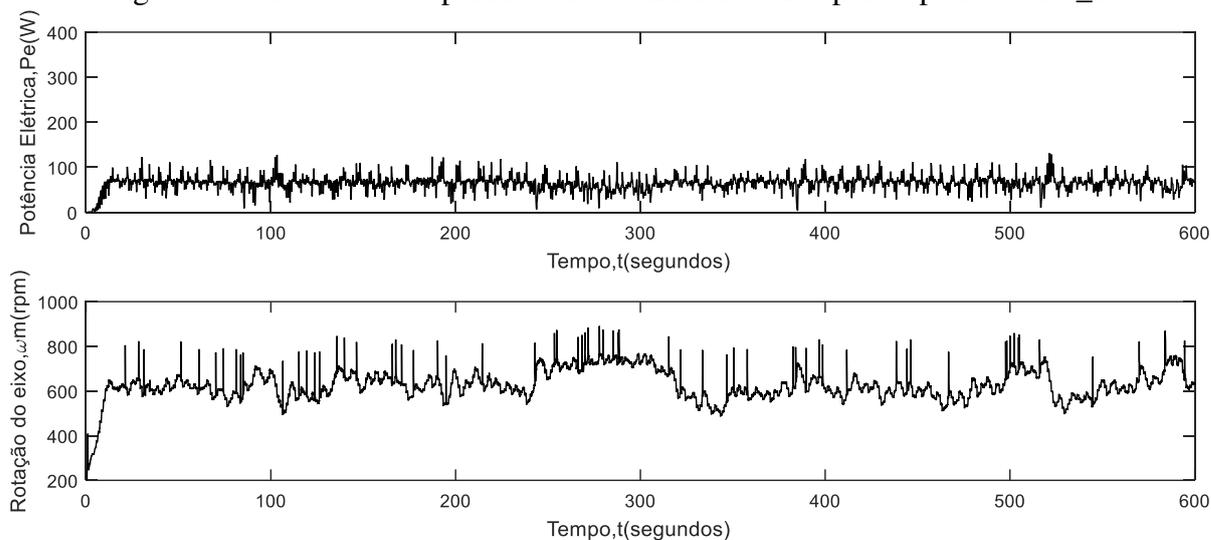


Fonte: desenvolvido pelo autor.

5.5.3 Resultados experimentais do MPPT NASPO

O algoritmo neural NASPO é avaliado experimentalmente para cada uns os perfis de vento também. Para o perfil Vento_I a resposta obtida é similar à do P&O, tendo grande variação em torno do ponto de máxima potência. Vale salientar que o tamanho do passo desse algoritmo é normalizado, mas que seu valor máximo é de 5x o passo do P&O, dando ao mesmo uma capacidade de resposta dinâmica mais robusta. A Figura 100 apresenta os gráficos de potência elétrica extraída e velocidade de rotação do eixo para este ensaio experimental.

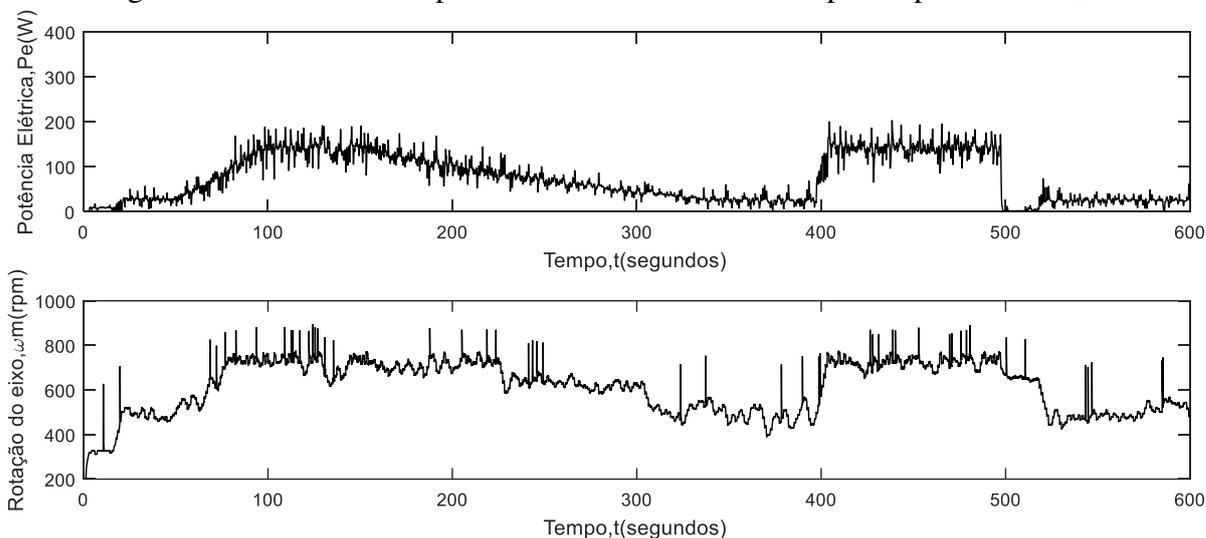
Figura 100 - Resultado experimental do MPPT NASPO para o perfil Vento_I



Fonte: desenvolvido pelo autor.

Na Figura 101 verifica-se os resultados do ensaio feito com o perfil Vento_II. A dinâmica do vento é compensada com o ajuste de velocidade de rotação, assim como no P&O. No gráfico nota-se um perfil um pouco mais bem definido que o P&O, mas apresentando ainda uma oscilação bem acentuada.

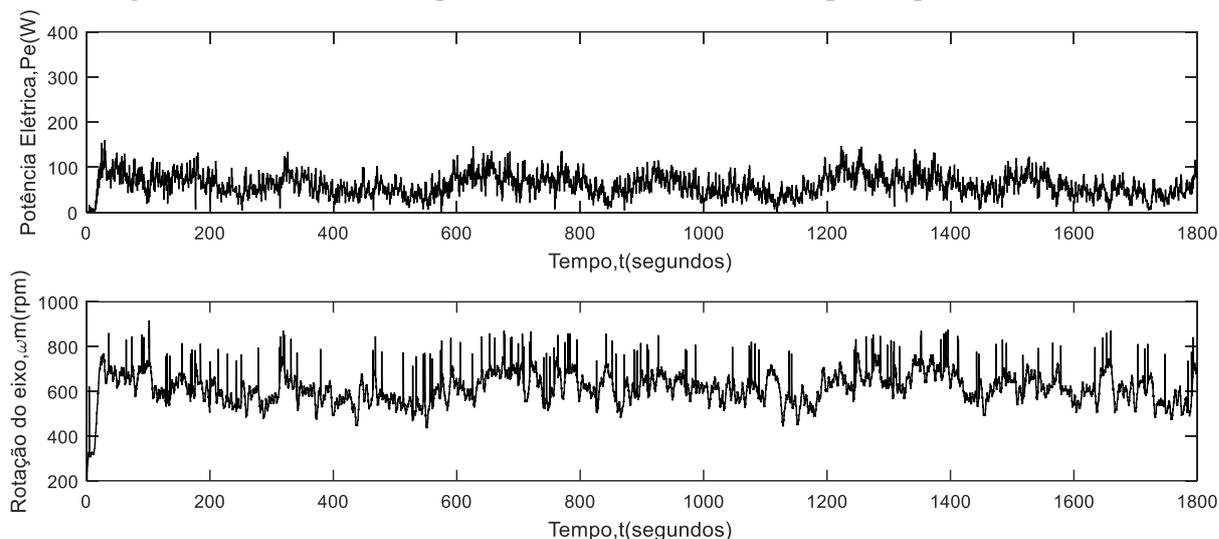
Figura 101 - Resultado experimental do MPPT NASPO para o perfil Vento_II



Fonte: desenvolvido pelo autor.

A Figura 102 e a Figura 103 apresentam os resultados para os perfis de vento Vento_III e Vento_IV, respectivamente. A comparação visual destes resultados é inviável, e, portanto, é feita por meio da quantização da potência extraída média mais a diante.

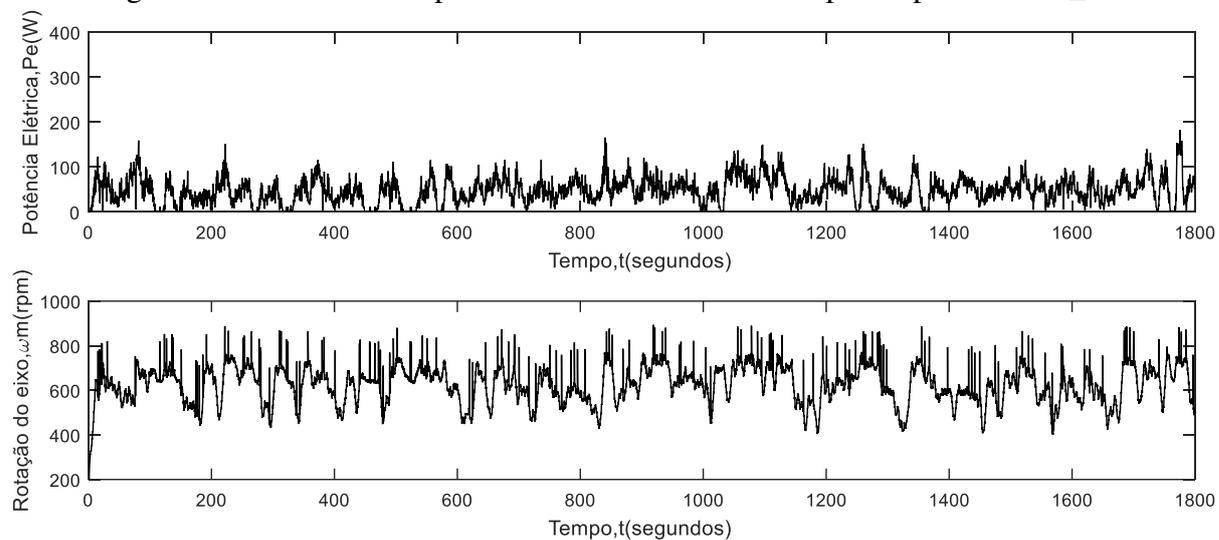
Figura 102 - Resultado experimental do MPPT NASPO para o perfil Vento_III



Fonte: desenvolvido pelo autor.

Os resultados obtidos com o algoritmo NASPO mostram-se promissores, embora ainda apresentem considerável nível de oscilação em torno do perfil ótimo de rotação.

Figura 103 - Resultado experimental do MPPT NASPO para o perfil Vento_IV



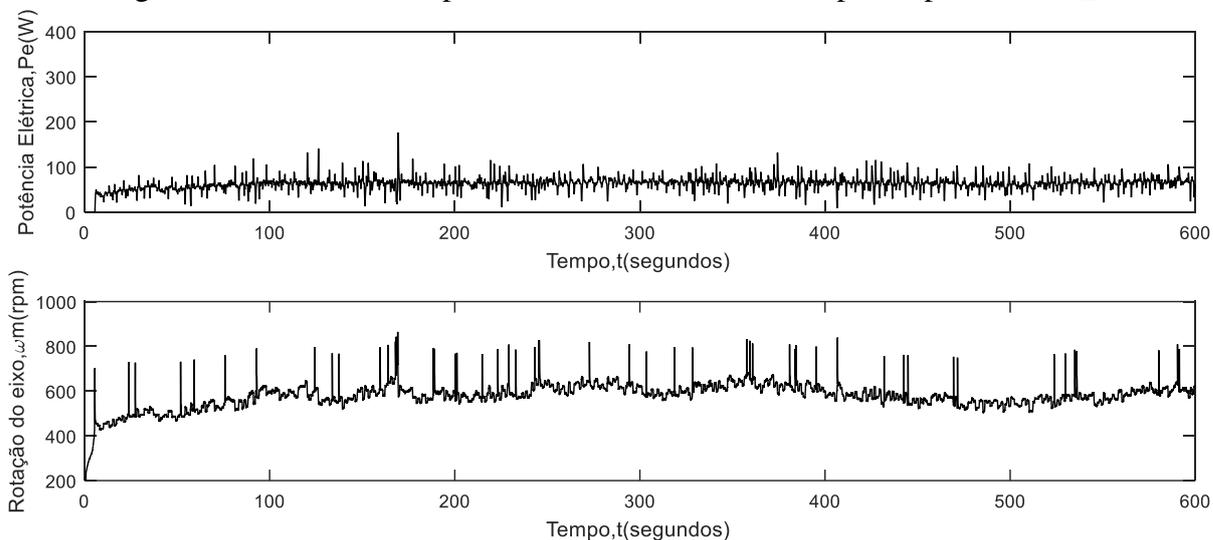
Fonte: desenvolvido pelo autor.

5.5.4 Resultados experimentais do MPPT CACLA

O algoritmo CACLA também foi testado para os quatro perfis de ventos propostos nesse trabalho. Na Figura 104 são apresentados a potência elétrica extraída e a velocidade de rotação do eixo do experimento realizado com o perfil Vento_I. Nota-se que a dinâmica do

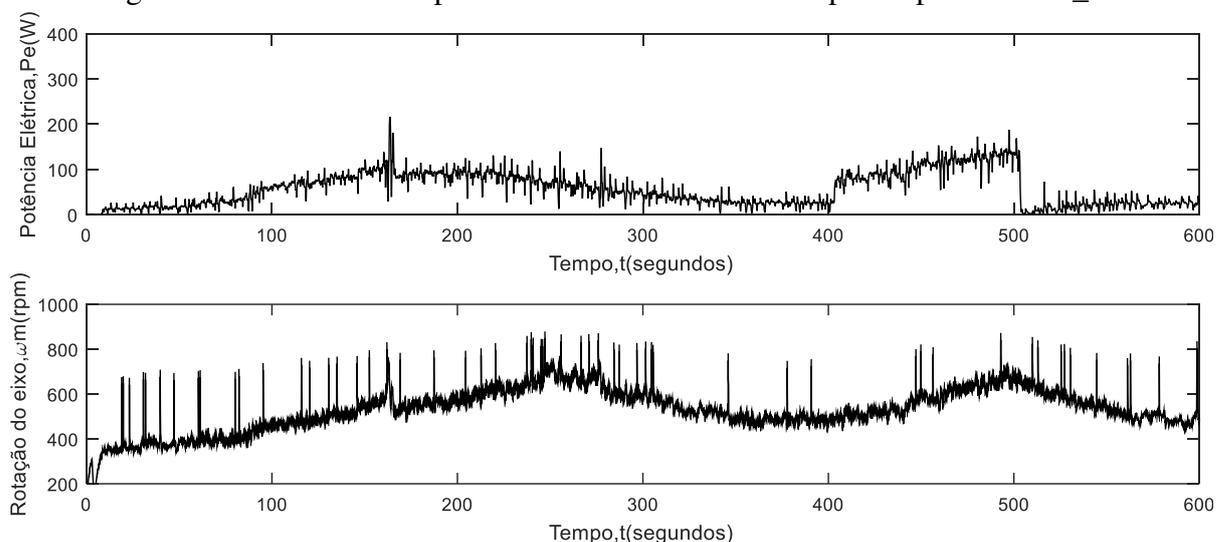
algoritmo é consideravelmente mais lenta que os demais, devido sua característica de aprendizagem por reforço.

Figura 104 - Resultado experimental do MPPT CACLA para o perfil Vento_I



Fonte: desenvolvido pelo autor.

Figura 105 - Resultado experimental do MPPT CACLA para o perfil Vento_II

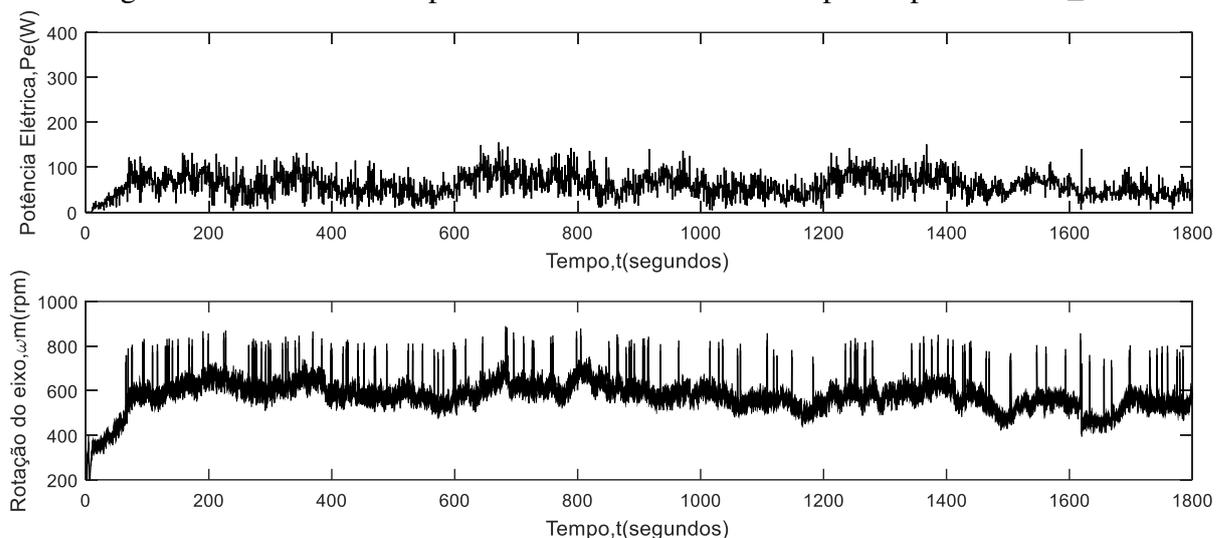


Fonte: desenvolvido pelo autor.

A característica de uma dinâmica mais lenta é vista novamente na Figura 105, onde o perfil Vento_II é utilizado.

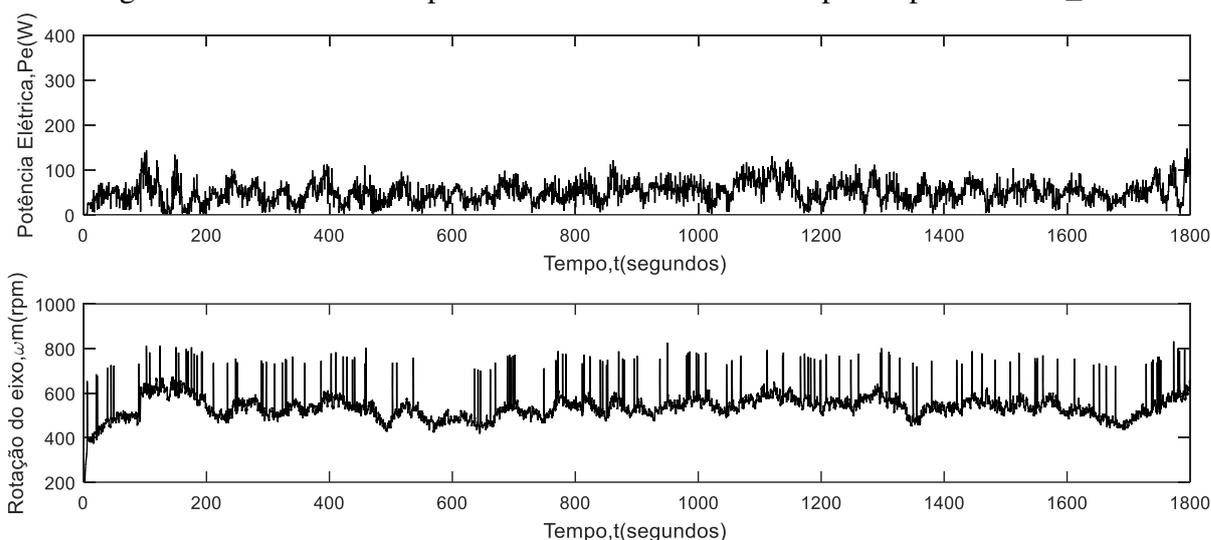
Na Figura 106 e na Figura 107 a resposta do CACLA aos perfis de vento Vento_III e Vento_IV, respectivamente, mostra-se visualmente equiparável aos demais algoritmos, sendo novamente necessária uma avaliação quantitativa que melhor defina a eficiência do mesmo.

Figura 106 - Resultado experimental do MPPT CACLA para o perfil Vento_III



Fonte: desenvolvido pelo autor.

Figura 107 - Resultado experimental do MPPT CACLA para o perfil Vento_IV



Fonte: desenvolvido pelo autor.

5.5.5 Análise experimental comparativa

Como citado anteriormente, devido à aproximação visual entre as respostas dos algoritmos propostos, faz-se necessário realizar uma análise quantitativa que permita analisar melhor as respostas obtidas. Para tal, utilizam-se os valores instantâneo de potência, coletados durante os ensaios experimentais e calcula-se os valores médio de potência para cada um deles.

Com bases nesses valores, montou-se a Tabela 13. Para critério de comparação, adota-se o valor de maior potência média como a norma para cada perfil de vento, e em seguida calcula-se o percentual da mesma que é extraído pelos demais algoritmos.

Tabela 13 – Dados comparativos entre os algoritmos e perfis de vento

Controle	Vento_I	Vento_II	Vento_III	Vento_IV
ω fixo	67,77 W (100,0%)	69,55 W (93,6%)	58,64 W (83,8%)	47,69 W (91,4%)
P&O	62,19 W (91,8%)	74,34 W (100,0%)	69,96 W (100,0%)	52,15 W (100,0%)
NASPO	64,88 W (95,7%)	72,92 W (98,1%)	62,19 W (88,9%)	47,76 W (91,6%)
CACLA	62,18 W (91,8%)	55,05 W (74,1%)	60,79 W (86,9%)	49,36 W (94,7%)

Fonte: desenvolvido pelo autor.

Através dos resultados apresentados, nota-se que de fato, o algoritmo P&O ainda apresenta valores de potência extraída maiores que os demais algoritmos. A justificativa para tal, reside no fato dos algoritmos NASPO e CACLA basearem-se em aprendizagem, e inicialmente terem um comportamento mais ingênuo. O valor de potência extraída para velocidade de rotação fixa no perfil Vento_I é esperada, uma vez que a velocidade adotada é fixada bem próxima ao ponto ótimo, e não apresenta a oscilação inerente dos demais algoritmos, mas não serve de indicativo de vantagem da mesma, uma vez que o perfil de comportamento do vento aproxima-se muito mais daquilo que é visto no perfil Vento_IV.

6 CONCLUSÃO

Através dos estudos bibliográficos realizados, foi possível estabelecer uma proposta diferenciada ante a problemática de rastreamento de máxima potência aplicado à geradores eólicos. Os algoritmos propostos têm fundamentação a partir de outros estudos realizado. O NASPO parte de uma modificação do P&O convencional, buscando inserir a capacidade de aprendizado inerente a redes neurais. Já o CACLA é um algoritmo de aprendizagem por reforço complexo, que ainda não havia sido aplicado à problemática de rastreamento de máxima potência. Ambos se mostram, dessa forma, inovadores dentro do que se propõem.

O projeto, desenvolvimento e validação da bancada experimental de emulação de uma turbina eólica mostrou-se crítico para a conclusão deste trabalho. Além do desenvolvimento sistema de acionamento baseado em conversores chaveados, existiu a necessidade de desenvolver o controle de forma que o motor se comportasse como uma turbina submetida ao vento. A estrutura mecânica do motor e sua respectiva não idealidade dentro desse sistema também se mostrou problemática.

Através da bancada emuladora desenvolvida se fez possível realizar os ensaios experimentais que validam o comportamento dos algoritmos propostos, o que permitiu uma análise mais minuciosa, que inclusive resultou em divergência entre os resultados de simulação e prático.

Os algoritmos propostos nesse trabalho, mostram-se capazes de realizar o rastreamento de máxima potência, contudo, não se mostraram capazes de superar algoritmos mais simples como o P&O. Como pontos que levaram a esses resultados destacam-se: o treinamento de redes neurais e algoritmos de aprendizagem por reforço são lentos, acelerar esse processo, acaba resultando em falhas; as dinâmicas de ventos proposta combinado com a inércia do sistema, não permitiram que os algoritmos tivessem respostas e recebessem estímulos para valores de ventos iguais, o que acaba alterando o estado, antes que fosse possível obter informação de aprendizado do mesmo.

Como proposta de trabalhos futuros, aponta-se a análise de custo computacional dos algoritmos propostos. Além disso, desenvolver o sistema inteligente em uma plataforma embarcada pode vir a permitir a utilização de uma malha de controle de torque, em detrimento do controle da malha de rotação. A utilização de controle de torque ótimo para junto aos algoritmos inteligentes, permitiria que os mesmos tivessem um gama de ações/reações muito mais elevada, haja vista o tempo necessário entre interações.

7 REFERÊNCIAS BIBLIOGRÁFICAS

- ABREU, M. C. S. DE et al. Fatores determinantes para o avanço da energia eólica no estado do ceará frente aos desafios das mudanças climáticas. **Eletrônica de Administração**, v. 78, n. 2, p. 274–304, 2014.
- ANEEL. **Resolução Normativa Nº 547**, 2013.
- BARBI, I. **Eletrônica de Potência**. 6. ed. Florianópolis: [s.n.].
- BARBI, I.; MARTINS, D. C. **Conversores CC-CC básicos não isolados**. 2. ed. Florianópolis: Edição do Autor, 2006.
- BARBOSA, A. U. **DESENVOLVIMENTO DE UM EMULADOR DE TURBINA EÓLICA** Fortaleza Universidade Federal do Ceará, , 2015a.
- BARBOSA, A. U. **DESENVOLVIMENTO DE UM EMULADOR DE TURBINA EÓLICA. The effects of brief mindfulness intervention on acute pain experience: An examination of individual difference**, v. 1, p. 1689–1699, 2015b.
- BHANDARE, A. M.; BANDEKAR, P. J.; MANE, S. S. **Wind energy maximum power extraction algorithms: A review**. 2013 International Conference on Energy Efficient Technologies for Sustainability. **Anais...IEEE**, abr. 2013
- BURTON, T. et al. **Wind Energy Handbook**. London: Wiley, 2001.
- BUSO, S.; MATTAVELLI, P. **Digital control in power electronics**. 1. ed. [s.l: s.n.].
- BUSO, S.; MATTAVELLI, P. Digital Control in Power Electronics. **Synthesis Lectures on Power Electronics**, v. 1, n. 1, p. 1–158, jan. 2006b.
- DALALA, Z. M. et al. Design and Analysis of an MPPT Technique for Small-Scale Wind Energy Conversion Systems. **IEEE Transactions on Energy Conversion**, v. 28, n. 3, p. 756–767, set. 2013.
- DALALA, Z. M.; ZAHID, Z. U.; LAI, J.-S. New Overall Control Strategy for Small-Scale WECS in MPPT and Stall Regions With Mode Transfer Control. **IEEE Transactions on Energy Conversion**, v. 28, n. 4, p. 1082–1092, dez. 2013.
- FITZGERALD, A. E.; KINGSLEY JR., C.; UMANS, S. D. M. **Máquinas Elétricas**. 6. ed. [s.l: s.n.].
- FREITAS, G. S.; DATHEIN, R. AS ENERGIAS RENOVÁVEIS NO BRASIL : uma avaliação acerca das implicações para o desenvolvimento socioeconômico e ambiental. **NEXos Econômicos**, v. 7, p. 71–94, 2013.
- GONDIM, F. **Neuropsicofisiologia introdução as neurociencias do comportamento humano**. 2. ed. Fortaleza: Premius, 2013.
- HARDY, T.; JEWELL, W. **Emulation of a 1.5MW wind turbine with a DC motor**. 2011 IEEE Power and Energy Society General Meeting. **Anais...IEEE**, jul. 2011
- HASSELT, V.; WIERING, H.; A., M. **Reinforcement Learning in Continuous Action Spaces**. 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning. **Anais...Utrecht: IEEE**, abr. 2007
- HAYKIN, S. **Neural networks: a comprehensive foundation**. 2. ed. Ontario: Pearson, 1999. v. 13
- HAYKIN, S. **Neural networks and learning machines**. 3. ed. Ontario: Pearson, 2009.
- HEIER, S.; WADDINGTON, R. **Grid Integration of Wind Energy Conversion Systems**. [s.l.] Wiley, 2006a.
- HEIER, S.; WADDINGTON, R. **Grid Integration of Wind Energy Conversion Systems**. [s.l.] Wiley, 2006b.
- JUNIOR, J. C. G. F.; RODRIGUES, M. G. Um Estudo Sobre a Energia Eólica No Brasil. **Ciência Atual**, v. 5, n. 1, p. 02–13, 2015.
- KOLTER, J. Z.; JACKOWSKI, Z.; TEDRAKE, R. **Design, analysis, and learning control of a fully actuated micro wind turbine**. 2012 American Control Conference (ACC).

Anais...IEEE, jun. 2012

KOUADRIA, S. et al. **Development of real time wind turbine emulator based on DC motor controlled by PI regulator**. 2013 Eighth International Conference and Exhibition on Ecological Vehicles and Renewable Energies (EVER). **Anais...Monte Carlo: IEEE**, mar. 2013

LALOUNI, S. et al. An improved MPPT algorithm for wind energy conversion system. **Journal of Electrical Systems**, v. 10, n. 4, p. 484–494, 2014.

LOPES, I. L.; PINHEIRO, C. A. M.; SANTOS, F. A. O. **Inteligência Artificial**. 1. ed. Rio de Janeiro: Elsevier, 2014.

MACHADO, I. R. **Sistema Eólico para Carregamento de Baterias**. Fortaleza: Universidade Federal do Ceará, 2007.

MACHELIS, C. **Influence of the measurement accuracy of wind sensors on wind system performance**. 2015 International Conference on Clean Electrical Power (ICCEP). **Anais...IEEE**, jun. 2015

MAHDY, A.; EL-HAKIM, S. M.; HANAFY, H. H. **Small wind turbine emulator with separately excited DC motor using analog electronic circuit**. IET Conference on Renewable Power Generation (RPG 2011). **Anais...IET**, 2011

MINISTERIO DE MINAS E ENERGIA, B. **Resenha Energética Brasileira - Exercício de 2014** Ministério de Minas e Energia. [s.l: s.n.].

MOHAN, N. **Electric machines and drives: a first course**. Minneapolis, USA: Wiley, 2012.

MOREIRA, M. B.; MEDEIROS, C. A. DE. **Princípios básicos de análise do comportamento**. Porto Alegre: Artmed, 2007.

NASCIMENTO, F. S. DO; STIVAL, G. L.; FONSECA, M. S. P. DA. **Estudo de viabilidade de desenvolvimento de um gerador eólico a partir de componentes de mercado**. [s.l.] Universidade Tecnológica Federal do Paraná, 2013.

NETO, T. R. F. **Short Primary Linear Drive Designed for Synchronous and Induction Operation Mode with On-Board Energy Storage**. [s.l.] Technischen Universität Darmstadt, 2012.

PATEL, V. et al. **A review: Maximum power extraction method in wind turbine system using different algorithm**. 2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO). **Anais...IEEE**, jan. 2015

SILVA, R. F. **Emulação de uma turbina eólica e controle vetorial do gerador de indução rotor gaiola de esquilo para um sistema eólico**. [s.l.] Universidade Federal do Rio de Janeiro, 2012.

SUBUDHI, B.; PRADHAN, R. A Comparative Study on Maximum Power Point Tracking Techniques for Photovoltaic Power Systems. **IEEE Transactions on Sustainable Energy**, v. 4, n. 1, p. 89–98, jan. 2013.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. 2. ed. London: Bradford, 2012.

TAVEIROS, F. E. V.; BARROS, L. S.; BEZERRA COSTA, F. **Wind turbine torque-speed feature emulator using a DC motor**. 2013 Brazilian Power Electronics Conference. **Anais...Gramado: IEEE**, out. 2013

VORPERIAN, V. Simplified Analysis of Pwm Converters Using Model of Pwm Switch Part I: Continuous Conduction Mode. **IEEE Transactions on Aerospace and Electronic Systems**, v. 26, n. 3, p. 490–496, 1990.

WATTES, J. L. et al. **A neural adaptative step size method for maximum power point tracking in low power wind turbine systems**. 2015 IEEE 13th Brazilian Power Electronics Conference and 1st Southern Power Electronics Conference (COBEP/SPEC). **Anais...Fortaleza: IEEE**, nov. 2015

WATTES, J. L. et al. **A Continuous Actor-Critic Maximum Power Point Tracker Applied to Low Power Wind Turbine Systems**. The Applied Power Electronics Conference and

Exposition 2016. **Anais...**Long Beach: IEEE, 2016

APÊNDICE A: ENSAIO DO MOTOR CC

1. Ensaio de parâmetros elétricos

Os principais parâmetros elétricos de motores de corrente-continua são as resistências e indutâncias dos enrolamentos de campo e de armadura. A medição das resistências é feita através de multímetros, uma vez que se visa a medição da resistência vista por uma componente contínua de corrente. Já as indutâncias são coletadas através de medidores de impedância configurados para as frequências em que os enrolamentos irão operar. Todas as medições devem ser realizadas para diferentes posições do eixo do motor, a fim de evitar leituras errôneas devido ao posicionamento das escovas do mesmo.

- R_{armadura} (multímetro): 1,1 Ω .

- R_{campo} (multímetro): 337 Ω .

- L_{armadura} (medidor de impedância, 20KHz): 6 mH.

- L_{campo} : 13,2 H.

2. Ensaio de parâmetros mecânicos

Os principais parâmetros mecânicos do motor de corrente-continua são os coeficientes de atrito dinâmico e estático e a inércia mecânica do sistema.

a) *Coefficiente de atrito dinâmico*

O ensaio para obtenção do coeficiente de atrito estático é feito com o sistema a vazio, de forma que não haja transmissão de energia mecânica do motor para outras cargas conectadas ao seu eixo. Dessa forma, conectam-se os terminais de campo do motor em tensão nominal e, utilizando um variômetro, alimenta-se os terminais da armadura do motor. Uma vez que a medição é feita via tacômetro mecânico, faz-se necessário definir o perímetro da superfície do eixo com a qual este estará em contato.

-Raio = 0,108 m.

-Perímetro = $2 \cdot \pi \cdot \text{Raio} = 0,6786$ m.

A seguir, para valores distintos de velocidade de rotação, medem-se as 3 variáveis a seguir:

- V_{armadura} : através de voltímetro, em V.

- I_{armadura} = através de tacômetro, em A.

- ω : através de tacômetro mecânico, em m/min.

Calcula-se então a velocidade de rotação em rad/s:

- ω [rad/s] = ω [m/min] / (60 x Raio).

A fim de obter o valor da potência perdida pelo atrito dinâmico, calcula-se a potência total entregue ao moto, e a potência perdida da condução de corrente pelos enrolamentos:

$$-P_{\text{total}} = I_{\text{armadura}} \times V_{\text{armadura}} \text{ [W]}.$$

$$-P_{\text{perdas-elétricas}} = R_{\text{armadura}} \times I_{\text{armadura}}^2 \text{ [W]}.$$

Dessa forma, o valor das perdas mecânicas é dado pela diferença entre as duas.

$$-P_{\text{perdas-mecânicas}} = P_{\text{total}} - P_{\text{perdas-elétricas}} \text{ [W]}.$$

Sabendo da potência perdida pelo atrito dinâmico, é possível encontrar o coeficiente do mesmo através da seguinte relação.

$$B_{\text{dinâmico}} = P_{\text{perdas-mecânicas}} / \omega^2 \text{ [N.m / (rad/s)]}.$$

Os valores medidos e calculados nos ensaios realizados são apresentados na tabela.

ω_{ref} [m/min]	ω [m/min]	V_{armadura} [V]	I_{armadura} [A]	ω [rad/s]	$P_{\text{elétrica}}$ [W]	$P_{\text{perdas-elétricas}}$ [W]	$P_{\text{perdas-mecânicas}}$ [W]	$B_{\text{dinâmico}}$ [N.m/(rad/s)]
136	126,00	21,00	1,11	19,81	23,31	1,36	21,95	0,055937
271	260,00	43,60	1,66	40,88	72,38	3,03	69,34	0,041494
407	404,00	67,90	2,28	63,52	154,81	5,72	149,09	0,036950
543	550,50	95,10	2,96	86,56	281,50	9,64	271,86	0,036286
679	665,00	118,00	3,58	104,56	422,44	14,10	408,34	0,037350

Adota-se portanto o valor médio dos coeficientes calculados, dado por 0,041603 Nm/(rad/s).

b) Torque de atrito estático

O atrito estático é dado pelo torque resistente visto em um corpo em repouso que é submetido a uma dada tração. Para mensurar tal parâmetro, utiliza-se uma balança portátil, e mede-se limite de peso aplicado no eixo necessário para que haja rotação.

$$-\text{Massa} = 0,50 \text{ kg}.$$

$$-\text{Peso} = 9,807 \times \text{Massa} = 4,413 \text{ N}.$$

$$T_{\text{estático}} = \text{Peso} \times \text{Raio} = 0,530 \text{ N.m}.$$

c) Ensaio de Inércia mecânica

A inércia mecânica, diferente do torque de atrito é a característica que relaciona o atraso entre a aplicação de uma força, e o início de rotação, por mais que ela supere o torque de atrito estático. A forma mais simples de calcular a inércia de um sistema é realizar a medição da constante de tempo mecânica ($\tau_{\text{mecânico}}$). Para isso, acelera-se a máquina a uma velocidade

conhecida e desliga-se a alimentação da mesma, observa-se então, o tempo necessário para que velocidade decaia a 1/3 da velocidade inicial, esse tempo é classificado como $\tau_{mecânico}$.

$$\tau_{mecânico} = 1 \text{ s.}$$

De posse do valor de $\tau_{mecânico}$ é possível calcular a inércia através da relação:

$$J = B_{dinâmico} \times \tau_{mecânico} = 0,041603 \text{ Kg.m}^2.$$

3. Ensaio de parâmetros eletromecânicos

a) Coeficiente de torque/corrente do motor de corrente-continua

Em motores de corrente continua, o torque empenhado pela máquina no eixo é proporcional à corrente que flui pelos enrolamentos da armadura da máquina, dessa forma, o seguinte ensaio visa obter o valor de K_T que relaciona a corrente de armadura com o torque mecânico. Os valores medidos no ensaio de atrito dinâmico podem ser utilizados uma vez que o torque mecânico pode ser calculado da seguinte forma:

$$-T_{mecânico} = P_{perdas-mecânicas} / \omega \text{ [N.m]}.$$

E uma vez que se tenha o torque mecânico e a corrente de armadura foi mensurada, o coeficiente K_T é dado por:

$$-K_T = T_{mecânico} / I_{armadura} \text{ [N.m / A]}.$$

ω_{ref} [m/min]	ω [m/min]	$V_{armadura}$ [V]	$I_{armadura}$ [A]	ω [rad/s]	$P_{perdas-mecânicas}$ [W]	$T_{mecânico}$ [N.m]	K_T [N.m/A]
136	126,00	21,00	1,11	19,81	21,95	1,11	0,9984
271	260,00	43,60	1,66	40,88	69,34	1,70	1,0219
407	404,00	67,90	2,28	63,52	149,09	2,35	1,0294
543	550,50	95,10	2,96	86,56	271,86	3,14	1,0611
679	665,00	118,00	3,58	104,56	408,34	3,91	1,0909

Através de uma média simples, define-se que o valor de K_T é 1,0403 Nm/A.

b) Coeficiente de Rotação/Tensão do motor de corrente-continua

Assim como em um motor de corrente-continua o torque se relaciona com a corrente, a tensão induzida relaciona-se com a velocidade de rotação do mesmo. Para definir essa relação, utiliza-se o seguinte equacionamento:

$$-K\omega = (V_{armadura} - I_{armadura} \times R_{armadura}) / \omega \text{ [V/(rad/s)]}.$$

ω_{ref} [m/min]	ω [m/min]	$V_{armadura}$ [V]	$I_{armadura}$ [A]	ω [rad/s]	$K\omega$ [V/(rad/s)]
136	126,00	21,00	1,11	19,81	0,9984
271	260,00	43,60	1,66	40,88	1,0219
407	404,00	67,90	2,28	63,52	1,0294
543	550,50	95,10	2,96	86,56	1,0611
679	665,00	118,00	3,58	104,56	1,0909

Assim, através dos dados coletados é possível definir $K\omega$ como igual a 1,0403.

APÊNDICE B: CÓDIGOS MPLAB DO CHOPPER

Índice do projeto WTE

- global.h
- controle.s
- event.c
- init.c
- main.c

global.h (WTE)
<pre> *****WTE***** #include "p30F2020.h" #include <libpic30.h> #include <string.h> #include <stdio.h> #include <stdint.h> #include <dsp.h> // #define MIPS 20 #define Fosc 10000000 // 10 MHz #define Fpll Fosc*32 // 320 MHz #define Fadc Fpll/2 // 160 MHz #define Fcy Fpll/16 // 20 MHz #define Tcy 1/Fcy //Tcy=1/Fcy // #define Fpwm 20000 //20kHz #define ptper ((Fpll*2)/Fpwm) //Up: 32000 #define spec_ev_pos 8 #define spec_ev1 ((ptper/4)/spec_ev_pos) //Special event em 90°- 1000 #define spec_ev2 (((ptper/4)*3)/spec_ev_pos) //Special event em 270°- 3000 #define max_duty (950) //ptper //Maximo duty cycle 100% #define min_duty (50) //Minimo duty cycle 0% #define duty_70 (ptper)*0.7 //duty cycle de 70% #define base_corrente 1310.7 // base Q15 #define Corrente_500mA base_corrente*0.5 #define Corrente_600mA base_corrente*0.6 #define Corrente_700mA base_corrente*0.7 #define Corrente_1A 1*base_corrente #define Corrente_1500mA 1966 //1.5*base_corrente #define Corrente_2A 2621 //2*base_corrente #define Corrente_2250mA 2949 //2.25*base_corrente #define Corrente_2500mA 3277 //2.5*base_corrente #define Corrente_3A 3932 //3*base_corrente #define Corrente_4A 5243 //4*base_corrente #define Corrente_4500mA 5898 //4.5*base_corrente #define Corrente_5A 5*base_corrente #define Corrente_6A 6*base_corrente #define Corrente_9A 11796 //9*base_corrente #define offset_sensor (32768) //Off-set do sensor de corrente (2.50V) // #define STOPPER 0 // Smaller than any datum #define MEDIAN_FILTER_SIZE (50) // #define _ISRNPVSV__attribute__((interrupt, no_auto_psv)) #define Tcy_nano 51.54 // 1/Fcy (nanosegundos) //#define Tcy 135.69 #define cyc 500/Tcy_nano #define ON 1 #define OFF 0 #define EN 1 #define DIS 0 </pre>

```

global.h (WTE)
#define DEFAULT_ 0
#define AR 10
#define debounce (50U) // em milissegundos
#define taxa_atualizacao (300U) // in mileseconds
#define f_base 1000
#define m_base f_base/1000
//
//----->Prioridades de Interrupções<-----
#define PRI_PWM 1
#define PRI_TX 2
#define PRI_RX 2
#define PRI_QEI 2
#define PRI_TMR1 4
#define PRI_TMR2 1
#define PRI_TMR3 1
#define PRI_AD 6
//
//----->BOTÕES E LEDS<-----
#define ler_botao1 PORTAbits.RA9
#define ler_botao2 PORTEbits.RE7
#define numero_de_botoes 2
#define incrementa 2
#define decrementa 1
#define led0 LATEbits.LATE6 // direita
//#define led1 LATCbits.LATC15 // esquerda -> reservado para oscilador
#define STATE_IDLE 0
#define STATE_SINGLE_1 1
#define STATE_LONG_1 2
#define STATE_SINGLE_2 3
#define STATE_LONG_2 4
#define STATE_RELEASE 5
//
// ----->QEI<-----
#define numero_de_pulsos_por_volta 10000
#define RPM_nominal 1800 //rpm
//
/*****PROTOTIPOS DE
FUNÇÕES*****/
extern void iniciarPIC(void);
extern void iniciarADC(void);
extern void iniciarPWM(void);
extern void iniciarTMR1(void);
extern void iniciarTMR2(void);
extern void iniciarTMR3(long int frequencia, long int prescale, int start);
extern void iniciarCN(void);
extern void delay_ms(long int x);
extern void delay_us(long int x);
extern void clrPin(void);
extern void tglPin(void);
extern void setPin(void);
extern void control_pi_corrente(void);
extern volatile uint16_t LUT_wind(uint16_t pec);
extern volatile uint16_t LUT_cp(uint16_t pec);
//
/*****VARIÁVEIS GLOBAIS*****/
/* X space 0x0800 a 0x08FF
 * Y space 0x0900 a 0x09FF
 * 30F2010 exclusivo!
 */
//Controle PI
extern volatile int_fast16_t _XDATA(16) PI_CORRENTE[3];
extern volatile int_fast16_t _YDATA(16) CONTROL_CORRENTE[3]; // e(k)->0,e(k-1)->1,u(k-1)->2
extern volatile int_fast16_t OUTPUT_CORRENTE;

```

global.h (WTE)

```
extern volatile int_fast16_t MIN_DUTY, MAX_DUTY;
//Leitura AD
extern volatile int_fast16_t Ia, Ia1, Ia2;
extern volatile uint_fast16_t adconvert;
//Variáveis de Emulação
extern volatile uint_fast16_t cont;
extern volatile uint_fast16_t cont_vento;
extern volatile uint_fast16_t vento;
extern volatile uint_fast32_t rpm;
extern volatile uint_fast16_t pulsos;
extern const uint_fast32_t k_omega;
extern volatile uint_fast32_t omega;
extern volatile uint_fast32_t lambda;
extern volatile uint_fast16_t Cp;
extern volatile uint_fast16_t vento_cubo;
extern volatile uint_fast32_t Pm_ref;
extern volatile uint_fast16_t Tm_ref;
extern volatile int_fast16_t Ia_ref;
//Gerais
extern unsigned int dj, dnop;
extern volatile uint_fast24_t tmr3_cont;
extern volatile uint_fast8_t press;
extern volatile uint_fast16_t voltas;
extern volatile uint_fast16_t timer_buff, refresh, pulsos_buffer;
//
```

controle.s (WTE)

```
;*****WTE*****
; Select the device family
.equ __dsPIC30F__, 1
;.equ __PIC24H__, 1
;
.ifdef __dsPIC30F__
.include "p30fxxxx.inc"
.endif
;
.global _control_pi_corrente; declarando como label global
"control_PI_corrente"
.global _emulador
.global _setPin
.global _clrPin
.global _tglPin
;
; Toggle RC14
_tglPin:BTG LATE,#6
return
;
; Set RC14
_setPin:BSET      LATE,#6
return
;
; Clear RC14
_clrPin:BCLR      LATE,#6
return
;
_control_pi_corrente:  ; início da função "control_PI_corrente"
                      ; (20 ciclos de clock ao todo)
    push.s            ;                               1 ciclo
    push W4           ;                               1 ciclo
    push W6           ;                               1 ciclo
    push W8           ;                               1 ciclo
    push W10          ;                               1 ciclo
```

```

controle.s (WTE)
    mov    #_PI_CORRENTE, W8    ;ponteiro primeiro elemento do vetor
PI_CORRENTE
    ;
    mov    #_CONTROL_CORRENTE, W10; ponteiro para o primeiro
elemento do vetor
    ;
    CONTROL_CORRENTE
    mov    _Ia_ref, W2
    mov    _Ia, W1
    sub    W2, W1, W0    ; calcula o erro
    mov    W0, [W10]
    movsac A, [W8]+=2, W4, [W10]+=2, W6    ; pré armazena os
valores de W4 e W6
    ;
    ;Eq. das diferenças utilizando a máquina DSP
    mpy    W4*W6, A, [W8]+=2, W4, [W10]+=2, W6; Acca=e(k)*c1
    1 ciclo
    mac    W4*W6, A, [W8]+=2, W4, [W10]+=2, W6; Acca=Acca-e(k-1)*c2
    1 ciclo
    mac    W4*W6, A, [W8]-=6, W4, [W10]-=6, W6; Acca=Acca+u(k-1)*c3
    1 ciclo
    sftac A, #-1    ; (<<1) - Q1.15*Q1.15=Q1.30 ->Q1.30 - Q1.31 ==-1
    1 ciclo
    sac.r A, #0, W1    ; (>>16) - dividindo por 2^(16-1) (Q15) Salva o
segundo word
    ;
    do Acca em W1
    ;Saturação negativa
    mov    #32767, W0    ;
    cpslt W1, W0    ; compara com o minimo
    1
ciclo
    mov    _MIN_DUTY, W1    ; pula se for maior que o minimo
    1
ciclo
    ;Saturação inferior
    mov    _MIN_DUTY, W0    ;
    cpsgt W1, W0    ; compara com o minimo
    1
ciclo
    mov    _MIN_DUTY, W1    ; pula se for maior que o minimo
    1
ciclo
    ;Saturação superior
    mov    _MAX_DUTY, W0    ;
    cpslt W1, W0    ; compara com o máximo
    1
ciclo
    mov    _MAX_DUTY, W1    ; pula se for menor que o máximo
    1
ciclo
    ;Registro dos valores em (t-1)
    mov    [W10], W0    ;
    1 ciclo
    mov    W0, [W10+2]    ; e(k-1)=e(k)
    1 ciclo
    mov    W1, [W10+4]    ; u(k-1)=u(k)
    1 ciclo
    mov    #_OUTPUT_CORRENTE, W0; ponteiro para a variável
OUTPUT_CORRENTE
    1 ciclo
    mov    W1, [W0]    ;
    1 ciclo
    pop    W10 ;
    1 ciclo
    pop    W8 ;
    1 ciclo
    pop    W6 ;
    1 ciclo
    pop    W4 ;
    1 ciclo
    pop.s ;
    1 ciclo
    return    ; retorno da função com o valor de W0
    3
ciclos
    ; Tempo total de execução da função
    XX ciclos =
1.0us
.end

```

```
event.c (WTE)
```

```

//*****WTE*****
#include "global.h"

```


event.c (WTE)

852, 856, 855, 850, 843, 832, 820, 807, 793, 779, 766, 754, 744, 736, 730, 725, 722, 719, 718, 717, 717, 717, 717, 718, 719, 720, 722, 725, 728, 732, 736, 741, 747, 753, 760, 765, 770, 774, 776, 776, 773, 767, 758, 746, 733, 717, 702, 686, 672, 659, 648, 640, 640, 640, 640, 640, 644, 650, 656, 661, 665, 668, 668, 667, 665, 662, 659, 657, 655, 655, 656, 660, 667, 677, 688, 700, 713, 727, 739, 750, 760, 767, 772, 773, 773, 770, 766, 760, 753, 745, 736, 728, 719, 710, 701, 692, 684, 676, 669, 662, 656, 650, 646, 642, 640, 640, 640, 640, 640, 640, 640, 640, 640, 643, 647, 652, 658, 665, 672, 681, 690, 701, 712, 724, 737, 750, 762, 774, 784, 791, 797, 799, 798, 793, 784, 774, 762, 750, 738, 728, 720, 715, 715, 720, 728, 740, 755, 770, 786, 801, 814, 825, 832, 835, 834, 830, 823, 814, 804, 793, 782, 771, 761, 753, 747, 742, 737, 733, 730, 726, 722, 718, 712, 706, 698, 690, 682, 673, 665, 657, 650, 644, 640, 640, 640, 640, 640, 641, 645, 651, 657, 665, 674, 684, 694, 705, 715, 725, 733, 740, 745, 748, 748, 745, 739, 732, 724, 715, 706, 699, 694, 690, 690, 694, 701, 709, 719, 730, 740, 749, 755, 759, 759, 755, 747, 736, 723, 708, 693, 677, 663, 650, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 641, 641, 640, 640, 640, 640, 641, 643, 648, 654, 663, 674, 686, 699, 713, 726, 738, 749, 758, 764, 768, 769, 768, 766, 763, 760, 756, 754, 753, 753, 755, 760, 765, 771, 777, 783, 788, 791, 792, 790, 785, 777, 768, 756, 744, 731, 717, 705, 693, 683, 674, 667, 662, 658, 655, 653, 651, 650, 648, 646, 643, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 648, 657, 668, 679, 691, 703, 713, 722, 730, 734, 736, 735, 733, 730, 726, 723, 721, 720, 722, 726, 735, 746, 758, 772, 785, 796, 806, 812, 814, 810, 801, 787, 768, 748, 726, 703, 683, 664, 650, 640, 640, 644, 655, 668, 684, 701, 719, 737, 754, 769, 783, 794, 804, 810, 814, 815, 813, 807, 798, 785, 769, 751, 731, 712, 693, 675, 660, 648, 640, 640, 640, 642, 649, 657, 666, 674, 681, 687, 689, 687, 683, 677, 669, 661, 653, 647, 643, 643, 646, 654, 665, 680, 696, 714, 731, 748, 763, 775, 783, 786, 786, 781, 775, 765, 755, 744, 732, 721, 711, 703, 696, 692, 689, 688, 689, 692, 697, 704, 713, 724, 736, 749, 762, 775, 787, 798, 807, 814, 817, 817, 814, 808, 801, 793, 785, 777, 770, 764, 761, 761, 763, 767, 771, 776, 781, 785, 787, 787, 784, 778, 770, 759, 747, 735, 724, 714, 706, 701, 701, 705, 712, 722, 734, 747, 759, 770, 778, 783, 783, 778, 769, 756, 740, 722, 704, 685, 668, 653, 640, 640, 640, 640, 640, 640, 640, 640, 648, 658, 669, 680, 692, 703, 714, 725, 735, 745, 753, 759, 764, 767, 769, 768, 767, 765, 762, 758, 754, 749, 745, 741, 738, 734, 731, 727, 724, 720, 716, 711, 706, 700, 694, 688, 683, 677, 672, 669, 666, 665, 666, 669, 673, 679, 685, 693, 700, 707, 714, 720, 724, 727, 729, 730, 730, 730, 729, 729, 729, 729, 731, 734, 738, 743, 748, 753, 759, 764, 769, 773, 775, 777, 777, 776, 775, 772, 769, 766, 761, 757, 753, 748, 743, 739, 734, 729, 724, 719, 713, 708, 701, 695, 689, 682, 675, 669, 663, 657, 652, 648, 645, 642, 641, 642, 643, 647, 653, 661, 671, 684, 700, 718, 739, 761, 783, 804, 823, 840, 852, 860, 862, 857, 847, 833, 815, 795, 775, 755, 737, 722, 711, 704, 703, 705, 710, 717, 725, 734, 743, 750, 756, 759, 760, 759, 757, 755, 752, 751, 750, 752, 756, 762, 771, 781, 792, 802, 811, 818, 822, 822, 818, 809, 796, 780, 762, 743, 724, 707, 691, 680, 673, 671, 675, 682, 693, 706, 720, 735, 750, 764, 776, 785, 791, 796, 798, 799, 799, 797, 795, 793, 790, 787, 785, 783, 780, 778, 777, 775, 774, 773, 772, 772, 772, 771, 771, 771, 771, 770, 770, 768, 767, 764, 762, 759, 756, 752, 749, 745, 742, 739, 736, 734, 732, 731, 730, 729, 729, 729, 730, 732, 734, 736, 739, 742, 745, 748, 752, 755, 758, 761, 763, 765, 766, 767, 768, 768, 767, 767, 766, 764, 763, 761, 759, 757, 754, 752, 750, 748, 746, 744, 742, 741, 739, 738, 737, 735, 732, 729, 725, 720, 714, 707, 699, 690, 681, 672, 663, 655, 648, 643, 640, 640, 640, 642, 647, 653, 660, 669, 678, 688, 699, 711, 722, 733, 743, 752, 759, 764, 767, 766, 762, 755, 744, 732, 718, 705, 692, 681, 672, 667, 667, 672, 681, 694, 710, 727, 746, 765, 783, 800, 814, 825, 833, 839, 842, 844, 844, 843, 841, 839, 837, 835, 834, 833, 832, 832, 831, 831, 830, 829, 828, 827, 825, 824, 822, 820, 817, 815, 812, 810, 807, 804, 801, 798, 795, 792, 789, 786, 783, 779, 776, 772, 769, 766, 763, 762, 762, 763, 767, 772, 780, 790, 802, 815, 828, 841, 853, 862, 869, 872, 870, 863, 852, 838, 821, 804, 786, 770, 756, 745, 738, 736, 738, 744, 754, 765, 777, 790, 803, 815, 825, 832, 837, 839, 840, 839, 836, 831, 825, 819, 811, 803, 794, 785, 775, 766, 756, 747, 737, 728, 720, 712, 705, 698, 691, 685, 679, 673, 668, 662, 656, 650, 645, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 649, 660, 672, 685, 697, 709, 719, 727, 731, 732, 728, 721, 711, 699, 686, 673, 661, 651, 644, 640, 641, 646, 654, 664, 676, 689, 702, 715, 726, 735, 742, 746, 748, 749, 748, 747, 745, 743, 742, 742, 743, 745, 748, 752, 756, 760, 764, 768, 772, 775, 776, 777, 777, 777, 775, 773, 770, 766, 762, 757, 752, 747, 742, 737, 734, 732, 732, 734, 739, 746, 757, 771, 785, 800, 814, 826, 835, 840, 839, 833, 819, 801, 778, 752, 726, 701, 679, 660, 648, 643, 647, 658, 676, 698, 723, 749, 775, 799, 820, 835, 845, 849, 847, 842, 832, 820, 806, 790, 773, 756, 740, 725, 711, 698, 688, 679, 672, 668, 667, 669, 674, 681, 691, 702, 714, 726, 738, 748, 757, 763, 767, 768, 767, 764, 759, 753, 746, 739, 731, 724, 716, 710, 703, 698, 692, 687, 683, 679, 675, 672, 669, 666, 664, 661, 659, 657, 654, 651, 648, 644, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 640, 651, 665, 681, 699, 717, 736, 754, 772, 787, 801, 812, 820, 825, 828, 827, 825, 819, 811, 801, 788, 772, 755, 737, 718, 700, 683, 667, 655, 645, 640, 640, 642, 649, 658, 669, 681, 694, 706, 718, 727, 735, 741, 745, 747, 749, 749, 748, 747, 746, 744, 743, 742, 741, 740, 739, 737, 735, 732, 728, 724, 718, 712, 705, 698, 691, 685, 679, 675, 672, 671, 672, 675, 680, 686, 693, 701, 710, 719, 729, 739, 749, 758, 767, 775, 783, 789, 794, 797, 799, 799, 796, 793, 788, 782, 775, 769, 763, 759, 755, 753, 753, 755, 758, 761, 764, 766, 767, 767, 764, 758, 748, 736, 722, 707, 692, 678, 664, 653, 645, 640, 640, 643, 650, 659, 670, 681, 693, 704, 714, 722, 728, 731, 731, 731, 728, 725, 720, 716, 711, 706, 701, 698, 694, 692, 690, 690, 690, 692,

```

event.c (WTE)
695, 700, 706, 713, 721, 730, 739, 747, 755, 762, 768, 772, 774, 774, 773, 770, 767, 763, 758, 752,
747, 742, 737, 733, 729, 726, 724, 723, 724, 726, 729, 735, 741, 750, 759, 769, 778, 787, 795, 802,
807, 809, 808, 805, 799, 792, 783, 774, 763, 753, 743, 734, 725, 718, 711, 706, 702, 700, 699, 700,
702, 706, 712, 718, 726, 734, 741, 748, 754, 757, 759, 758, 754, 747, 738, 727, 716, 704, 693, 682,
673, 665, 661, 658, 658, 659, 663, 667, 673, 681, 689, 697, 706, 715, 724, 733, 742, 750, 756, 762,
767, 769, 770, 770, 769, 766, 764, 761, 758, 755, 754, 753, 754, 755, 758, 761, 764, 766, 768, 768,
767, 765, 760, 754, 746, 737, 728, 718, 709, 700, 692, 685, 680, 677, 675, 674, 675, 677, 681, 685,
690, 696, 703, 711, 718, 726, 733, 740, 746, 751, 755, 757, 758, 757, 754, 750, 745, 739, 731, 724,
715, 706, 697, 688, 679, 670, 662, 655, 649, 644, 641, 640, 641, 643, 647, 652, 658, 664, 669, 675,
679, 683, 685, 685, 686, 686, 686, 686, 688, 691, 696, 703, 713, 725, 739, 753, 767, 781, 794, 806,
815, 821, 823, 823, 820, 815, 808, 800, 791, 781, 772, 764, 757, 751, 745, 741, 737, 734, 732, 730,
729, 728, 727, 727, 727, 728, 730, 732, 736, 740, 745, 752, 760, 769, 779, 789, 799, 809, 819, 827,
835, 841, 846, 849, 850, 848, 845, 840, 832, 821, 809, 793, 775, 756, 736, 716, 698, 682, 670, 663,
661, 666, 679, 697, 720, 746, 773, 799, 823, 844, 858, 865, 863, 854, 838, 817, 792, 765, 736, 707,
679, 654, 640, 640, 640, 640, 640, 660, 702, 759, 830, 896, 896, 896, 896, 896, 896, 896, 896,
896
};
pec = lookup[pec];
return pec;
}

volatile uint16_t LUT_cp(uint16_t pec)
{
    static const uint16_t __attribute__((space(auto_psv))) lookup[256] =
    {
        24, 37, 50, 63, 76, 89, 102, 115, 128, 141, 154, 167, 180, 193, 205, 218, 231, 244, 257, 270, 283,
        296, 309, 322, 335, 348, 362, 375, 389, 404, 418, 434, 450, 467, 485, 504, 525, 548, 572, 598, 627,
        658, 691, 727, 767, 809, 855, 905, 958, 1014, 1075, 1139, 1208, 1281, 1357, 1438, 1523, 1612, 1705,
        1803, 1904, 2009, 2118, 2231, 2348, 2468, 2592, 2719, 2849, 2982, 3119, 3258, 3399, 3543, 3690,
        3838, 3989, 4141, 4295, 4451, 4607, 4765, 4924, 5084, 5244, 5405, 5566, 5728, 5889, 6050, 6211,
        6372, 6532, 6691, 6850, 7007, 7164, 7319, 7473, 7626, 7777, 7926, 8074, 8220, 8364, 8506, 8645,
        8783, 8918, 9051, 9182, 9310, 9436, 9559, 9680, 9797, 9912, 10024, 10134, 10240, 10344, 10444,
        10542, 10637, 10728, 10817, 10902, 10984, 11063, 11139, 11212, 11282, 11348, 11412, 11472,
        11529, 11583, 11633, 11680, 11725, 11766, 11803, 11838, 11869, 11897, 11923, 11944, 11963,
        11979, 11992, 12001, 12007, 12011, 12011, 12008, 12003, 11994, 11982, 11968, 11950, 11930,
        11907, 11880, 11851, 11820, 11785, 11748, 11708, 11665, 11619, 11571, 11520, 11467, 11411,
        11353, 11292, 11228, 11162, 11093, 11023, 10949, 10874, 10796, 10715, 10633, 10548, 10461,
        10371, 10280, 10186, 10090, 9992, 9892, 9790, 9686, 9580, 9472, 9362, 9250, 9136, 9020, 8903,
        8783, 8662, 8539, 8415, 8288, 8160, 8030, 7899, 7766, 7631, 7495, 7357, 7218, 7077, 6935, 6791,
        6646, 6499, 6351, 6202, 6051, 5899, 5745, 5591, 5435, 5278, 5119, 4960, 4799, 4637, 4474, 4309,
        4144, 3978, 3810, 3641, 3472, 3301, 3130, 2957, 2783, 2609, 2433, 2257, 2080, 1902, 1723, 1543,
        1362, 1181, 998, 815, 632, 0
    };
    pec = lookup[pec];
    return pec;
}

/*****PROTOTIPOS DE
INTERRUPÇÕES*****/
/***** Interrupção do módulo A/D *****/
void _ISRNPVSV _ADCInterrupt(void){
    /* Leitura dos canais do ADC */
    ClrWdt(); //Kick the Dog
    PTCONbits.SEVTPS = 0; /*SPECIAL EVENT POSTSCALE 1:8*/
    if( SEVTCMPbits.SEVTCMP == spec_ev1 ){
        SEVTCMPbits.SEVTCMP = spec_ev2;
        adconvert = ADCBUF2;
        /*** Limitar leitura de corrente em valores positivos apenas
        if (adconvert-offset_sensor>= 32767) Ia1 = 32767;
        else if (adconvert<=offset_sensor) Ia1 = 0;
        else Ia1 = adconvert-offset_sensor;
        }
    }
}
else{

```

event.c (WTE)

```
SEVTCMPbits.SEVTCMP = spec_ev1;
adconvert = ADCBUF2;
/** Limitar leitura de corrente em valores positivos apenas
if (adconvert-offset_sensor >= 32767) Ia2 = 32767;
else if (adconvert <= offset_sensor) Ia2 = 0;
else Ia2 = adconvert - offset_sensor;
Ia = (Ia1 + Ia2) >> 1;

control_pi_corrente(); // cálculo de u(k)

PDC2 = OUTPUT_CORRENTE << 5;
}
ClrWdt(); // Kick the Dog
IFS0bits.ADIF = 0; // Limpar flag da interrupção
ADSTATbits.PIRDY = 0; // LIMPA ADSTAT PARA O PAR1(AN2-AN3)
return;
}

void _ISR_NPSV_CNInterrupt(void)
{
IFS1bits.CNIF = 0; // Clear CN interrupt
if (PORTBbits.RB3) {
pulsos++;
}
return;
}

void _ISR_NPSV_T1Interrupt(void) // Fazendo a parada da interrupção do timer1 na contagem de
tempo entre duas interrupções do QEI module
{
_T1IF = 0;
// Ajuste de vento
cont = cont + 1;
if (cont >= 100) { // Mudança de vento a cada 100 * 10ms = 1s
cont = 0;
cont_vento = cont_vento + 1;
if (cont_vento >= 1800) cont_vento = 0;
}
// acesso a LUT com os valores de vento
vento = LUT_wind(cont_vento);
// vento = 7 << 7;

// cálculo do rpm
// Speed = pulsos * 0.6;
rpm = (pulsos * 19) >> 5; // 0,5937
pulsos = 0;

/*
* 60 * (Pulsos_volta * contagem_estouro + contagem) 60 * (10.000 * cont + POSCNT)
* RPM = ----- = -----
* Pulsos_volta * tempo_amostragem 10.000 * 10ms
*/

// cálculo da corrente de referência Iref
if (rpm < 1) rpm = 1;
if (vento < 1) vento = 1;
omega = rpm * k_omega;
// omega = __builtin_muluu(rpm, k_omega);

lambda = omega / vento;
// lambda = __builtin_divud(omega, vento); // lambda = (omega * Raio) / vento;
// lambda = 154;
```

event.c (WTE)

```

if (lambda > 256) lambda = 256;           //lambda_opt=154
else if (lambda < 1) lambda = 1;

// acesso a LUT com os valores de lambda
Cp = LUT_cp(lambda-1);

vento_cubo = (__builtin_muluu(vento,vento))>>7; //vento_cubo = vento^3;

vento_cubo = (__builtin_muluu(vento_cubo,vento))>>7;

Pm_ref = (__builtin_muluu(vento_cubo,Cp))>>7; //Pm_ref = (Cp*vento_cubo*wind_gain);

Tm_ref = __builtin_divud(Pm_ref,rpm); //Tm_ref = Pm_ref/omega;

Ia_ref = Tm_ref; //B=0.036*(2*pi/60)*(2^16-1/5)*0.1 = 5.627

if (Ia_ref >= Corrente_6A ) Ia_ref = Corrente_6A ; // Limite superior
if((Ia_ref <= Corrente_1500mA) & (lambda<=154))Ia_ref = Corrente_1500mA ; // Limite
inferior

//if(Ia_ref >= Corrente_4500mA) Ia_ref = Corrente_2250mA;
//else Ia_ref = Corrente_4500mA;
//Ia_ref = Corrente_4A;
return;
}

void _ISRNPVSV _T2Interrupt(void)
{
    _T2IF = 0;
    return;
}

void _ISRNPVSV _T3Interrupt (void)
{
    IFS0bits.T3IF=0; // CLEAR FLAG
    tmr3_cont++; // general counter to "delay_ms()"
    /*
    FSM_botao();
    if (refresh >= taxa_atualizacao) refresh = 0;
    else refresh++;
    */
    return;
}

//Interrupções de software
/*
Primary Exception Vector handlers:
These routines are used if INTCON2bits.ALTIVT = 0.
All trap service routines in this file simply ensure that device
continuously executes code within the trap service routine. Users
may modify the basic framework provided here to suit to the needs
of their application.
*/
/*
void _ISRNPVSV _OscillatorFail(void)
{
    INTCON1bits.OSCFAIL = 0; //Clear the trap flag
    FRASE("OscFail");
    while (1);
}

void _ISRNPVSV _AddressError(void)

```

event.c (WTE)

```
{
    INTCON1bits.ADDRERR = 0;    //Clear the trap flag
    FRASE("AddressError");
    while (1);
}
void _ISRNPVSV _StackError(void)
{
    INTCON1bits.STKERR = 0;    //Clear the trap flag
    FRASE("StackError");
    while (1);
}

void _ISRNPVSV _MathError(void)
{
    INTCON1bits.MATHERR = 0;    //Clear the trap flag
    FRASE("MathError");
    while (1);
}
*/
/*
Alternate Exception Vector handlers:
These routines are used if INTCON2bits.ALTVT = 1.
All trap service routines in this file simply ensure that device
continuously executes code within the trap service routine. Users
may modify the basic framework provided here to suit to the needs
of their application.
*/
/*
void _ISRNPVSV _AltOscillatorFail(void)
{
    INTCON1bits.OSCFAIL = 0;
    FRASE("AltOscillatorFai");
    while (1);
}

void _ISRNPVSV _AltAddressError(void)
{
    INTCON1bits.ADDRERR = 0;
    FRASE("AltAddressError");
    while (1);
}

void _ISRNPVSV _AltStackError(void)
{
    INTCON1bits.STKERR = 0;
    FRASE("AltStackError");
    while (1);
}

void _ISRNPVSV _AltMathError(void)
{
    INTCON1bits.MATHERR = 0;
    FRASE("AltMathError");
    while (1);
}
*/

//*****COMANDOS DE
INTERFACE*****//
/*
static void escrever_speed(double Speed, char linha, char coluna)
{
    unsigned int Speed_buffH = (unsigned int)Speed;
```

event.c (WTE)

```
unsigned int Speed_buffL = (unsigned int)(Speed*10);

posicao_lcd(linha,coluna);
FRASE("    ");
LCD_LF();
posicao_lcd(linha,coluna);

if (Speed < 9){
    EN_LCD(Speed_buffH,1,0,0);
    ESCREVE_LCD('.');
    EN_LCD(Speed_buffL,1,0,0);
}
else if (Speed < 99){
    EN_LCD(Speed_buffH,2,0,0);
    ESCREVE_LCD('.');
    EN_LCD(Speed_buffL,1,0,0);
}
else if (Speed < 999){
    EN_LCD(Speed_buffH,3,0,0);
    ESCREVE_LCD('.');
    EN_LCD(Speed_buffL,1,0,0);
}
else {
    EN_LCD(Speed_buffH,4,0,0);
    ESCREVE_LCD('.');
    EN_LCD(Speed_buffL,1,0,0);
}
FRASE (" rpm");

return;
}*/

/*
int median_filter(unsigned int datum)
{
    struct pair
    {
        struct pair *point;           // Pointers forming list linked in sorted order
        unsigned int value;          // Values to sort
    };

    static struct pair buffer[MEDIAN_FILTER_SIZE]; // = {0}; // Buffer of nwidth pairs
    static struct pair *datapoint = buffer;        // Pointer into circular buffer of data
    static struct pair small = {NULL, STOPPER};    // Chain stopper
    static struct pair big = {&small, 0};         // Pointer to head (largest) of linked list.

    struct pair *successor;           // Pointer to successor of replaced data item
    struct pair *scan;                // Pointer used to scan down the sorted list
    struct pair *scanold;             // Previous value of scan
    struct pair *median;              // Pointer to median
    unsigned int i;

    if (datum == STOPPER)
    {
        datum = STOPPER + 1;          // No stoppers allowed.
    }

    if ( (++datapoint - buffer) >= MEDIAN_FILTER_SIZE)
    {
        datapoint = buffer;           // Increment and wrap data in pointer.
    }

    datapoint->value = datum;         // Copy in new datum
```

event.c (WTE)

```
    successor = datpoint->point;           // Save pointer to old value's successor
    median = &big;                          // Median initially to first in chain
    scanold = NULL;                          // Scanold initially null.
    scan = &big;                             // Points to pointer to first (largest) datum in chain

// Handle chain-out of first item in chain as special case
if (scan->point == datpoint)
{
    scan->point = successor;
}
scanold = scan;                             // Save this pointer and
scan = scan->point ;                         // step down chain

// Loop through the chain, normal loop exit via break.
for (i = 0 ; i < MEDIAN_FILTER_SIZE; ++i)
{
// Handle odd-numbered item in chain
if (scan->point == datpoint)
{
    scan->point = successor;                // Chain out the old datum.
}

if (scan->value < datum)                    // If datum is larger than scanned value,
{
    datpoint->point = scanold->point;        // Chain it in here.
    scanold->point = datpoint;              // Mark it chained in.
    datum = STOPPER;
};

// Step median pointer down chain after doing odd-numbered element
median = median->point;                      // Step median pointer.
if (scan == &small)
{
    break;                                  // Break at end of chain
}
scanold = scan;                             // Save this pointer and
scan = scan->point;                          // step down chain

// Handle even-numbered item in chain.
if (scan->point == datpoint)
{
    scan->point = successor;
}

if (scan->value < datum)
{
    datpoint->point = scanold->point;
    scanold->point = datpoint;
    datum = STOPPER;
}

if (scan == &small)
{
    break;
}

    scanold = scan;
    scan = scan->point;
}
return median->value;
}*/
/*
```

event.c (WTE)

```
void atualizar_LCD(void)
{
    unsigned int PWM1, PWM2;
    static uint_fast8_t pisca;
#ifdef 0
    uint_fast8_t lut;
    static uint_fast8_t cc;
    static uint_fast16_t foo;
#endif

    switch (display.tela)
    {
        case 0:
        {
            LIMPA_LCD();
            //PWM2 = (1 - ((double)PDC2/max_duty))*100;
            PWM2 = (((double)PDC2/max_duty))*100;
            FRASE("AD=");
            EN_LCD(adconvert,4,0,0);
            FRASE(" Ia=");
            EN_LCD(Ia,4,0,0);
            escrever_speed(Speed,2,1);
            posicao_lcd(2,11);
            FRASE("pwm");
            EN_LCD(PWM2,3,0,0);
            break;
        }

        case 1:
        {
            if (!display.set)
            {
                LIMPA_LCD();
                //PWM1 = (1 - ((double)PDC1/max_duty))*100;
                //PWM2 = (1 - ((double)PDC2/max_duty))*100;
                PWM1 = (((double)PDC1/max_duty))*100;
                PWM2 = (((double)PDC2/max_duty))*100;
                FRASE("PWM1=");
                EN_LCD(PWM1,3,0,0);
                posicao_lcd(1,10);
                EN_LCD(cont_vento,2,0,0);
                linha_lcd(2);
                FRASE("PWM2=");
                EN_LCD(PWM2,3,0,0);
            }
            else
            {
                pisca ^= 1;
                if (press == incrementa)
                    PDC1 -= 49; // +5% do ciclo de trabalho
                else if (press == decrementa)
                    PDC1 += 49; // -5% do ciclo de trabalho
                press = 0;
                //PWM1 = (1 - ((double)PDC1/max_duty))*100;
                PWM1 = (((double)PDC1/max_duty))*100;
                posicao_lcd(1,6);
                if (!pisca)
                {
                    EN_LCD(PWM1,3,0,0);
                }
                else
                    FRASE(" ");
            }
        }
    }
}
```

event.c (WTE)

```
        break;
    }

    case 2:
    {
        LIMPA_LCD();
#ifdef 0
        if (display.set == 0)
        {
            cc++;
            lut = pec_Update(cc);
            FRASE("LUT PSV model...");
            linha_lcd(2);
            EN_LCD(lut,3,0,0);
        }
        else
        {
            pisca ^= 1;
            if (press == incrementa)
                foo++;
            else if (press == decrementa)
                foo--;
            press = 0;
            FRASE("Alterando: ");
            if (pisca == 0)
                EN_LCD(foo,3,0,0);
        }
#endif
        escrever_speed(Speed,1,1);
        linha_lcd(2);
        EN_LCD(Tm_ref,5,0,0);
        FRASE(" voltas");

        break;
    }

    case 3:
    {
        //      double current;
        //      current = Ia_ref;
        //      current = current/base_corrente;
        //      //PWM2 = (1 - ((double)PDC2/max_duty))*100;
        //      PWM2 = (((double)PDC2/max_duty))*100;
        //      LIMPA_LCD();
        //      //CONTROL_CORRENTE[0]
        //      FRASE("Re=");
        //      //EN_LCD((Ia_ref),4,0,0);
        //      escrever_speed(current,1,4);
        //      posicao_lcd(1,8);
        //      FRASE("A");
        //      FRASE(" PWM=");
        //      EN_LCD(PWM2,3,0,0);
        //      linha_lcd(2);
        //      FRASE("Lamb=");
        //      EN_LCD(lambda,5,0,0); //era erro
        //      FRASE(" ");
        //      EN_LCD(rpm,4,0,0);
        break;
    }

    default:
    {
```

event.c (WTE)

```

        // Nothing
    }

    break;
}

return;
}*/

/*
void FSM_botao(void)
{
    static uint_fast8_t mode;
    static uint_fast16_t fsm_cont;

    switch (mode)
    {
        case STATE_IDLE: // idle
        {
            if (ler_botao1 && ler_botao2)
            {
                mode = STATE_IDLE; // idle
                fsm_cont = 0;
            }
            else if (!ler_botao1)
                mode = STATE_SINGLE_1; // if RB0 or RB1...
            else if (!ler_botao2)
                mode = STATE_SINGLE_2;
            break;
        }
        case STATE_SINGLE_1:
        {
            if (!ler_botao1)
            {
                ++fsm_cont;
                if (fsm_cont >= debounce)
                {
                    mode = STATE_LONG_1; // long press
                    botao.spb1 = 1; // single press botao 0
                }
                else mode = STATE_SINGLE_1; // single press
            }
            else
            {
                mode = STATE_IDLE;
                fsm_cont = 0;
                botao.botao1 = 0;
                botao.botao2 = 0;
            }

            break;
        }
        case STATE_LONG_1:
        {
            if (!ler_botao1)
            {
                ++fsm_cont;
                if (fsm_cont >= (debounce*20))
                {
                    mode = STATE_RELEASE; // deciser
                    botao.lpb1 = 1;
                    botao.spb1 = 0;
                }
            }
        }
    }
}

```

event.c (WTE)

```
        else mode = STATE_LONG_1;          // long press
    }
    else
    {
        mode = STATE_RELEASE;
        fsm_cont = 0;
        botao.spb1 = 1;
    }

    break;
}
case STATE_SINGLE_2:
{
    if (!ler_botao2)
    {
        ++fsm_cont;
        if (fsm_cont >= debounce)
        {
            mode = STATE_LONG_2; // long press
            botao.spb2 = 1; // single press button 1
        }
        else mode = STATE_SINGLE_2;        // single press
    }
    else
    {
        mode = STATE_IDLE;
        fsm_cont = 0;
        botao.botao1 = 0;
        botao.botao2 = 0;
    }

    break;
}
case STATE_LONG_2:
{
    if (!ler_botao2)
    {
        ++fsm_cont;
        if (fsm_cont >= (debounce*20))
        {
            mode = STATE_RELEASE; // deciser
            botao.lpb2 = 1;
            botao.spb2 = 0;
        }
        else mode = STATE_LONG_2;          // long press
    }
    else
    {
        mode = STATE_RELEASE;
        fsm_cont = 0;
        botao.spb2 = 1;
    }

    break;
}
case STATE_RELEASE:
{
    comando(botao);
    botao.botao1 = 0;
    botao.botao2 = 0;
    fsm_cont = 0;
    mode = STATE_IDLE;
}
```

event.c (WTE)

```
        break;
    }
    default:
    {
        //Nothing
    }

    break;
}
return;
}*/

/*
void comando(BOTAOBITS botao)
{

    if (botao.spb1)
    {
        if (display.set ==0)
        {
            if (display.tela >= 3)
                display.tela = 0;
            else
                display.tela++;
        }
        else press = incrementa;
    }

    if (botao.spb2)
    {
        if (display.set ==0)
        {
            if (display.tela <= 0)
                display.tela = 3;
            else
                display.tela--;
        }
        else press = decrementa;
    }
    else if (botao.lpb2) display.set ^= 1;

    while(!ler_botao1 || !ler_botao2);
    return;
}*/
```

init.c (WTE)

```
/******WTE*****
#include "global.h"

/** -----Função de configurações gerais do dsPIC----- */
void iniciarPIC(void){
    while(!OSCCONbits.LOCK);
    TRISB = 0x0000;    // Porta B como saída
    TRISEbits.TRISE6 = 0;
    LATEbits.LATE6 = 1;
    TRISD = 0x0000;    // Porta D como saída
    TRISE = 0x0000;    // Porta E como saída
    TRISF = 0x0000;    // Porta F como saída
    TRISBbits.TRISB0 = 1; // Pino AN0
    TRISBbits.TRISB1 = 1; // Pino AN1
    TRISBbits.TRISB2 = 1; // Pino AN2
    TRISAbits.TRISA9 = 1; // botao 1
    TRISEbits.TRISE7 = 1; // botao 2
```

init.c (WTE)

```
INTCON2bits.ALTIVT = 0; // habilita ISR's de falha

return;
}

/** -----Função de configurações do módulo A/D----- */
void iniciarADC(void){
    ADCONbits.ADON = 0;    /*DESLIGA O MÓDULO ADC*/
    ADCONbits.FORM = 1;    /*SELECIONA O MODO FRACIONAL (dddd dddd dd00 0000)*/
    ADCONbits.EIE = 1;    /*INTERRUPÇÃO APÓS PRIMEIRA CONFIG*/
    ADCONbits.ADCS = 0b010; /*CLOCK DE 20 MHZ*/
    ADCONbits.ORDER = 0;  /*CANALIS PARES CONVERTIDOS PRIMEIRO*/
    ADSTAT=0;             /*LIMPA O REGISTRO DE STATUS*/
    ADPCFG=0xFFFB;       /*AN2 COMO ENTRADA ANALÓGICA*/

    ADCPC0bits.TRGSRC1=3; /* Trigger conversion on PWM Special Event */
    ADCPC0bits.IRQEN1=1; /* Enable the interrupt AN3-AN2 */

    /*INTERRUPÇÕES*/
    IFS0bits.ADIF = 0;    /* Clear AD Interrupt Flag */
    IPC2bits.ADIP = PRI_AD; /* Set ADC Interrupt Priority */
    IEC0bits.ADIE = 1;    /* Enable the ADC Interrupt */

    ADCONbits.ADON = 1;    /* Start the ADC module */
    return;
}

/** -----Função de configurações do módulo PWM----- */
void iniciarPWM(void){

    PTCONbits.PTEN = 0; /*DESLIGA O MÓDULO PWM*/
    //PTCONbits.SEIEN = 0; /*HABILITA INTERRUPÇÃO SPECIAL EVENT*/
    PTCONbits.EIPU = 0; /*ATUALIZAÇÃO DE DUTY SINCRONA*/
    //Special Event
    PTCONbits.SEVTPS = 0; /*SPECIAL EVENT POSTSCALE 1:1*/
    SEVTCMPbits.SEVTCMP = spec_ev1;

    /*MÓDULO PWM1*/
    PWMCON1 = 0x0081; /*PDCx updates duty cycle, SEM DEAD TIME, UPDATE DO PDC
IMEDIATA*/
    FCLCON1 = 0x0003; /*FAULT DESABILIDADO*/
    IOCON1 = 0x4440; /*PWM1L, MODO INDEPENDENTE, */
    PDC1 = duty_70; /*DUTY CYCLE - 70% */

    /*MÓDULO PWM2*/
    PWMCON2 = 0x0081; /*PDCx updates duty cycle, SEM DEAD TIME, UPDATE DO PDC
IMEDIATA*/
    FCLCON2 = 0x0003; /*FAULT DESABILIDADO*/
    IOCON2 = 0x4440; /*PWM1L, MODO INDEPENDENTE, */
    PDC2 = 0; /*DUTY CYCLE - 70% */

    PTPER = ptper; /*PWM DE 20KHZ */
    PTCONbits.PTEN = 1; /* HABILITA O MÓDULO PWM */

    return;
}

void iniciarTMR1(void)
{
    //CONFIGURANDO O REGISTRADOR DE CONTROLE DO TIMER1

```

init.c (WTE)

```
T1CONbits.TON = 0; //desligando o timer pra poder fazer as config no maluco
T1CONbits.TCS = 0;
T1CONbits.TGATE = 0; //nessa linha e na de cima eu configuro o mlk pra operar no modo "timer"
T1CONbits.TCKPS = 0b01; //prescale de 1:8

TMR1 = 0; //limpando a merda do registrador do timer
PR1 = Fcy/(8*100);

//CONFIGURANDO A PORRA DA INTERRUPÇÃO DO MALUCO
IPC0bits.T1IP = PRI_TMR1; //define a merda com maior prioridade
IFS0bits.T1IF = 0; //limpando a flag da interrup do T1
IEC0bits.T1IE = 1;

T1CONbits.TON = ON;
return;
}

void iniciarTMR2(void)
{
    //CONFIGURANDO O REGISTRADOR DE CONTROLE DO TIMER2
    T2CONbits.TON = 0; //desligando o timer pra poder fazer as config no maluco
    T2CONbits.TCS = 0;
    T2CONbits.TGATE = 0; //nessa linha e na de cima eu configuro o mlk pra operar no modo "timer"
    T2CONbits.TCKPS = 0b11;
    //0x00 - prescale de 1:1
    //0x01 - prescale de 1:8
    //0x10 - prescale de 1:64
    //0x11 - prescale de 1:256

    TMR2 = 0; //limpando a merda do registrador do timer
    PR2 = 3125; // (1:1 - 20000:1ms) // (1:64) - 3125:10ms

    //CONFIGURANDO A INTERRUPÇÃO
    IPC1bits.T2IP = PRI_TMR2; //define a merda com maior prioridade
    IFS0bits.T2IF = 0; //limpando a flag da interrup do T1
    IEC0bits.T2IE = 1;
    T2CONbits.TON = 1; // Liga Timer
    return;
}

void iniciarTMR3(long int frequencia, long int prescale, int start)
{
    //----->Inicialização<-----
    T3CONbits.TON = 0; //desligado
    PR3 = Fcy/(prescale*frequencia);
    TMR3 = 0;
    T3CONbits.TGATE = 0;
    //T1CONbits.T32 = 0;
    T3CONbits.TCS = 0;
    //----->Interrupções<-----
    _T3IP = PRI_TMR3;
    IFS0bits.T3IF = 0;
    IEC0bits.T3IE = 1;
    //----->Prescale<-----
    if (prescale == 1) //Atribuindo time base input prescale
        T3CONbits.TCKPS = 0b00;
    else if (prescale == 8)
        T3CONbits.TCKPS = 0b01;
    else if (prescale == 64)
        T3CONbits.TCKPS = 0b10;
    else if (prescale == 256)
        T3CONbits.TCKPS = 0b11;
}
```

init.c (WTE)

```
T3CONbits.TCKPS = 0b11;
else
T3CONbits.TCKPS = 0b00;
//----->Start<-----
T3CONbits.TON = start;

return;
}

void iniciarCN(void)
{
ADPCFG |= 0x0008; // força pino INDEX a ser entrada digital
TRISBbits.TRISB3 = 1;
T1CONbits.TON = ON;
IEC1bits.CNIE = 1;
IFS1bits.CNIF = 0;
IPC6bits.CNIP = PRI_QEI;
//CNPU1bits.CN5PUE = 1; // habilita pull-up. usar só se nao tiver funcionando
CNEN1bits.CN5IE = 1;
}
```

main.c (WTE)

```
/**WTE***/
#include "global.h"

_FOSCSEL(PRIOSSEL_PLL)
_FOSC(HS & OSC2_IO & FRC_HI_RANGE & CSW_FSCM_OFF)
_FPOR(PWRT_128)
_FWDT(FWDTEN_OFF)
//_FWDT(WDTPRE_PR128 & WDTPRST_PS128 & WINDIS_OFF & FWDTEN_ON)
_FGS(CODE_PROT_OFF)

/** -----Declaração de variáveis----- */
/* X space 0x0800 a 0x08FF
* Y space 0x0900 a 0x09FF
* 30F2010 exclusivo!
*/
volatile int_fast16_t _XDATA(16) PI_CORRENTE[3];
volatile int_fast16_t _YDATA(16) CONTROL_CORRENTE[3]; // e(k)->0,e(k-1)->1,u(k-1)->2
volatile int_fast16_t OUTPUT_CORRENTE = 0;
volatile int_fast16_t MIN_DUTY=min_duty, MAX_DUTY=max_duty;
//Leitura AD
volatile int_fast16_t Ia, Ia1, Ia2;
volatile uint_fast16_t adconvert;
volatile int_fast16_t Ia_ref; // = Corrente_3A;
volatile uint_fast8_t degrau_referencia = 0;
//Variáveis de Emulação
volatile uint_fast16_t cont;
volatile uint_fast16_t cont_vento;
volatile uint_fast16_t vento;
volatile uint_fast32_t rpm;
volatile uint_fast16_t pulsos;
const uint_fast32_t k_omega=174;
volatile uint_fast32_t omega;
volatile uint_fast32_t lambda;
volatile uint_fast16_t Cp;
volatile uint_fast16_t vento_cubo;
volatile uint_fast32_t Pm_ref;
volatile uint_fast16_t Tm_ref;
volatile int_fast16_t Ia_ref;
//Gerais
unsigned int dj,dnop;
volatile uint_fast24_t tmr3_cont;
```

```
main.c (WTE)
```

```
volatile uint_fast8_t press;
volatile uint_fast16_t voltas;
volatile uint_fast16_t timer_buff, refresh, pulsos_buffer;

int main(void) {
    // Boot
    OSCTUNbits.TUN=0b0000; /*CONFIGURA O AJUSTE FINO PARA 100% */
    setPin();
    iniciarPIC();          // Configuração de pinagem
    iniciarADC();          // Configuração do módulo A/D
    iniciarPWM();          // Configuração do PWM
    iniciarCN();           // Configuração do CN - QEI
    iniciarTMR1();         // usado para controle de referência
    //iniciarTMR2();        // Não utilizado
    iniciarTMR3(f_base,8,1); // Ligar funções 'delay'
    CORCON |= 0x00F1;      // computa inteiros sinalizados e ativa saturação dos acumuladores A
e B

    delay_ms(1500);

    /*
    //LIGA O DISPLAY LCD
    CONFIG_LCD();          // Config display
    delay_ms(300);         // Self-test
    LIMPA_LCD();
    FRASE("dsPIC 30F2020");
    posicao_lcd(2,1);
    FRASE ("Iniciando...");
    delay_ms(500);
    LIMPA_LCD();
    delay_ms(300);
    FRASE("Aperte para");
    linha_lcd(2);
    FRASE("iniciar...");
    //while(!botao.spb1 && !botao.spb2);
    */

    PI_CORRENTE[0] = 6309 ;
    PI_CORRENTE[1] = -5593 ;
    PI_CORRENTE[2] = 32767 ;
    /* 2kHz -80°
    num: {[0.1925 -0.1707]}
    den: {[1 -1]}
    c1: 6.3088e+03
    c2: -5.5931e+03
    c3: 32767
    */
    clrPin();

    while (1){
        /*
        if (refresh >= taxa_atualizacao){
            degrau_referencia++;
            refresh = 0;
            atualizar_LCD();
        }*/
    }
}
```


PCB da placa do processador do conversor Boost controlador de carga

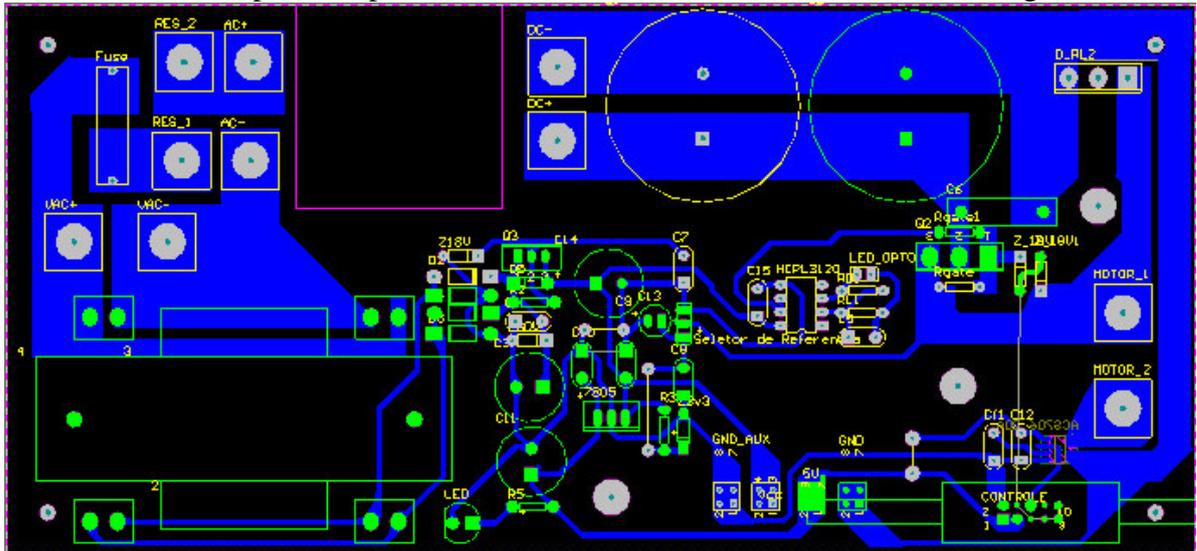
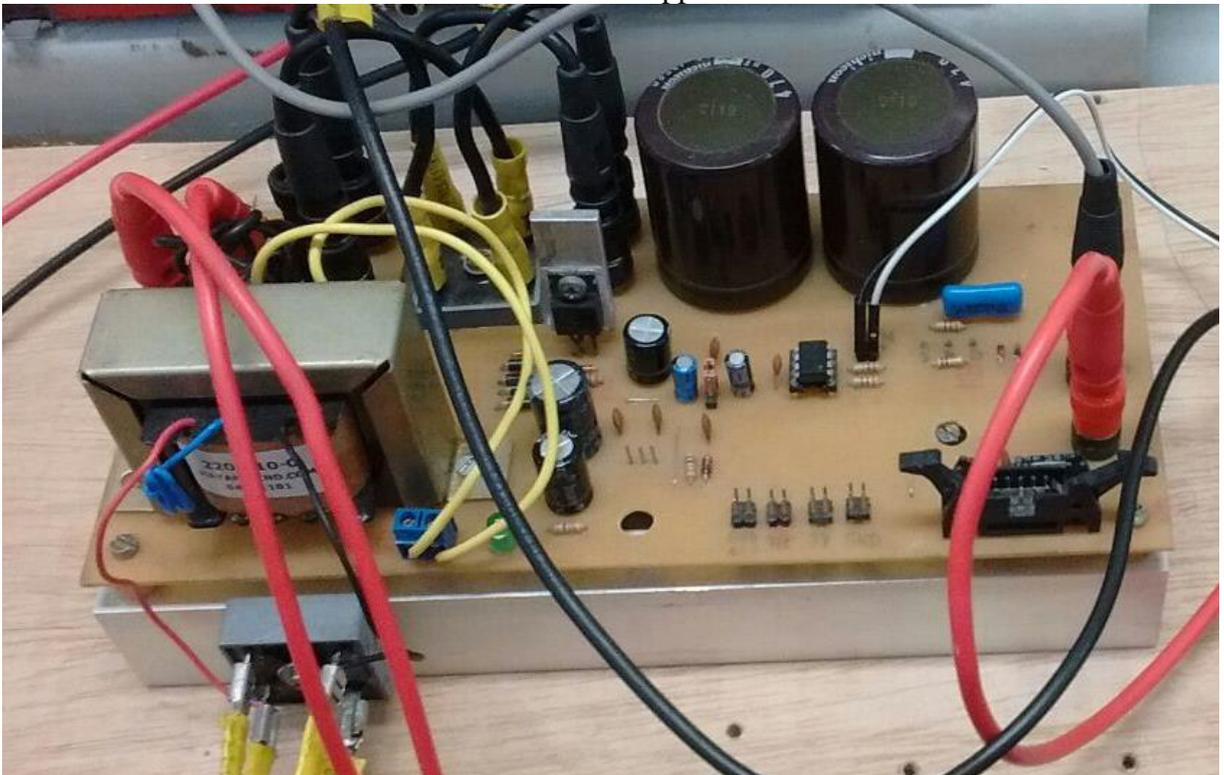


Foto do conversor Chopper finalizado



APÊNDICE D: CÓDIGOS MPLAB DO BOOST

Índice do projeto WECS

-global.h
-controle.s
-event.c
-init.c
-main.c

```
global.h (WECS)
/*****WECS*****/
#include <p30F4011.h>
#include <stdint.h>
#include <dsp.h>
//
/*****VARIÁVEIS GLOBAIS*****/
// PI Control //
/* X space 0x0800 a 0x0BFF __attribute__((section (".xbss, bss, xmemory")))
//__attribute__((address(0x0800)))
* Y space 0x0C00 a 0x0FFF __attribute__((section (".ybss, bss, ymemory")))
//__attribute__((address(0x0C00)))
* 30F2010 exclusivo! __attribute__ ( (space(xmemory)))
*/
extern volatile int_fast16_t _XDATA(16) CONST_PI[3];
extern volatile int_fast16_t _YDATA(16) CONTROL_PI[3];
extern volatile int_fast16_t _XDATA(16) PI_ROTACAO[3];
extern volatile int_fast16_t _YDATA(16) CONTROL_ROTACAO[3];
extern volatile int_fast16_t _XDATA(16) PI_CORRENTE[3];
extern volatile int_fast16_t _YDATA(16) CONTROL_CORRENTE[3];
extern volatile int_fast16_t IL, Wm, Wm_old, Vi;
extern volatile int_fast16_t IL_ref, Wm_ref, Wm_aux;
extern volatile int_fast16_t MIN_DUTY, MAX_DUTY;
extern volatile int_fast16_t MAX_IL, MIN_IL;
extern volatile int_fast16_t OUTPUT_CORRENTE;
extern volatile int_fast16_t OUTPUT_ROTACAO;
// Generic
extern volatile uint_fast8_t flag_control, cont_control, cont_uart;
extern volatile uint_fast24_t tmr3_cont;
// QIE
extern volatile int_fast16_t timer_buff, cont_buff, cont, refresh, voltas;
extern volatile int_fast16_t pulsos, pulsos_buffer;
// Média Corrente
extern volatile uint_fast16_t IL_buffer[16];
extern volatile uint_fast8_t cont_IL_mean;
//
/*****DEFINIÇÕES DE OPERAÇÃO*****/
#define MIPS 20
#define frc_osc 1000000 //cristal 10MHz
#define Fosc 8000000 //Fosc=8*frc_osc
#define FCY 2000000 //Fcy=Fosc/4
#define Tcy 0.000000050 //Tcy=1/Fcy
//
/*****DEFINIÇÕES DE PWM*****/
#define Fpwm 20000 //20kHz
#define ptpcr 499 //((FCY/2)/Fpwm)-1 //Up-Down: 499
#define max_duty 948 //(2*ptpcr)*0.95 //Maximo duty cycle 95%
#define min_duty 50 //50 //(2*ptpcr)*0.05 //Minimo duty cycle 05%
#define duty_90 898 //(2*ptpcr)*0.9 //duty cycle de 90%
#define duty_80 798 //(2*ptpcr)*0.9 //duty cycle de 80%
#define duty_70 699 //(2*ptpcr)*0.7 //duty cycle de 70%
#define duty_50 499 //(2*ptpcr)*0.5 //duty cycle de 30%
```

```

global.h (WECS)
#define duty_30 299      //(2*ptper)*0.3 //duty cicle de 30%
#define duty_10 100     //(2*ptper)*0.1 //duty cicle de 10%
#define duty_02 20      //(2*ptper)*0.02 //duty cicle de 02%
//
/*****DEFINIÇÕES DE ADC*****/
#define base_corrente 2621 //Boost_Hi 0.4057
#define max_corrente 4*base_corrente/32
#define min_corrente 0
#define Corrente_500mA 0.5*base_corrente
#define Corrente_600mA 0.6*base_corrente
#define Corrente_1500mA 1.5*base_corrente
#define Corrente_1A 1*base_corrente
#define Corrente_1500mA 1.5*base_corrente
#define Corrente_2A 2*base_corrente
#define Corrente_2500mA 2.5*base_corrente
#define Corrente_3A 3*base_corrente
#define Corrente_3500mA 3.5*base_corrente
#define Corrente_4A 4*base_corrente
#define STOPPER 0 // Smaller than any datum
#define MEDIAN_FILTER_SIZE (50)
//
/*****DEFINIÇÕES DE UART*****/
#define BaudGen 4 // Baud Rate = Fcy/(16*(BRG+1))
// Baud Rate = 256000 -> BRG = 4 (250000)
// Baud Rate = 115200 -> BRG = 10 (113636)
// Baud Rate = 9600 -> BRG = 129 (9615)

/*****DEFINIÇÕES DE PRIORIDADE DE INTERRUPÇÕES*****/
#define PRI_PWM 1
#define PRI_TX 1
#define PRI_RX 2
#define PRI_TMR1 1
#define PRI_TMR2 4
#define PRI_TMR3 1
#define PRI_TMR45 3
#define PRI_QEI 3
#define PRI_AD 5
//
/*****DEFINIÇÕES DE QUADRATURE ENCODER (QEI)*****/
#define numero_de_pulsos_por_volta 10000
#define RPM_nominal 1800 //rpm
#define base_rotacao 32 // 2^5=32
#define max_rotacao 1000*base_rotacao
#define min_rotacao 0
#define rotacao_300rpm 300*base_rotacao
#define rotacao_400rpm 400*base_rotacao
#define rotacao_500rpm 500*base_rotacao
#define rotacao_600rpm 600*base_rotacao
//
/*****DEFINIÇÕES DE TIMERx*****/
#define f_base 1000
#define m_base f_base/1000
#define Tcy_nano 50 // 1/Fcy (nanosegundos)
// #define Tcy 135.69
#define cyc 500/Tcy_nano
//
/*****DEFINIÇÕES DE GERAIS*****/
#define _ISRNPVS __attribute__((interrupt, no_auto_psv))
#define ON 1
#define OFF 0
#define EN 1
#define DIS 0
#define DEFAULT_ 0

```

global.h (WECS)

```
#define AR 10
#define debounce (50U) // em milissegundos
#define taxa_atualizacao (300U) // in mileseconds
//
/*****PROTOTIPOS DE FUNÇÕES*****/
/* esse artifício é melhor compreendido pelo compilador para executar uma otimiza
 * ação External Functions extern void escrever_speed(double Speed, char linha,
 * char coluna); moved to static qualifier declaration*/
extern void iniciarPIC(void);
extern void iniciarADC(void);
extern void iniciarPWM(void);
extern void iniciarUART(void);
extern void iniciarQEI(void);
extern void iniciarTMR1(void);
extern void iniciarTMR2(void);
extern void iniciarTMR3(long int frequencia, long int prescale, int start);
extern void iniciarTMR45(void);
extern void delay_ms (long int x);
extern void delay_us (long int x);
//
extern int control_pi_corrente(void);
extern int control_pi_rotacao(void);
extern int tglLed(void);
extern int setLed(void);
extern int clrLed(void);
//
/*****PROTOTIPOS DE INTERRUPÇÃO*****/
extern void _ISRNPVSV _ADCInterrupt(void);
extern void _ISRNPVSV _QEIIInterrupt(void);
extern void _ISRNPVSV _T1Interrupt (void);
extern void _ISRNPVSV _T2Interrupt (void);
extern void _ISRNPVSV _T3Interrupt (void);
extern void _ISRNPVSV _T5Interrupt (void);
extern void _ISRNPVSV _U2TXInterrupt (void);
extern void _ISRNPVSV _U2RXInterrupt (void);
```

controle.s (WECS)

```
*****WECS*****
; Select the device family
.equ __dsPIC30F__,1
;equ __PIC24H__,1
;
.ifdef __dsPIC30F__
.include "p30fxxxx.inc"
.endif
;
.global _control_pi_corrente; declarando como label global "control_PI_corrente"
.global _control_pi_rotacao ; declarando como label global "control_PI_corrente"
.global _tglLed
.global _setLed
.global _clrLed
;
;
;
_control_pi_corrente::função "control_PI_corrente"
    push.s;
    push W4 ; 1 ciclo
    push W6 ; 1 ciclo
    push W8 ; 1 ciclo
    push W10 ; 1 ciclo
;
```


controle.s (WECS)

```

;Saturação negativa
mov #32767,W0 ; 1 ciclo
cpslt W1,W0 ; compara com o minimo 1 ciclo
mov _MIN_IL,W1 ; pula se for maior que o minimo 1 ciclo
;Saturação inferior
mov _MIN_IL,W0 ; 1 ciclo
cpsgt W1,W0 ; compara com o minimo 1 ciclo
mov _MIN_IL, W1 ; pula se for maior que o minimo 1 ciclo
;Saturação superior
mov _MAX_IL,W0 ; 1 ciclo
cpslt W1,W0 ; compara com o máximo 1 ciclo
mov _MAX_IL, W1 ; pula se for menor que o máximo 1 ciclo
;
mov #_OUTPUT_ROTACAO, W0; ponteiro para a variável OUTPUT_ROTACAO 1 ciclo
mov W1,[W0] ; 1 ciclo
;wk=u1k*(Tamos/Ti)+w1k*(1-(Tamos/Ti))
movsac B,[W9]+=2, W5, [W11]+=2, W7 ; pré armazena os valores de W5 e W6
; 1 ciclo
mpy W5*W7, B, [W9]-=2, W5, [W11]-=2, W7 ; Acca=u1k*(Tamos/Ti)
; 1 ciclo
mac W5*W7, B ; Acca=Acca + w1k*(1-(Tamos/Ti)) 1 ciclo
sac.r B,#-1, W1 ; 1 ciclo
mov W1,[W11] ; 1 ciclo
add #2,W11 ; 1 ciclo
mov #_OUTPUT_ROTACAO, W0; ponteiro para a variável OUTPUT_ROTACAO 1 ciclo
mov [W0],W1 ; 1 ciclo
mov W1,[W11] ; 1 ciclo
;
pop W11 ; 1 ciclo
pop W9 ; 1 ciclo
pop W7 ; 1 ciclo
pop W5 ; 1 ciclo
pop.s ; 1 ciclo
return ; retorno da função com o valor de W0 3 ciclos
; Tempo total de execução da função 44 ciclos = 1.5us
;
; Toggle RC14
_tglLed:
    BTG LATE,#2
return
;
; Set RC14
_setLed:
    bset LATE,#2
return
;
; Clear RC14
_clrLed:
    bclr LATE,#2
return
.end

```

event.c (WECS)

```

//*****WECS*****
#include "global.h"
/*****Interrupção do módulo ADC*****/
void _ISRNPSV _ADCInterrupt(void){
    setLed();
    /* Leitura dos canais do ADC */
    while(!ADCON1bits.DONE);
    Vi = ADCBUF1>>1; //Leitura do canal CH1-AN0
}

```

event.c (WECS)

```
IL = ADCBUF2>>1; //Leitura do canal CH2-AN1
IL_buffer[cont_IL_mean]=IL;
cont_IL_mean=cont_IL_mean+1;
if(cont_IL_mean==16) cont_IL_mean=0;
/*****PROTOCOLO DE APLICAÇÃO DAS MALHAS DE CONTROLE*****/
control_pi_corrente(); // cálculo de u(k)
PDC1 = OUTPUT_CORRENTE;
/*****CONFIGURAÇÃO DE ESPECIAL EVENT*****/
if(SEVTCMPbits.SEVTCMP==0){
    SEVTCMPbits.SEVTDIR = 1;
    SEVTCMPbits.SEVTCMP = ptpcr;
}
else if(SEVTCMPbits.SEVTCMP==ptpcr){
    SEVTCMPbits.SEVTDIR = 0;
    SEVTCMPbits.SEVTCMP = 0;
}
clrLed();
IFS0bits.ADIF = 0; // Limpar flag da interrupção
return;
}
/*****ISR Quadrature Encoder Interface Module*****/
void _ISRNPSV_QEInterrupt(void)
{
    _QEIIF = 0; // limpa a flag
    cont = cont + 1;
    return;
}
//
void _ISRNPSV_T1Interrupt (void)
{
    _T1IF = 0;
    return;
}
//
void _ISRNPSV_T2Interrupt (void)
{
    IFS0bits.T2IF = 0;
    // ****Encoder operando em tempo fixo**** //
    if(QEICONbits.UPDN==1){
        if (POSCNT>=pulsos_buffer) pulsos = POSCNT - pulsos_buffer;
        else pulsos = (numero_de_pulsos_por_volta*2 - pulsos_buffer) + POSCNT;
    }
    else {
        if (POSCNT<=pulsos_buffer) pulsos = pulsos_buffer - POSCNT;
        else pulsos = (numero_de_pulsos_por_volta*2 - POSCNT) + pulsos_buffer;
    }
    pulsos_buffer = POSCNT;
    if (cont>0) Wm = 10*((numero_de_pulsos_por_volta*2*cont) + pulsos);
    else Wm = 10*pulsos;
    cont = -1;
}
/*
 * 60*(Pulsos_volta*contagem_estouro + contagem) 60*(20.000*cont + POSCNT)
 *RPM=----- x 32
 * Pulsos_volta * tempo_amostragem 20.000 * 10ms
 */
control_pi_rotacao();
IL_ref = OUTPUT_ROTACAO<<5;
TMR2 = 0;
return;
}
//
void _ISRNPSV_T3Interrupt (void)
{
```

event.c (WECS)

```

IFS0bits.T3IF=0; // CLEAR FLAG
tmr3_cont++; // general counter to "delay_ms()"
return;
}
//
void _ISRNPVSV _T5Interrupt (void)
{
    TMR4 = 0; //limpando o registrador do timer
    TMR5 = 0; //limpando o registrador do timer

    uint_fast32_t IL_mean=0;
    int i=0;
    for (i=0;i<16;i++){
        IL_mean=IL_mean+IL_buffer[i];
    }
    IL_mean=IL_mean>>4;
    U2TXREG = Vi>>8;
    while(U2STAbits.TRMT || U2STAbits.UTXBF);
    U2TXREG = Vi;
    while(U2STAbits.TRMT || U2STAbits.UTXBF);
    U2TXREG = IL_mean>>8;
    while(U2STAbits.TRMT || U2STAbits.UTXBF);
    U2TXREG = IL_mean;
    while(U2STAbits.TRMT || U2STAbits.UTXBF);
    U2TXREG = Wm>>8;
    while(U2STAbits.TRMT || U2STAbits.UTXBF);
    U2TXREG = Wm;
    while(U2STAbits.TRMT || U2STAbits.UTXBF);
    while(U2STAbits.URXDA==0);
    Wm_aux = U2RXREG;
    Wm_aux = Wm_aux<<8;
    while(U2STAbits.URXDA==0);
    Wm_ref = Wm_aux + U2RXREG;
    IFS1bits.T5IF=0; // CLEAR FLAG
    return;
}
//
void _ISRNPVSV _U2TXInterrupt (void)
{
    IFS1bits.U2TXIF=0;// CLEAR FLAG
    return;
}
//
void _ISRNPVSV _U2RXInterrupt (void)
{
    IFS1bits.U2RXIF=0;// CLEAR FLAG
    return;
}
}

```

init.c (WECS)

```

//*****WECS*****
#include "global.h"
/**** -----Função de configurações gerais do dsPIC----- ****/
void iniciarPIC(void){
    while(!OSCCONbits.LOCK);
    TRISB = 0xFFFF; // Porta B como entrada
    TRISC = 0xFFFF; // Porta C como entrada
    TRISD = 0xFFFF; // Porta D como entrada
    TRISE = 0xFFFF; // Porta E como entrada
    TRISF = 0xFFFF; // Porta F como entrada
    TRISBbits.TRISB7 = 1; // Pino B7 como entrada (Vi)
    TRISBbits.TRISB8 = 1; // Pino B8 como entrada (IL)
}

```

init.c (WECS)

```
TRISEbits.TRISE2 = 0; // Pino D3 como saída (LED)
TRISEbits.TRISE3 = 1; // Pino E3 como entrada (QEI_A)
TRISEbits.TRISE4 = 1; // Pino E4 como entrada (QEI_B)
TRISEbits.TRISE1=0; // Pino E1 como saída (PWM)
TRISFbits.TRISF4 = 1; // Pino F4 como entrada (Rx)
TRISFbits.TRISF5 = 0; // Pino F5 como entrada (Tx)
CNPU1 = 0x0000; // Resistores de Pull Up desabilitados
CNPU2 = 0x0000;
//INTCON2bits.ALTIPT = 0; // habilita ISR's de falha
return;
}
//
/**/ -----Função de configurações do módulo PWM----- ***/
void iniciarPWM(void){
    PTCON = 0x0003; // PWM desabilitado, continua funcionando em hibernação
                // postscale de 1:1, prescale de 1:1
                // modo contínuo up/down
    PTMR = 0x0000; // Limpando o registrador de contagem do PWM
    PTPER = ptper; // Frequência precisa de 20kHz
    SEVTCMP = 0x0000; // special event quando a contagem é crescente
                // para um valor de PTMR = 0x0000
    SEVTCMPbits.SEVTCMP = ptper;
    SEVTCMPbits.SEVTDIR = 0;
    PWMCON1 = 0x0F00; // Pares de PWM em modo independente
    PWMCON1bits.PEN1H=1;// Pino do PWM1 H1 é controlados pelo módulo PWM
    //PWMCON1bits.PEN2L=1;// Pino do PWM1 L2 é controlados pelo módulo PWM
    //PWMCON1bits.PEN2H=1;// Pino do PWM1 H2 é controlados pelo módulo PWM
    PWMCON2 = 0x0000; // Postscale de special event de 1:1
    PWMCON2bits.IUE = 0;// Atualização da razão cíclica sincronizada
    PWMCON2bits.OSYNC = 1;
    DTCON1 = 0x0000; // Prescale do tempo morto 1:1
                // Sem tempo morto
    PTCONbits.PTEN = 1; // habilita modulo PWM
    return;
}
//
/**/ -----Função de configurações do módulo A/D----- ***/
void iniciarADC(void){
    ADCON1 = 0x026C; /*ADCON1 = 0x0064;*/ // módulo A/D desabilitado
                // modo contínuo de operação durante hibernação
                // formato Fracionado sem sinal (dddd dddd dd00 0000)
                // trigger com o módulo PWM
                // amostragem simultanea dos canais CH0 (apenas)
                // amostragem inicia logo após a última conversão
    ADCON2 = 0x0300; /*ADCON2 &= 0;*/
                // referências AVDD (VrefH) e AVSS (VrefL)
                // como SIMSAM = 1
                // canais selecionados: CH0,CH1,CH2,CH3
                // somente válido se o buffer não for de 16 bits
                // interrupção para cada 1 sequência de amostragens
                // buffer de 16 bits
                // utiliza sempre o multiplexador A
    ADCON3bits.SAMC = 1; // tempo de amostragem automática de 1 Tad
                // 1 TAD (varia de 1 até 64)
    ADCON3bits.ADRS = 0; // fonte de clock: derivado do sistema de clock
    ADCON3bits.ADCS = 2; // TAD mínimo de 84 ns, ADCS = 9(TAD = 2*50ns = 1us)
                // Pag. 218 Datasheet dsPIC30F4011
    ADCHS = 0x0000; // não possui MUX B, zerando os registradores
                // referência negativa (Vref-) para os canais
                // CH1 a CH3
                // referência negativa (Vref-) para o canal CH0
    ADCHSbits.CH123SA = 0; // CH1 no pino AN0
                // CH2 no pino AN1
```

```
init.c (WECS)
```

```
        // CH3 no pino AN2
ADCHSbits.CH0SA = 3; // CH0 no pino AN3
ADPCFG |= 0xFFFF; // todos os pinos digitais
ADPCFGbits.PCFG0 = 0; // pino AN0 analógico
ADPCFGbits.PCFG1 = 0; // pino AN1 analógico
IFS0bits.ADIF = 0; // zera flag da interrupção
IPC2bits.ADIP = PRI_AD; // prioridade da interrupção
IEC0bits.ADIE = 1; // habilita interrupção do módulo AD
ADCON1bits.ADON = 1; // habilita módulo AD
return;
}
//
void iniciarUART(void){
    U2MODE = 0x00; // desabilita o modulo UART
                // 8-bits / sem bit de paridade/ 1 stop bit
    U2STA = 0x20C0; // Interrupção quando o último envio acaba
                // Interrupção de Rx apos receber 4 bytes
    U2BRG = BaudGen; // Configura o gerador de Baud Rate
    U2MODEbits.UARTEN = 1; // habilita o modulo UART
    U2STAbits.UTXEN = 1; // habilita Tx
    return;
}
//
void iniciarQEI(void)
{
    POSCNT = 0; // Reset position counter
    ADPCFG |= 0x0038; // força pinos INDEX, QEA, QEB a serem entradas digitais
//----- QEICON REGISTER-----//
    QEICONbits.CNTERR = 0; // BIT <15> - clear any count errors
    QEICONbits.QEISIDL = 0; //BIT <13> - Stop in Idle Mode bit-desabilitado
                // quando no modo IDLE
    QEICONbits.UPDN = 1; //BIT <11> - POSITION COUNTER DIRECTION STATUS
                // BIT - obs.: somente leitura quando
                // habilitado o QEI
    QEICONbits.QEIM = 0; //BIT <10:8> - modos do QEI / QEI desabilitado
    QEICONbits.SWPAB = 0; //BIT <7> - Phase A and Phase B Input Swap
                // Select bit - entradas não trocadas
    QEICONbits.PCDOUT = 0; //BIT <6> - normal i/o pin operation - datasheet
                // do 2010 nao tem esse bit
    QEICONbits.POSRES = 0; //BIT <2> - POSITION COUNTER RESET ENABLE BIT -
                // POSRES = 0 - index pulse does not
                // reset position couter
    QEICONbits.TQCS = 1; //BIT <1> - Timer Clock Source Select bit
    QEICONbits.UPDN_SRC = 0; //BIT <0> - WHEN CONFIGURED QEI MODE, THIS
                // CONTROL BIT IS A DON'T CARE
                // obs.: datasheet nomeia como UPDN_SRC
//----- DFLTCON REGISTER-----//
    DFLTCONbits.CEID = 1; // count error interrupts disabled
    DFLTCONbits.QEOUT = 1; // digital filters output enable for QEn pins
    DFLTCONbits.QECK = 3; // 1:16 clock divide for digital filter for QEI
    POSCNT = 0;
    //MAXCNT = 20; //Encoder operando com numero de pulsos fixo
    MAXCNT = numero_de_pulsos_por_volta*2; //Encoder operando em tempo fixo
    _QEIP = PRI_QEI;
    _QEIIF = 0;
    _QEIIIE = 1;
    QEICONbits.QEIM = 5; // modo de operação em 2x com maxcnt resetando contagem
                // e gerando interrupção
    return;
}
//
void iniciarTMR1(void)
```

init.c (WECS)

```
{
//CONFIGURANDO O REGISTRADOR DE CONTROLE DO TIMER1
T1CONbits.TON = 0; //desligando o timer pra poder config
T1CONbits.TCS = 0;
T1CONbits.TGATE = 0; //nessa linha e na de cima eu configura operação "timer"
T1CONbits.TCKPS = 0; //prescale de 1:1
TMR1 = 0; //limpando a merda do registrador do timer
PR1 = 0xFFFF; // carrega o limite da contagem com 65535, morô?
//CONFIGURANDO A PORRA DA INTERRUPÇÃO DO MALUCO
IPC0bits.T1IP = PRI_TMR1; //define a merda com maior prioridade
IFS0bits.T1IF = 0; //limpando a flag da interrup do T1
IEC0bits.T1IE = 1;
return;
}
//
void iniciarTMR2(void)
{
//CONFIGURANDO O REGISTRADOR DE CONTROLE DO TIMER1
T2CONbits.TON = 0; //desligando o timer pra poder config
T2CONbits.TCS = 0;
T2CONbits.TGATE = 0; //nessa linha e na de cima eu configura operação "timer"
T2CONbits.TCKPS = 0b10;
//0x00 - prescale de 1:1
//0x01 - prescale de 1:8
//0x10 - prescale de 1:64
//0x11 - prescale de 1:256
TMR2 = 0; //limpando a merda do registrador do timer
PR2 = 3125; // (1:1 - 20000:1ms) // (1:64) - 3125:10ms
//CONFIGURANDO A INTERRUPÇÃO
IPC1bits.T2IP = PRI_TMR2; //define a merda com maior prioridade
IFS0bits.T2IF = 0; //limpando a flag da interrup do T1
IEC0bits.T2IE = 1;
T2CONbits.TON = 1; // Liga Timer
return;
}
//
void iniciarTMR3 (long int frequencia, long int prescale, int start)
{
//----->Inicialização<-----//
    T3CONbits.TON = 0; //desligado
    PR3 = FCY/(prescale*frequencia);
    TMR3 = 0;
    T3CONbits.TGATE = 0;
    //T1CONbits.T32 = 0;
    T3CONbits.TCS = 0;
//----->Interrupções<-----//
    _T3IP = PRI_TMR3;
    IFS0bits.T3IF = 0;
    IEC0bits.T3IE = 1;
//----->Prescale<-----//
    if (prescale == 1) //Atribuindo time base input prescale
        T3CONbits.TCKPS = 0b00;
    else if (prescale == 8)
        T3CONbits.TCKPS = 0b01;
    else if (prescale == 64)
        T3CONbits.TCKPS = 0b10;
    else if (prescale == 256)
        T3CONbits.TCKPS = 0b11;
    else
        T3CONbits.TCKPS = 0b00;
//----->Start<-----//
    T3CONbits.TON = start;
return;
}
```

init.c (WECS)

```
}
//
void iniciarTMR45(void)
{
    //CONFIGURANDO O REGISTRADOR DE CONTROLE DO TIMER2
    T4CONbits.TON = 0; //desligando o timer pra poder fazer as config no TMR4
    T5CONbits.TON = 0; //desligando o timer pra poder fazer as config no TMR5
    T4CONbits.T32 = 1; //Contador de 32 bits
    T4CONbits.TCS = 0; //configuração pra operar no modo "timer"
    T4CONbits.TGATE = 0;
    T4CONbits.TCKPS = 0b00; //prescale de 1:1
    TMR4 = 0; //limpando o registrador do timer
    TMR5 = 0; //limpando o registrador do timer
    TMR5HLD = 0; //limpando o registrador do timer
    // interupcao a cada 0.1s
    //PR5 = 0x001E;
    //PR4 = 0x8480;
    // interupcao a cada 0.05s
    //PR5 = 0x000F;
    //PR4 = 0x4240;
    // interupcao a cada 0.01s
    PR5 = 0x0003;
    PR4 = 0x0D40;
    //CONFIGURANDO A INTERRUPÇÃO DO TMR5
    IPC5bits.T5IP = PRI_TMR45; //seta prioridade do TMR45
    IFS1bits.T5IF = 0; //limpando a flag da interrup do T5
    IEC1bits.T5IE = 1; //habilita interrupção do TMR45
    //Ligando
    T4CONbits.TON = 1; //ligando o TMR4
    T5CONbits.TON = 1; //ligando o TMR5
    return;
}
//
void delay_ms (long int x)
{
    unsigned long int cont_break;
    tmr3_cont = 0;
    cont_break = x*m_base;
    while (tmr3_cont != cont_break);
}
//
void delay_us (long int x)
{
    unsigned int dj,dnop;
    x *= cyc;
    for (dj=0;dj<=x;dj++)
    {
        dnop++;
    }
}
}
```

main.c (WECS)

```
/**
*****WECS*****
/*****
* UNIVERSIDADE FEDERAL DO CEARÁ
* CENTRO DE TECNOLOGIA
* DEPARTAMENTO DE ENGENHARIA ELÉTRICA
* GRUPO DE PROCESSAMENTO DE ENERGIA E CONTROLE (GPEC)
*
*
* FileName:    main.c
* Processor:   dsPIC30F2010
**/
```

main.c (WECS)

```
* Compiler:    MPLABX v3.10 & XC16 v1.25 or higher
* Authors:     Jorge e Allan
*
* REVISION HISTORY:
* ~~~~~
* Author      Date    Comments on this revision
* ~~~~~
* Jorge       00/01/15  Início do código
*
* ~~~~~
*
* ADDITIONAL NOTES:
*
*****/
#include "global.h"
_FOSC(CSW_FSCM_OFF & XT_PLL8);
_FWDT(WDT_OFF);
_FGS(CODE_PROT_OFF);
_FBORPOR(PWMxH_ACT_HI & PBOR_OFF & MCLR_EN & BORV45 & RST_IOPIN);
/** -----Declaração de variáveis----- **/
// PI Control
/* X space 0x0800 a 0x0BFF
 * Y space 0x0C00 a 0x0FFF
 * 30F2010 exclusivo!
 */
volatile int_fast16_t _XDATA(16) CONST_PI[3];
volatile int_fast16_t _YDATA(16) CONTROL_PI[3];
volatile int_fast16_t _XDATA(16) PI_ROTACAO[3];
volatile int_fast16_t _YDATA(16) CONTROL_ROTACAO[3];
volatile int_fast16_t _XDATA(16) PI_CORRENTE[3];
volatile int_fast16_t _YDATA(16) CONTROL_CORRENTE[3];
// Generic
volatile int_fast16_t Vi, IL, IL_ref, Wm, Wm_old, Wm_ref, Wm_aux;
volatile int_fast16_t OUTPUT_CORRENTE;
volatile int_fast16_t OUTPUT_ROTACAO;
volatile int_fast16_t MIN_DUTY=min_duty, MAX_DUTY=max_duty;
volatile int_fast16_t MIN_IL=min_corrente, MAX_IL=max_corrente;
volatile uint_fast8_t flag_control=0,cont_control=0, cont_uart=0;
volatile uint_fast8_t flag_referencia, degrau_referencia = 0;
volatile uint_fast24_t tmr3_cont;
// QIE
volatile int_fast16_t timer_buff, cont_buff, cont, refresh, voltas;
volatile int_fast16_t pulsos, pulsos_buffer;
// Média Corrente
volatile uint_fast16_t IL_buffer[16];
volatile uint_fast8_t cont_IL_mean=0;

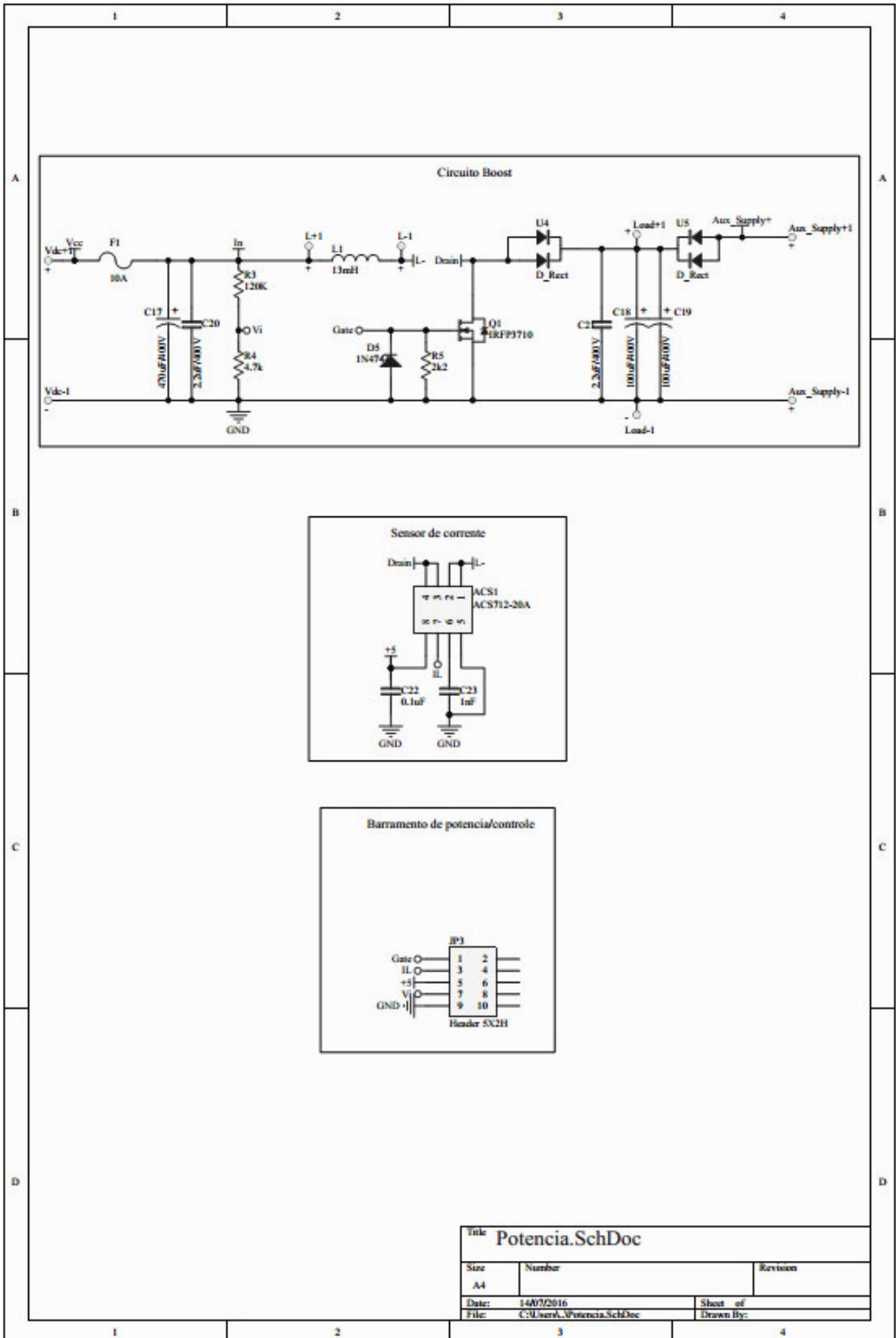
int main(void) {
    // Boot
    iniciarPIC();          // Configuração de pinagem
    setLed();
    iniciarTMR3(f_base,8,1); // Ligar funções 'delay'
    iniciarADC();          // Configuração do módulo A/D
    iniciarPWM();          // Configuração do módulo PWM
    iniciarUART();         // Configuração do módulo UART
    iniciarTMR45();        // Configuração do timer de segundos
    iniciarQEI();          // inicia encoder
    iniciarTMR1();         // Configuração do timer de segundos
    iniciarTMR2();         // Configuração do timer de segundos

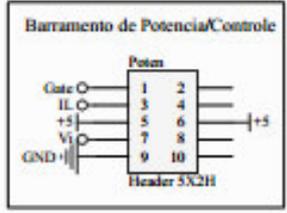
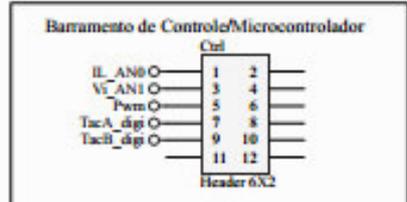
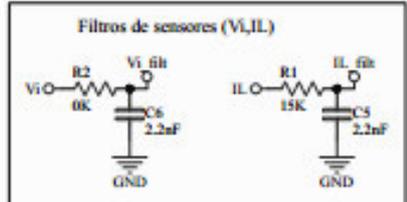
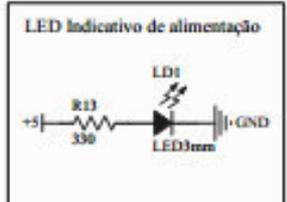
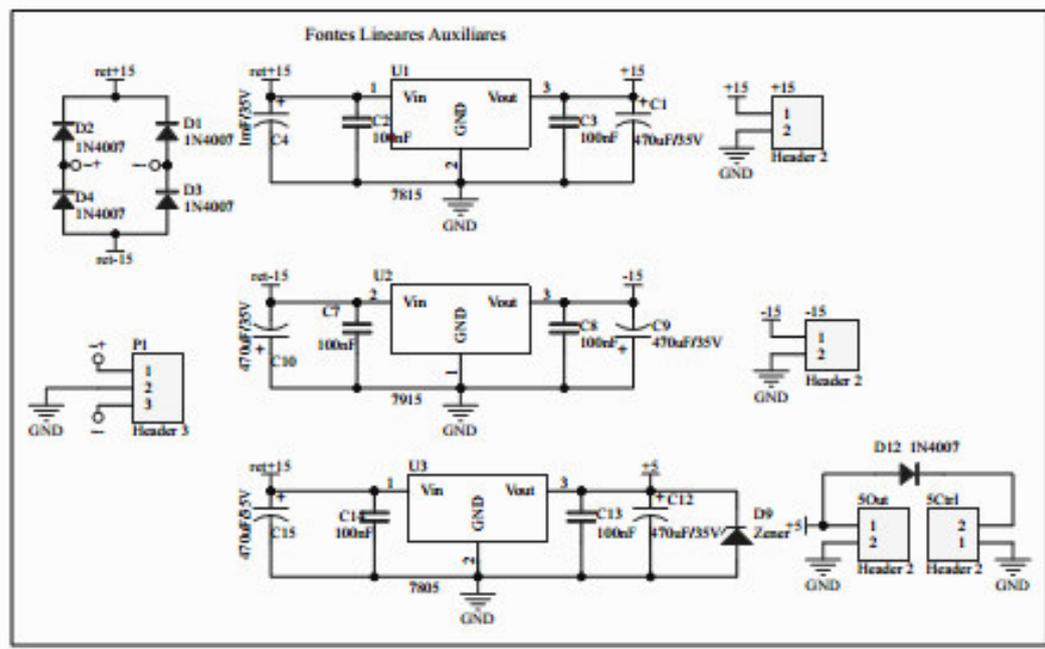
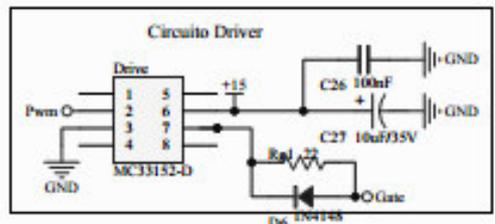
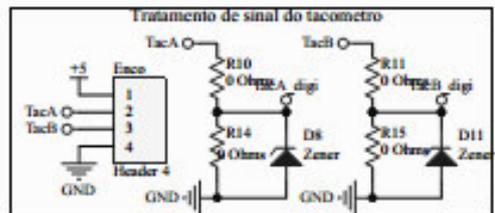
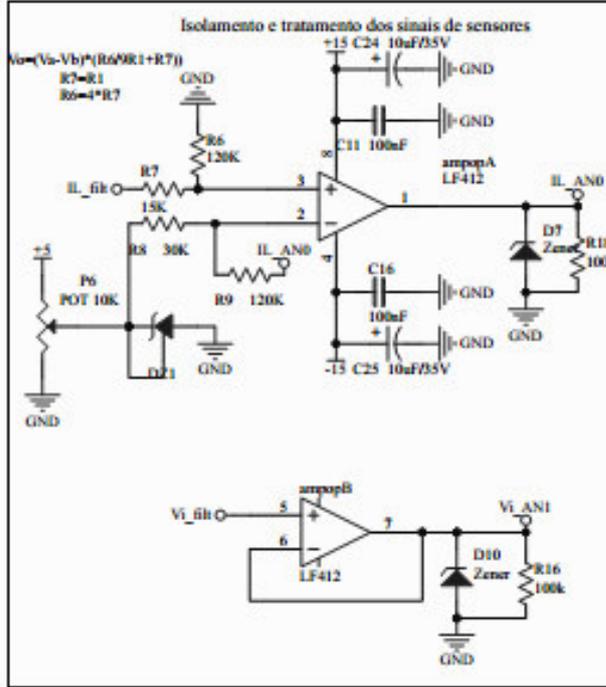
    clrLed();
    // computa inteiros sinalizados e ativa saturação dos acumuladores A e B
    CORCON |= 0x00F1;
```

main.c (WECS)

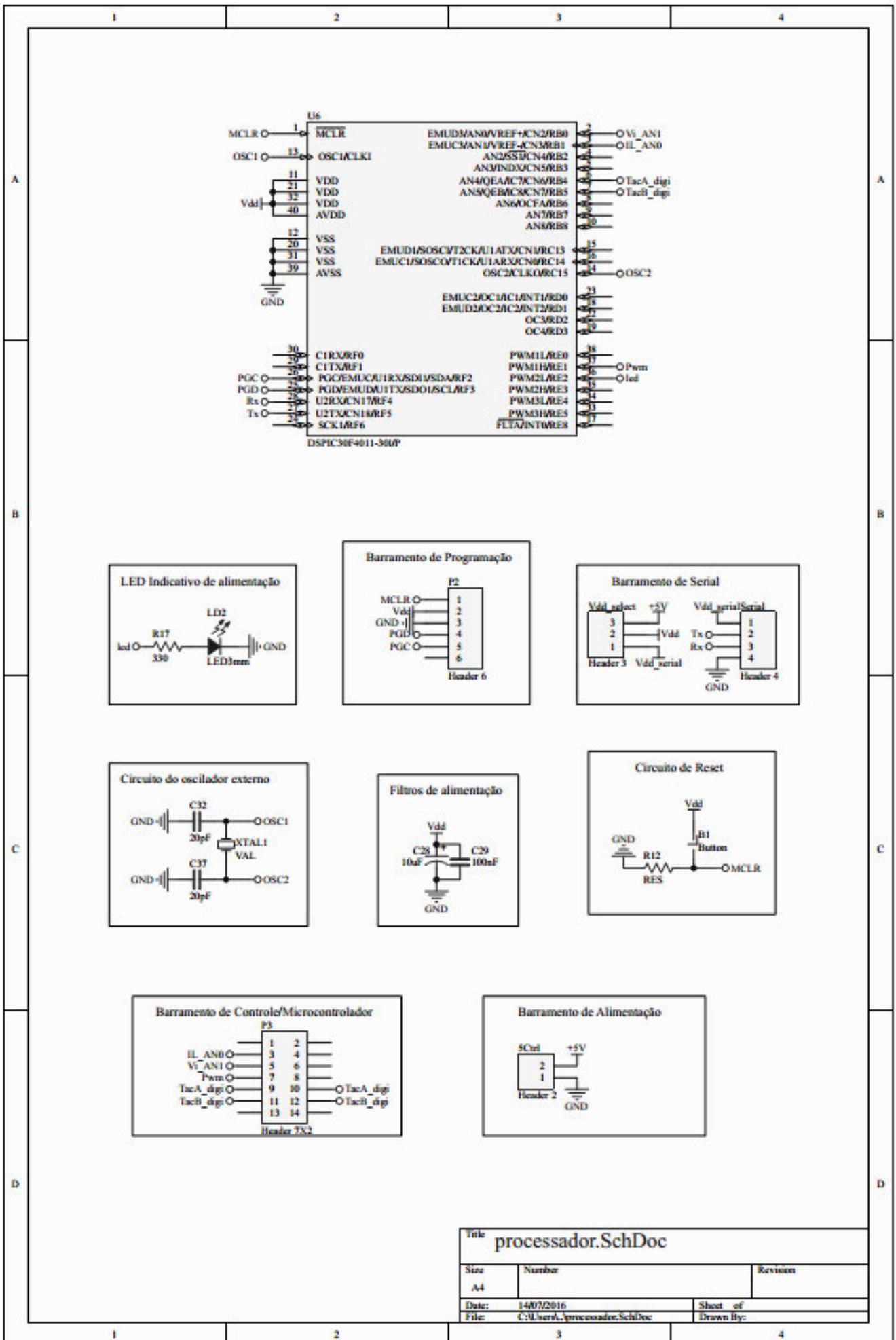
```
PDC1=duty_50;
Wm = 0;
Vi = 0;
IL = 0;
IL_ref = Corrente_2A;
Wm_ref = 200<<5;          //200 RPM
PI_CORRENTE[0] = 3095;
PI_CORRENTE[1] = -2853;
PI_CORRENTE[2] = 32767;
/* 2kHz - 70°
   num: {[0.0945 -0.0871]}
   den: {[1 -1]}
   c1: 3.0955e+03
   c2: -2.8532e+03
   c3: 32767
*/
PI_ROTACAO[0] = -12904 ;      // e(k)
PI_ROTACAO[1] = (32768 - 4122) ; // w(k-1)
PI_ROTACAO[2] = 4122;      // u(k-1)
/* 5Hz - 70° (x32)
   c1a: -1.2904e+04 // -12904
   c2a: 4.1222e+03 // 4122
*/
CONST_PI[0] = PI_CORRENTE[0];
CONST_PI[1] = PI_CORRENTE[1];
CONST_PI[2] = PI_CORRENTE[2];
while(1);
return 0;
}
```

APÊNDICE E: ESQUEMÁTICOS DO BOOST

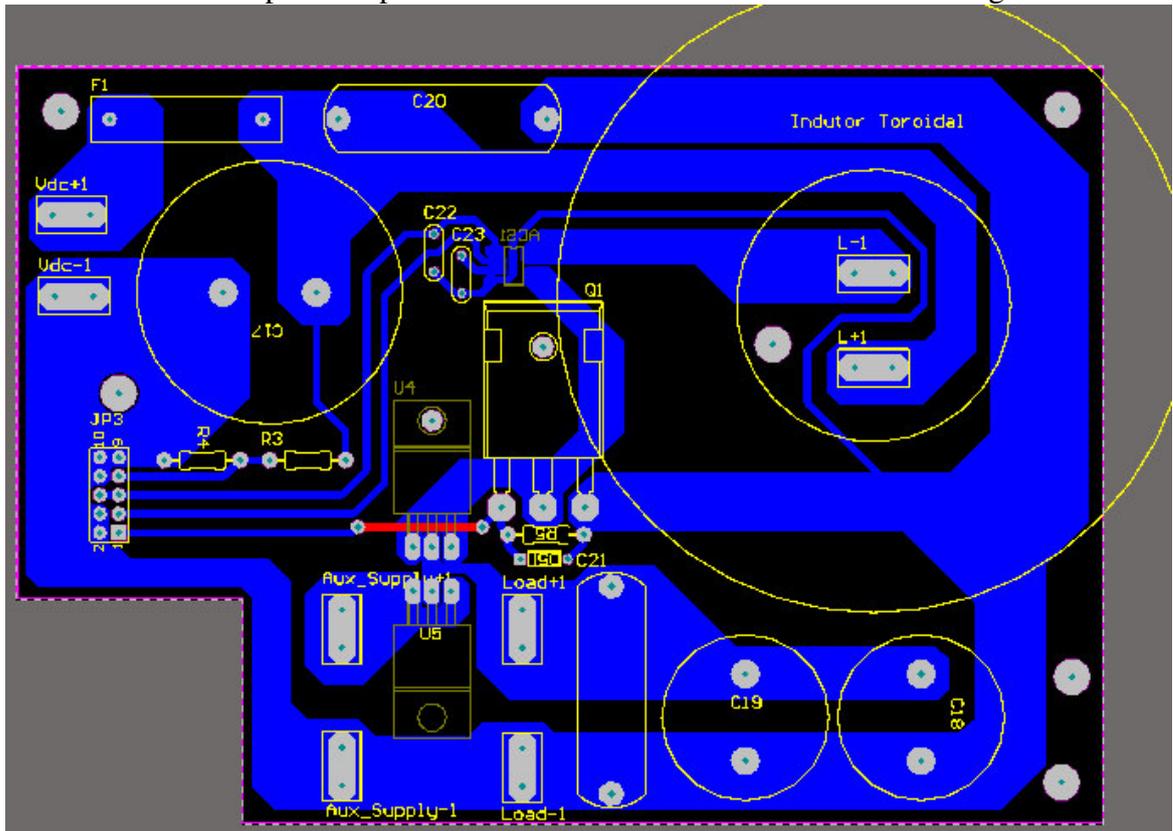




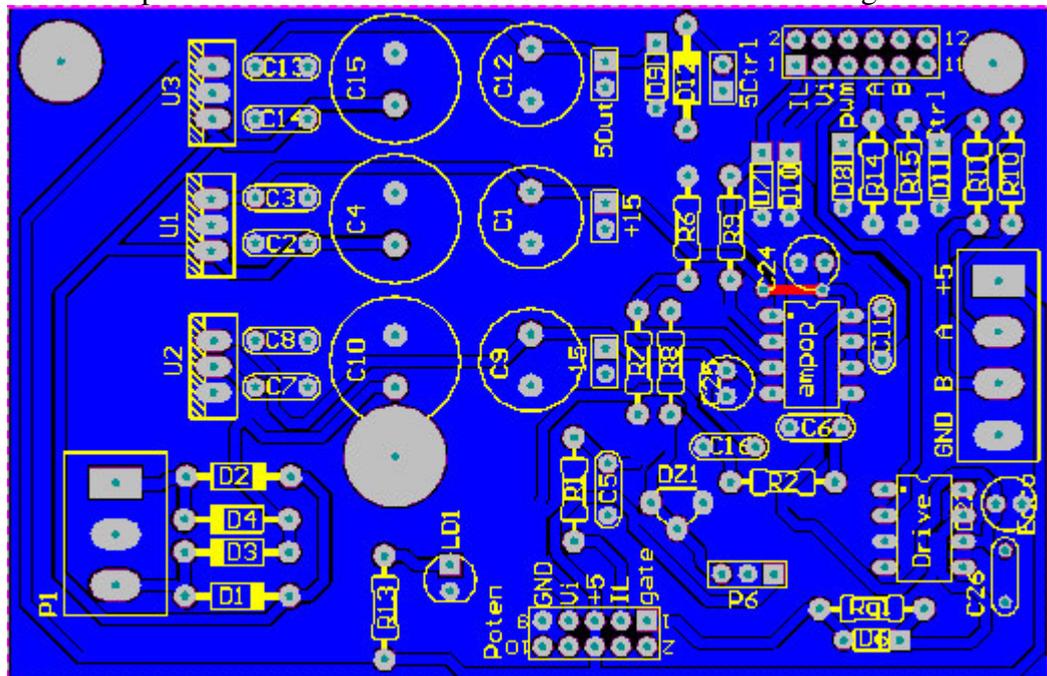
Title Controle.SchDoc		
Size A4	Number	Revision
Date: 14/07/2016	Sheet of	
File: C:\Users\... \Controle.SchDoc	Drawn By:	



PCB da placa de potência do conversor Boost controlador de carga



PCB da placa de controle do conversor Boost controlador de carga



PCB da placa do processador do conversor Boost controlador de carga

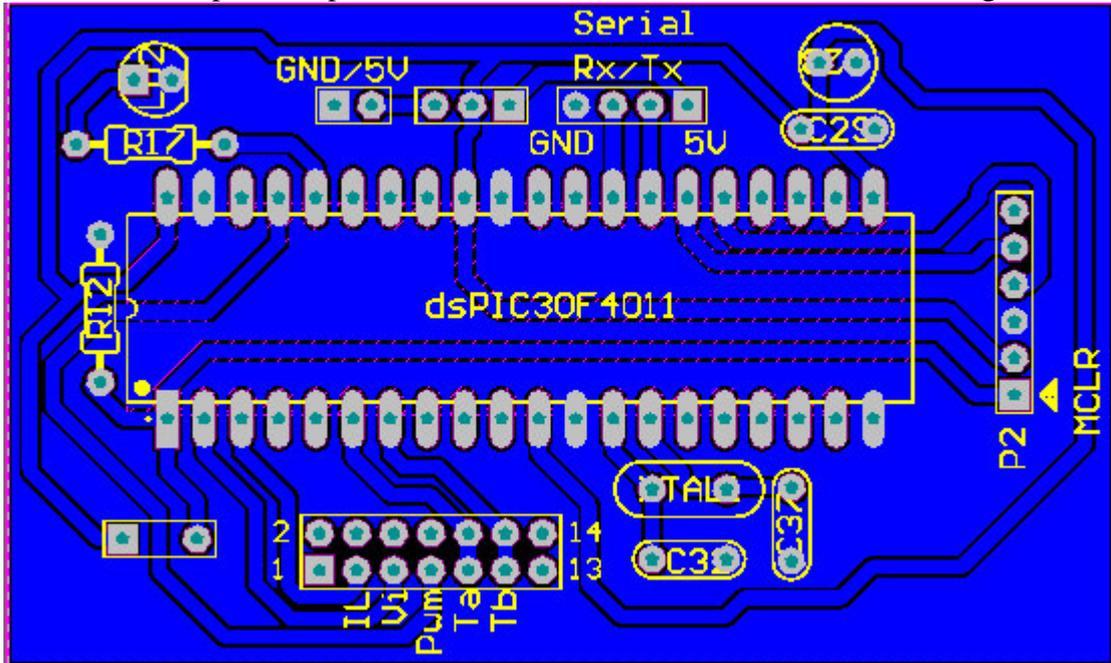
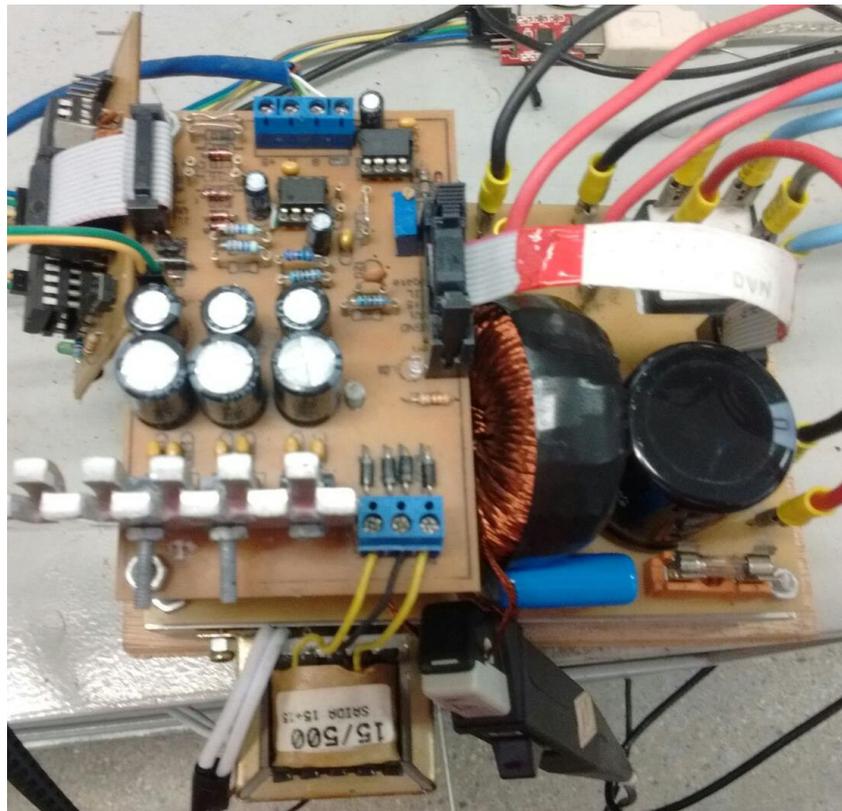


Foto do conversor Boost finalizado



APÊNDICE F: CÓDIGOS DE COMUNICAÇÃO MATLAB

Comunicação RS-232 – dsPIC30F / Matlab

-controle.m

-communicate.m

-plotter.m

```
controle.m
%% CONTROLE
clear all; close all; clc
addpath('C:\Users\Jorge Wattes\Google
Drive\UFC\Mestrado\Dissertação\Simulações')
addpath('C:\Users\Jorge Wattes\Google
Drive\UFC\Mestrado\Dissertação\Simulações')
run('Boot.m')
load('CACLA_WEIGHTH.mat')
load('NASPO_WEIGHTH.mat')
close all

%abrindo porta
TIMEOUT = 2;
porta = serial('COM6', ...
              'BaudRate', 250000, ...
              'Parity', 'none', ...
              'DataBits', 8, ...
              'StopBits', 1); %Change depending on mbed configuration
set(porta, 'Timeout', TIMEOUT); %Set Timeout
fopen(porta); %Open serial connection

%Inicialização de variáveis
stop=0;
b =
uicontrol('style','push','string','stop','callback','stop=stop+1','Position', [20 10 40 20]);
c = uicontrol('style','push','string','-100','callback','Wm_ref=Wm_ref-100','Position', [20 30 40 20]);
d =
uicontrol('style','push','string','+100','callback','Wm_ref=Wm_ref+100','Position', [20 50 40 20]);
% e =
uicontrol('style','push','string','700rpm','callback','Wm_ref=700','Position', [20 70 40 20]);
% f =
uicontrol('style','push','string','750rpm','callback','Wm_ref=750','Position', [20 90 40 20]);
% g =
uicontrol('style','push','string','800rpm','callback','Wm_ref=800','Position', [20 110 40 20]);
% h =
uicontrol('style','push','string','850rpm','callback','Wm_ref=850','Position', [20 130 40 20]);
n=0;
Vi = 0;
IL = 0;
Pe = 0;
Wm = 0;
Wm_ref = 650;
Wm_old = Wm_ref;
Vi_h = Vi;
IL_h = IL;
Pe_h = Pe;
```

controle.m

```
Wm_h = Wm;
Wm_ref_h = Wm_ref;
%Variáveis neurais
Act_t=0;
Act_a=0;
r=0;
Vs=0;
rh=0;
y=0;
Act_t_h=Act_t;
Act_a_h=Act_a;
r_h=r;
Vs_h=Vs;
rh_h=rh;
y_h=y;

time=0;
inte=0;
ctrl=0;
updw=0;
tic;
% Iterações
while (stop<1)

    % Comunicação com o controlador (dsPIC30F4011)
    [Vi,IL,Wm] = communicate(Wm_ref*32,porta);
    tempo = toc;
    Vi = Vi/(Boost.Hadc*Boost.Hv);
    Vi = 0.1*Vi+0.9*Vi_h(length(Vi_h));
    IL = IL/(Boost.Hadc*Boost.Hi);
    IL = 0.1*IL+0.9*IL_h(length(IL_h));
    Wm = Wm/32;
    Wm = 0.2*Wm+0.8*Wm_h(length(Wm_h));
    Pe = Vi*IL;
    Pe = 0.1*Pe+0.9*Pe_h(length(Pe_h));

    %Registro de dados
    Vi_h = [Vi_h, Vi];
    IL_h = [IL_h, IL];
    Pe_h = [Pe_h, Pe];
    Wm_h = [Wm_h, Wm];
    Wm_ref_h = [Wm_ref_h, Wm_old];
    time = [time, tempo];%inte*Tsampl];
    Wm_old=Wm_ref;

    % Controle de MPPT
    if (tempo>=n*MPPT.period)
        n=n+1;

%% Rampa
%         if (updw==0)
%             %Wm_ref=Wm_ref+5;
%             if (Wm_ref>=1000)
%                 updw=1;
%             end
%         else
%             %Wm_ref=Wm_ref-5;
%             if (Wm_ref<=200)
%                 updw=0;
%             end
%         end
%% Degraus
%         if (Wm_ref==400)
```

controle.m

```
%
    Wm_ref=500;
%
    elseif(Wm_ref==500)
%
    Wm_ref=600;
%
    elseif(Wm_ref==600)
%
    Wm_ref=700;
%
    elseif(Wm_ref==700)
%
    Wm_ref=800;
%
    elseif(Wm_ref==800)
%
    Wm_ref=900;
%
    else
%
    Wm_ref=400;
%
    end

%% MPPT
    %% Wm_ref = MPPT(Vi, IL, Wm)
    %Fixo
    %Wm_ref=650;
    %P&O
    Wm_ref = PerturbObserve (Pe, Wm, WECS.RPM_max, WECS.RPM_min);
    %NASPO
    %[Wm_ref, r, rh, y] = NASPO (Pe, Wm, WECS.RPM_max, WECS.RPM_min);
    %CACLA
    %[Wm_ref, Act_t, Act_a, r, Vs, wh_act, wo_act, wh_crt, wo_crt] =
CACLA (Pe, Wm, WECS.RPM_max, WECS.RPM_min, wh_act, wo_act, wh_crt, wo_crt);

    Act_t_h=[Act_t_h, Act_t];
    Act_a_h=[Act_a_h, Act_a];
    r_h=[r_h, r];
    Vs_h=[Vs_h, Vs];
    rh_h=[rh_h, rh];
    y_h=[y_h, y];

    %Observação dos dados
    clc
    plotter(Vi_h, IL_h, Pe_h, Wm_h, Wm_ref_h, time)

end

end

%Finalização
plotter(Vi_h, IL_h, Pe_h, Wm_h, Wm_ref_h, time)
save('experimental.mat', 'Vi_h', 'IL_h', 'Pe_h', 'Wm_h', 'Wm_ref_h', 'time', 'Ac
t_t_h', 'Act_a_h', 'r_h', 'Vs_h', 'rh_h', 'y_h')
[Vi, IL, Wm] = communicate(0, porta);
fclose(porta);
```

communicate.m

```
function [Vi, IL, Wm]=communicate(Wm_ref, porta)

try
    Wm_ref_MSB = uint8(floor(Wm_ref/(2^8)));
    Wm_ref_LSB = uint8(Wm_ref - Wm_ref_MSB*(2^8));
    fwrite(porta, round(Wm_ref_MSB), 'uint8');
    fwrite(porta, Wm_ref_LSB, 'uint8');
    Vi=fread(porta, 1, 'uint8')*(2^8);
    Vi=Vi+fread(porta, 1, 'uint8');
    IL=fread(porta, 1, 'uint8')*(2^8);
    IL=IL+fread(porta, 1, 'uint8');
    Wm=fread(porta, 1, 'uint8')*(2^8);
    Wm=Wm+fread(porta, 1, 'uint8');
catch %in case of error or Serial being disconnected
```

```
communicate.m
```

```
disp('Communication failed!');  
end
```

```
plotter.m
```

```
function plotter(Vi_h, IL_h, Pe_h, Wm_h, Wm_ref_h, time)  
  
    vento = 7;  
    Pmax_h=[];  
  
    %Gráficos  
    figure(1)  
    subplot(2,2,1)  
    plot(time,Vi_h)  
    axis('tight')  
    ylabel('Tensão [V]');  
    xlabel('tempo [s]');  
    subplot(2,2,2)  
    plot(time,IL_h)  
    axis tight  
    ylabel('Corrente [A]');  
    xlabel('tempo [s]');  
    subplot(2,2,3)  
    plot(time,Pe_h)  
    hold on  
    %    x = (2*pi/60)*0.69*Wm_h/vento;  
    %    cp = 0.5*(116*(1./x - 0.035/1)-5).*exp(-21*(1./x - 0.035/1)) +  
    0.01*x;  
    %    Pmax = max(cp*0.5*(0.69^2)*1.229*pi*vento^3,0);  
    %    Pmax_h=[Pmax_h;Pmax]  
    %    plot(time,Pmax_h,'red')  
    hold off  
    axis tight  
    ylabel('Potência [W]');  
    xlabel('tempo [s]');  
    subplot(2,2,4)  
    hold off  
    plot(time,Wm_ref_h,'red--')  
    hold on  
    plot(time,Wm_h,'blue')  
    axis tight  
    ylabel('Rotação [RPM]');  
    xlabel('tempo [s]');  
    axis tight  
    drawnow  
  
    Vi=(Vi_h(length(Vi_h)));  
    IL=IL_h(length(IL_h));  
    Wm=Wm_h(length(Wm_h));  
    Wm_ref=Wm_ref_h(length(Wm_ref_h));  
    tempo=time(length(time));  
    rot=Wm/1.4737;  
    clc  
    fprintf('Vi=%3.2f[V], IL=%2.2f[A], Wm=%5.2f[RPM], Wm_ref=%5.2f[RPM],  
    Tempo=%3.2f[s]',Vi,IL,Wm,Wm_ref,tempo);
```

APÊNDICE G: CÓDIGOS DE MPPT MATLAB

Funções de MPPT - Matlab

-PerturbObserve.m

-NASPO.m

-CACLA.m

PerturbObserve.m

```
function [Wm_ref_val] = PerturbObserve(Pe,Wm,Wm_max,Wm_min)

%% Inicialização
persistent Pe_old dWm dPe_min Wm_ref
if isempty(Pe_old)
    %Parametros
    Pe_old = 0;
    %dWm=50; %Simulação
    dWm=10; %experimental
    dPe_min=0;
    Wm_ref = Wm_min;
end

%% Iterações
% [next step]
if (Wm<=Wm_min)
    Wm_ref=Wm_min*1.05;
elseif (Wm>=Wm_max)
    Wm_ref=Wm_max*0.95;
else
    dPe=Pe-Pe_old;
    Pe_old = Pe;
    if(abs(dPe)<dPe_min)
        Wm_ref = Wm;
    elseif(dPe>dPe_min)
        Wm_ref = Wm+dWm;
    elseif(dPe<-dPe_min)
        dWm = -dWm;
        Wm_ref = Wm + dWm;
    end
end

Wm_ref_val=Wm_ref;
```

NASPO.m

```
function [Wm_ref_val, r_val, rh_val, y_val,wh_,wo_] =
NASPO(Pe,Wm,Wm_max,Wm_min)

% Parametros
step = 50;
h_size = 15;      %Neurônios na camada oculta
i_size = 2;      %Neurônios na camada de entrada
o_size = 1;      %Neurônios de saída
nabla = 0.02;    %Coeficiente de aprendizagem(nabla)
dPe_wm=1;
dPe_wind=15;
hist_buffer=5;

%% Inicialização
persistent wh wo Pe_old y d Wm_ref r p erro Pe_Wm_hist rh
if isempty(wh)

    %% Inicialização dos pesos, elegibilidades e avaliação
```

NASPO.m

```
    %[Initialize weights w=0 and eligibilities e=0]
    %wh=wh_init;
    %wo=wo_init;
    wh = random('unif',0,0.1,[h_size,i_size+1]);
    wo = random('unif',0,0.1,[o_size,h_size+1]);
    Pe_old = Pe;
    r=0;
    rh=0;
    erro=0;
    Pe_Wm_hist=zeros(hist_buffer,2);
    p=[Pe;Wm];
    d = 1./(1+exp(-(wh*[p;1])));
    y = wo*[d;1];

    % Ajuste de referência
    Wm_ref=200;
end

%% Iterações
%[next step]
dPe=Pe-Pe_old;
Pe_old=Pe;
if(abs(dPe)>dPe_wind)
    Pe_Wm_hist=zeros(hist_buffer,2);
end
%Pe_Wm_hist(2:hist_buffer,:)=Pe_Wm_hist(1:(hist_buffer-1),:);
%Pe_Wm_hist(1,:)=Pe,Wm];
Pe_Wm_hist=[[Pe,Wm];Pe_Wm_hist(1:(hist_buffer-1),:)];

%% Calculo da recompensa instantanea
if (abs(dPe)<dPe_wm)
    r=0;
elseif (dPe>dPe_wm)
    if(y>=0)
        r=1;
    else
        r=-1;
    end
elseif (dPe<-dPe_wm)
    if(y>=0)
        r=-1;
    else
        r=1;
    end
end
end
%%Calculo da recompensa histórica
[v,i]=max(Pe_Wm_hist(:,1));
if(Wm<Pe_Wm_hist(i,2))
    rh=1;
elseif(Wm>Pe_Wm_hist(i,2))
    rh=-1;
else
    rh=0;
end
erro=(rh+r)-y;

%% Ajuste dos pesos com base no valor de saida anterior e atual das ANNs
% Calculo das sensibilidades
d_o=erro;
d_h=zeros(h_size,1);
for h=1:h_size
    d_h(h)=d_o*wo(1,h)*(d(h)*(1-d(h)));
end
```

NASPO.m

```
% Calculo dos ajustes
wo = wo + nabla*d_o*[d;1]';
wh = wh + nabla*d_h*[p;1]';

%% Iteração sináptica
 %[Record current Q-value, Q_t]
 p=[Pe;Wm];
 d = 1./(1+exp(-(wh*[p;1])));
 y = wo*[d;1];

%% Ajuste de referência
 Wm_ref=Wm+step*y;
 if(Wm_ref<=Wm_min)
     Wm_ref=Wm_min*1.05;
 elseif(Wm_ref>=Wm_max)
     Wm_ref=Wm_max*0.95;
 end

 Wm_ref_val = Wm_ref;
 r_val = r;
 rh_val= rh;
 y_val = y;
 Pe_val=Pe;
 wh_=wh;
 wo_=wo;
```

CACLA.m

```
function [Wm_ref_val,Act_t_val,
Act_a_val,r_val,Vs_val,wh_act_o,wo_act_o,wh_crt_o,wo_crt_o] =
CACLA(Pe,Wm,Wm_max,Wm_min,wh_act_i,wo_act_i,wh_crt_i,wo_crt_i)

% Parametros Actor
h_size_act = 15;    %Neurônios na camada oculta
i_size_act = 2;    %Neurônios na camada de entrada
o_size_act = 1;    %Neurônios de saída
% Parametros Critic
h_size_crt = 15;   %Neurônios na camada oculta
i_size_crt = 2;   %Neurônios na camada de entrada
o_size_crt = 1;   %Neurônios de saída
% Parametros de aprendizagem (TD)
alpha = 0.05;%0.05;    % Coeficiente de aprendizagem do Actor
beta = 0.98;          % Coeficiente de aprendizagem do Critic
gamma = 0.95;        % Coeficiente de amortização das avaliações futuras
do Critic
sigma=0.05;          % Coeficiente de aleatoriedade
epsilon=0.0;

% %Parametros do controle
% error_PI_max = 4;    %Diferença máxima admitida no erro do controle
PI, para verificação de saturação
%dPe_min = 1;        %Variação mínima de potência para observar
melhoria/piora na potência

%% Inicialização
persistent wh_act wo_act eh_act eo_act Act_t Act_a
persistent wh_crt wo_crt eh_crt eo_crt Vs Vs_old TD_error
persistent Wm_ref r Pe_old dPe dPe_min
if isempty(wh_act)
    %Parametros
    r=0;
    Pe_old = 0;
    dPe=0;
    Wm_ref=Wm_min;
```

CACLA.m

```

dPe_min=1;

%% Inicialização dos pesos, elegibilidades e avaliação
% Actor
wh_act = random('unif',-10,10,[h_size_act,i_size_act+1]);
wo_act = random('unif',-10,10,[o_size_act,h_size_act+1]);
%wh_act = wh_act_i;
%wo_act = wo_act_i;
eh_act = zeros(h_size_act,i_size_act+1);
eo_act = zeros(o_size_act,h_size_act+1);
% Critic
wh_crt = random('unif',-10,10,[h_size_crt,i_size_crt+1]);
wo_crt = random('unif',-10,10,[o_size_crt,h_size_crt+1]);
%wh_crt = wh_crt_i;
%wo_crt = wo_crt_i;
eh_crt = zeros(h_size_crt,i_size_act+1);
eo_crt = zeros(o_size_crt,h_size_act+1);

%Inicialização de referência
Act_a=0;
Act_t=0;
Vs=0;
Vs_old=0;
TD_error=0;

end

%% Iterações
% [next step]
%% Calculo da recompensa
dPe=Pe-Pe_old;
Pe_old = Pe;
if(abs(dPe)<dPe_min)
    r=0;
    %           dPe_min=max(dPe_min*0.9,1);
    %           sigma=max(sigma*0.5,0.02);
elseif(dPe>dPe_min)
    r=1;
    %           dPe_min=min(dPe_min/0.9,5);
    %           sigma=min(sigma/0.5,0.1);
elseif(dPe<-dPe_min)
    r=-1;
    %           dPe_min=min(dPe_min/0.9,5);
    %           sigma=min(sigma/0.5,0.1);
end

%% Ajuste do Critic
% Sinapse
in_crt = [Pe;Wm;1];
dh_crt = wh_crt*in_crt;
yh_crt = [(1./(1+exp(-dh_crt)))]'; %Função sigmoid f(x)=1/(1+e^-x)
Vs = wo_crt*yh_crt;           % do_crt=yo_crt=Vs
% Cálculo do TD
TD_error = (r + gamma*Vs - Vs_old);
% Ajuste (TD+BP)
wo_crt = wo_crt + beta*TD_error*eo_crt;
wh_crt = wh_crt + beta*TD_error*eh_crt;
% Cálculo da nova avaliação
in_crt = [Pe;Wm;1];
dh_crt = wh_crt*in_crt;
yh_crt = [(1./(1+exp(-dh_crt)))]'; %Função sigmoid f(x)=1/(1+e^-x)
Vs = wo_crt*yh_crt;           % do_crt=yo_crt=Vs
Vs_old = Vs;

```

CACLA.m

```

%% Ajuste do Actor
if(TD_error>0)
    wo_act = wo_act + alpha*(Act_t-Act_a)*eo_act;
    wh_act = wh_act + alpha*(Act_t-Act_a)*eh_act;
elseif(TD_error<0)
    wo_act = wo_act - alpha*(Act_t-Act_a)*eo_act;
    wh_act = wh_act - alpha*(Act_t-Act_a)*eh_act;
end
%     wo_act = wo_act + alpha*TD_error*(Act_t-Act_a)*eo_act;
%     wh_act = wh_act + alpha*TD_error*(Act_t-Act_a)*eh_act;
% Sinapse
in_act = [Pe;Wm;1];
dh_act = wh_act*in_act;
yh_act = [(1./(1+exp(-dh_act))));1]; %Função sigmoid f(x)=1/(1+e^-x)
Act_a = wo_act*yh_act; % do_act=yo_act=Act_a

%% Seleção semi-aleatória de ação (Uniformily)
%-----Ação->referência (Wm)-----
%     %Gaussiana
%     Act_r = random('unif',-0.5,0.5);
%     Act_r = sign(Act_r)*(sigma/0.2)*(1-exp(-(Act_r^2)));
%     Act_t = Act_a*(1+Act_r);
%     %Uniforme
%     Act_r = random('unif',1-sigma,1+sigma);
%     Act_t = abs(Act_r*Act_a);
%e-greed
if(epsilon>random('unif',0,1))
    Act_t=Act_a;
else
    Act_t=Act_a*(1+random('unif',-sigma,sigma));
end

if(Wm<=Wm_min)
    Act_t=Wm_min*1.05;
elseif(Wm>=Wm_max)
    Act_t=Wm_max*0.95;
end
Act_t=max(min(Wm_max,Act_t),Wm_min);
Wm_ref = Act_t;

%     %-----Ação->variação de referência (dWm)-----
%     Act_r = random('unif',-0.5,0.5);
%     Act_r = sign(Act_r)*(sigma/0.2)*(1-exp(-(Act_r^2)));
%     Act_t = Act_a+Act_r;
%
% %     Act_r = random('unif',-30*sigma,+30*sigma);
% %     Act_t = Act_r + Act_a;
%     if(Wm<=Wm_min)
%         Act_t=abs(Act_t);
%     elseif(Wm>=Wm_max)
%         Act_t=-abs(Act_t);
%     end
%     Wm_ref = Wm+Act_t;

%% Backpropagation error do Critic
for o=1:o_size_crt % Camada de saída
    for h=1:h_size_crt+1
        eo_crt(o,h) = yh_crt(h,1);
    end
end
for o=1:o_size_crt % Camada oculta

```

CACLA.m

```
    for h=1:h_size_crt
        for i=1:i_size_crt+1
            eh_crt(h,i) = wo_crt(o,h)*in_crt(i,1)...
                *yh_crt(h,1)*(1-yh_crt(h,1));
            % deriv. sigmoid: f'(x)*(1-f(x))
        end
    end
end
%% Backpropagation error do Actor
for o=1:o_size_act % Camada de saída
    for h=1:h_size_act+1
        eo_act(o,h) = yh_act(h,1);
    end
end
for o=1:o_size_act % Camada oculta
    for h=1:h_size_act
        for i=1:i_size_act+1
            eh_act(h,i) = wo_act(o,h)*in_act(i,1)...
                *yh_act(h,1)*(1-yh_act(h,1));
            % deriv. sigmoid: f'(x)*(1-f(x))
        end
    end
end

Wm_ref_val=Wm_ref;
Act_t_val=Act_t;
Act_a_val=Act_a;
r_val=r;
Vs_val=Vs;
wh_act_o=wh_act;
wo_act_o=wo_act;
wh_crt_o=wh_crt;
wo_crt_o=wo_crt;
```

APÊNDICE H: CÓDIGO DE PARÂMETROS DO PROJETO MATLAB

```
Boot.m
%% Código de inicialização de parâmetros para a simulação
close all;clear all;clc;
% PARÂMETROS DO SISTEMA WECS/WTG
% Universidade Federal do Ceará
% Autor: Jorge Luiz Wattes Oliveira Junior

%% Parametros do Retificador trifásico a diodo [Livro azul Ivo Barbi]
Rect.Vd = 0.0;           % Queda de tensão nos diodos
Rect.Rd = 0.0;           % Resistencia de condução dos diodos
Rect.Kret = 2.32;        % Vcc_med=2.32*Vrms_fase
% Filtro capacitivo
Rect.C = 470e-6;
Rect.rC = 0.7

%% Sistema de emulação de turbina eólica[WTE]
% Parametros do Motor CC
MCC.Va = 220;            % Tensão de armadura nominal do Motor CC
MCC.Ia=9;                % Corrente nominal Motor CC
MCC.If=0.65;
MCC.RPM=1800;
MCC.La = 6e-3;           % Indutância de armadura
MCC.Ra = 1.1;            % Resistência de armadura
MCC.tau_e = MCC.La/MCC.Ra;
MCC.Lf = 13.2;           % Indutância de campo
MCC.Rf = 337;            % Resistência de campo
MCC.B = 0.041603;        % Coeficiente de inéfrica
MCC.tau_m = 1;
MCC.J = MCC.tau_m*MCC.B; % Coeficiente de atrito viscoso
MCC.Kt = 1.0403;         % Coeficiente de relação Torque/Corrente
MCC.Kw = 1.0403

% Parametros do conversor Chopper [WTE]
Chopper.Vi=sqrt(2)*220; % Tensão de entrada do Chopper
Chopper.Vo=220;         % Tensão de saída do Chopper
Chopper.fs=20e3;        % Frequencia de chaveamento do Chopper
Chopper.Tsample = 1/Chopper.fs; % Período de amostragem/controlado do
Chopper
Chopper.Vtri = 1000;     % Pico da triangular
Chopper.Hi = 0.1;       % Ganho do sensor de corrente 2.5 ~ 4.5V
: 0 ~ 20A
Chopper.Kpwm = Chopper.Vi/Chopper.Vtri; % Ganho do modulador PWM
Chopper.Hadc = (2^16-1)/5; % Ganho do conversor AD
Chopper.base_corrente = Chopper.Hadc*Chopper.Hi;
Chopper.Ci = 2*470e-6

%% Parametros da Turbina Eólica
WECS.poles = 14;
WECS.B = 0.041603;      % Coeficiente de inéfrica
WECS.tau = 1;
WECS.J = WECS.tau*WECS.B; % Coeficiente de atrito viscoso
WECS.Ke = 0.4119;       % [Vrms_fase/(rad/s)] 61Vpk/kRPM ->
(0.061*60)/(2*pi*sqrt(2)) Vrms/(rad/s) [Dissertação Isaac]
WECS.Vpkkrpm = 1000*sqrt(3*2)*0.4119*2*pi/60; % Psim
WECS.Kt = 0.847;%2.32*WECS.Ke; % Corrente CC -> 2.32*Icc_med =
Irms_fase
WECS.Tmax = 4*WECS.Kt;
WECS.Ra = 0.5;
```

Boot.m

```
WECS.Ls = 0.00335;           % Indutância própria dos enrolamentos [H]
WECS.Lm = 0.00309;           % Indutância mutua dos enrolamentos [H]
WECS.La = (3/2)*WECS.Ls;
WECS.Lq = (3/2)*(WECS.Ls + WECS.Lm);
WECS.Ld = (3/2)*(WECS.Ls - WECS.Lm);
WECS.ro = 1.2928;
WECS.raio = 1;
WECS.Lambda_opt = 8;        % 8.1736
WECS.Cp_max = 0.35;         % 0.4916
WECS.Cp_max2 = 0.4916;
WECS.Klambda = 8.1736/8;
WECS.Kcp = 0.35/0.4916;
WECS.Pnom = 350;
WECS.V_nom = (WECS.Pnom/(0.5*WECS.ro*pi*WECS.Cp_max*WECS.raio^2))^(1/3);
WECS.V_min = 4;
WECS.V_max = 10;
WECS.V_mean = (WECS.V_max+WECS.V_min)/2;
WECS.Wm_nom=WECS.Lambda_opt*WECS.V_nom/WECS.raio;
WECS.Wm_min=floor(WECS.Lambda_opt*WECS.V_min/WECS.raio);
WECS.Wm_max=ceil(WECS.Lambda_opt*WECS.V_max/WECS.raio);
WECS.Wm_mean=(WECS.Wm_max+WECS.Wm_min)/2;
WECS.RPM_min=floor(60*WECS.Wm_min/(2*pi));
WECS.RPM_max=ceil(60*WECS.Wm_max/(2*pi));
WECS.FEM_min=Rect.Kret*WECS.Ke*WECS.Wm_min;
WECS.FEM_max=Rect.Kret*WECS.Ke*WECS.Wm_max;
WECS.EmuPrd=0.01;
WECS.beta = 0

%% Parametros do conversor Boost [WECS]
Boost.Vi = 40;
Boost.Vi_max=120;
Boost.Vi_min=40;
Boost.Vo = 150;
Boost.Po = 400;
Boost.IL = 2*Boost.Po/(Boost.Vi_max+Boost.Vi_min);
Boost.IL_max = 4;
Boost.fs = 20e3;
Boost.D = 0.74;
Boost.Dl = 1-Boost.D;
%Indutor
Boost.L = 3e-3;
Boost.rL = 0.01;
%Capacitor
Boost.C = 200e-6;
Boost.rC = 0.815;
%Carga
Boost.R=1.071;
%Chave - IIRFP3710
Boost.rS = 0.014;
Boost.Vs = 1.3;
Boost.Tfs=48e-9;
%Diodo
Boost.rD = 0.001;
Boost.Vd =1.75;
%Paramtros de controle
Boost.Vtri=998;
Boost.Kpwm=1/Boost.Vtri;
% Sensor de tensão
Boost.Hv = 4.7e3/(120.7e3+4.68e3);           % Ganho do sensor de tensão
Boost.Hv = 0.185/5;                         % Ganho real
Boost.Hv_filt_fh = 1e3;                     % 1kHz frequencia de projeto
Boost.Hv_filt_C = 1e-9;                     % 1nF
Boost.Hv_filt_R = 1/(2*pi*Boost.Hv_filt_C*Boost.Hv_filt_fh);
Boost.Hv_filt_R = 120.7e3;
```

Boot.m

```
Boost.Hv_filt_fh = 1/(2*pi*Boost.Hv_filt_C*Boost.Hv_filt_R);% Freqüencia
obtida
Boost.Hv_filt = tf(1,[(Boost.Hv_filt_C*Boost.Hv_filt_R),1]);
% Ganho do ADC
Boost.Hadc = ((2^15)-1)/5;
% Sensor de corrente
Boost.Hi = (46.6/(5.8+5.7))*2/20;          % Ganho do sensor de corrente
%Boost.Hi = 0.41;
Boost.Hi_filt_fh = Boost.fs;      % 20kHz frequencia de projeto
Boost.Hi_filt_C = 1.5e-9;        % 10nF
Boost.Hi_filt_R = 1/(2*pi*Boost.Hi_filt_C*Boost.Hi_filt_fh);
Boost.Hi_filt_R = 5.07e+03;
Boost.Hi_filt_fh = 1/(2*pi*Boost.Hi_filt_C*Boost.Hi_filt_R);% Freqüencia
obtida
Boost.Hi_filt = tf(1,[(Boost.Hi_filt_C*Boost.Hi_filt_R),1]);
Boost.base_corrente = Boost.Hi*Boost.Hadc;
%Sensor de velocidade
Boost.Hw = (2^5)*(60/(2*pi));
Boost.Hw_Tsample = 0.01; % Ganho do sensor de rotaçäo (leitura em RPM)
Boost.Hw_freq = 1/Boost.Hw_Tsample;
Boost.Hw_tf=tf(1,[(1/(2*pi*Boost.Hw_freq/4)),1]);
Boost.Hw_tfd=c2d(Boost.Hw_tf,Boost.Hw_Tsample,'tustin');
[Boost.Hw_num,Boost.Hw_den]=tfdata(Boost.Hw_tfd);
Boost.Hw_num{1,1}=Boost.Hw_num{1,1}*(9.6)*(2^10-1);
Boost.Hw_den{1,1}=Boost.Hw_den{1,1}*(2^10-1)

%% Controle WTG
% Controle de Torque(corrent) do PMSG
WECSCid.Tsample = 1/(2*Boost.fs);
WECSCid.TFg = Boost.Kpwm*tf(Boost.Vo,[Boost.L,0]);
WECSCid.TFgd = c2d(WECSCid.TFg,WECSCid.Tsample,'tustin');
WECSCid.TFh = Boost.Hi_filt; %Boost.Hi_filt*Boost.Hi*Boost.Hadc;
WECSCid.TFhd = c2d(WECSCid.TFh,WECSCid.Tsample,'tustin');
WECSCid.TFc = 0;
WECSCid.TFcd = 0;
WECSCid.TFmasc = WECSCid.TFg*WECSCid.TFh;
WECSCid.TFmascd = WECSCid.TFgd*WECSCid.TFhd;
WECSCid.TFmacc = 0;
WECSCid.TFmaccd = 0;
WECSCid.TFmf = 0;
WECSCid.TFmfd = 0;
figure('position',[100 100 1000 400])
h = bodeplot(WECSCid.TFmasc,'black',WECSCid.TFmascd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 1e4]);
legend('Contínuo','Discreto');
grid on;
title('[WTG - Corrente] Malha aberta sem compensador')
WECSCid.Wci = (2*pi*Boost.fs)/10;          % Freqüencia de cruzamento
WECSCid.Mpi = 70;                          % Margem de fase
opts = pidtuneOptions('PhaseMargin',WECSCid.Mpi);
[Ci,infoi] = pidtune(WECSCid.TFmasc,'pi',WECSCid.Wci,opts);
WECSCid.Wci = infoi.CrossoverFrequency/(2*pi);
WECSCid.Mpi = infoi.PhaseMargin;
WECSCid.TFc = tf(Ci);
WECSCid.TFcd = c2d(WECSCid.TFc,WECSCid.Tsample,'tustin');
WECSCid.TFmacc = WECSCid.TFmasc*WECSCid.TFc;
WECSCid.TFmaccd = WECSCid.TFmascd*WECSCid.TFcd;
figure('position',[100 100 1000 400])
h = bodeplot(WECSCid.TFmacc,'black',WECSCid.TFmaccd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 1e4]);
legend('Contínuo','Discreto');
grid on;
title('[WTG - Corrente] Malha aberta com compensador')
WECSCid.TFmf = feedback(WECSCid.TFmacc,WECSCid.TFh);
```

Boot.m

```
WECSCid.TFmfd = feedback(WECSCid.TFmaccd,WECSCid.TFhd);
[WECSId.num,WECSId.den]=tfdata(WECSCid.TFcd);
WECSCid.num{1,1} = WECSCid.num{1,1}/(Boost.Hi*Boost.Hadc);
WECSCid.c1 = ((2^15)-1)*WECSCid.num{1,1}(1,1)/WECSId.den{1,1}(1,1);
WECSCid.c2 = ((2^15)-1)*WECSCid.num{1,1}(1,2)/WECSId.den{1,1}(1,1);
WECSCid.c3 = -((2^15)-1)*WECSId.den{1,1}(1,2)/WECSId.den{1,1}(1,1)
figure('position',[100 100 1000 400])
h = bodeplot(WECSCid.TFmf,'black',WECSCid.TFmfd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 1e4]);
legend('Contínuo','Discreto');
grid on;
title('[WTG - Corrente] Malha fechada')
figure('position',[100 100 1000 400])
step(WECSCid.TFmf,'black',WECSCid.TFmfd,'black--');
legend('Contínuo','Discreto');
grid on;
title('[WTG - Corrente] Degrau de referência')
clear Ci Cid infoi infoid

%% Controle de Velocidade do PMSG
WECSCwi.Tsample = Boost.Hw_Tsample; % Boost.Tsample;
WECSCwi.TFg = -(32/(Boost.Hi*Boost.Hadc))*tf(WECS.Kt,[WECS.J,WECS.B]);
%Boost.Hadc
WECSCwi.TFgd = c2d(WECSCwi.TFg,WECSCwi.Tsample,'tustin');
WECSCwi.TFh = 1; % (Boost.Hw);
WECSCwi.TFc = 0;
WECSCwi.TFcd = 0;
WECSCwi.TFmasc = WECSCwi.TFg*WECSCwi.TFh;
WECSCwi.TFmascd = WECSCwi.TFgd*WECSCwi.TFh;
WECSCwi.TFmacc = 0;
WECSCwi.TFmaccd = 0;
WECSCwi.TFmf = 0;
WECSCwi.TFmfd = 0;
figure('position',[100 100 1000 400])
h = bodeplot(WECSCwi.TFmasc,'black',WECSCwi.TFmascd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 0.5/Boost.Hw_Tsample]);
legend('Contínuo','Discreto');
grid on;
title('[WTG - Rotação] Malha aberta sem compensador')
WECSCwi.Wcw = (2*pi*Boost.Hw_freq)/20; % Frequencia de cruzamento
da malha de velocidade(Omega)
WECSCwi.Mpw = 70; % Margem de fase da malha
de velocidade(Omega)
opts = pidtuneOptions('PhaseMargin',WECSCwi.Mpw);
[Cw,infow] = pidtune(WECSCwi.TFmasc,'pi',WECSCwi.Wcw,opts);
WECSCwi.Wcw = infow.CrossoverFrequency/(2*pi);
WECSCwi.Mpw = infow.PhaseMargin;
WECSCwi.TFc = tf(Cw);
WECSCwi.TFcd = c2d(WECSCwi.TFc,WECSCwi.Tsample,'tustin');
WECSCwi.TFmacc = WECSCwi.TFmasc*WECSCwi.TFc;
WECSCwi.TFmaccd = WECSCwi.TFmascd*WECSCwi.TFcd;
figure('position',[100 100 1000 400])
h = bodeplot(WECSCwi.TFmacc,'black',WECSCwi.TFmaccd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 0.5/Boost.Hw_Tsample]);
legend('Contínuo','Discreto');
grid on;
title('[WTG - Rotação] Malha aberta com compensador')
WECSCwi.TFmf = feedback(WECSCwi.TFmacc,WECSCwi.TFh);
WECSCwi.TFmfd = feedback(WECSCwi.TFmaccd,WECSCwi.TFh);
[WECSwi.num,WECSwi.den]=tfdata(WECSCwi.TFcd);
WECSCwi.num{1,1}=WECSwi.num{1,1}/Boost.Hw;
%Com anti wind-up
Cw = c2d(Cw,WECSCwi.Tsample,'tustin');
WECSCwi.c1a = (((2^15)-1)*Cw.Kp)/Boost.Hw;
```

Boot.m

```
WECSCwi.c2a = ((2^15)-1)*(WECSCwi.Tsample)/(Cw.Kp/Cw.Ki);
WECSCwi.c1=((2^15)-1)*WECSCwi.num{1,1}(1,1)/WECSCwi.den{1,1}(1,1);
WECSCwi.c2=((2^15)-1)*WECSCwi.num{1,1}(1,2)/WECSCwi.den{1,1}(1,1);
WECSCwi.c3=-((2^15)-1)*WECSCwi.den{1,1}(1,2)/WECSCwi.den{1,1}(1,1)
figure('position',[100 100 1000 400])
h = bodeplot(WECSCwi.TFmf,'black',WECSCwi.TFmfd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 0.5/Boost.Hw_Tsample]);
legend('Contínuo','Discreto');
grid on;
title('[WTG - Rotação] Malha fechada')
figure('position',[100 100 1000 400])
step(WECSCwi.TFmf,'black',WECSCwi.TFmfd,'black--');
legend('Contínuo','Discreto');
grid on;
title('[WTG - Rotação] Degrau de referência')
%clear Cw infow Cwd infowd

% Controle do conversor Chopper [MCCId]
% Malha de Torque/corrente
MCCId.Tsample = 1/(Chopper.fs);
MCCId.TFg = Chopper.Kpwm*tf(1,[MCC.La, MCC.Ra]);
MCCId.TFgd = c2d(MCCId.TFg,MCCId.Tsample,'tustin');
MCCId.TFh = 1; %Chopper.Hi*Chopper.Hadc;
MCCId.TFc = 0;
MCCId.TFcd = 0;
MCCId.TFmasc = MCCId.TFg*MCCId.TFh;
MCCId.TFmascd = MCCId.TFgd*MCCId.TFh;
MCCId.TFmacc = 0;
MCCId.TFmaccd = 0;
MCCId.TFmf = 0;
MCCId.TFmfd = 0;
figure('position',[100 100 1000 400])
bode(MCCId.TFmasc);
h = bodeplot(MCCId.TFmasc,'black',MCCId.TFmascd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 1e3]);
legend('Contínuo','Discreto');
grid on;
title('[WTE - Corrente] Malha aberta sem compensador')
MCCId.Wci = (2*pi*Chopper.fs)/10; % crossover frequency
MCCId.Mpi = 80; % margem de fase
%% Utilizando pidtunes
opts = pidtuneOptions('PhaseMargin',MCCId.Mpi);
[Ci,infoi] = pidtune(MCCId.TFmasc,'pi',MCCId.Wci,opts);
%[Cid,infoid] = pidtune(MCCId.TFmascd,'pi',MCCId.Wci,opts);
Cid=c2d(Ci,MCCId.Tsample,'tustin');
MCCId.Wci = infoi.CrossoverFrequency/(2*pi);
MCCId.Mpi = infoi.PhaseMargin;
%% Segundo Mohan
%MCCId.Ki = (MCCId.Wci)*MCC.Ra)/(Chopper.Kpwm*MCCId.TFh);
%MCCId.Kp = MCC.tau_e*MCCId.Ki;
%Ci = pid(MCCId.Kp,MCCId.Ki)
%Cid = c2d(Ci,MCCId.Tsample,'tustin')
MCCId.TFc = tf(Ci);
MCCId.TFcd = tf(Cid);
MCCId.TFmacc = MCCId.TFmasc*MCCId.TFc;
MCCId.TFmaccd = MCCId.TFmascd*MCCId.TFcd;
figure('position',[100 100 1000 400])
h = bodeplot(MCCId.TFmacc,'black',MCCId.TFmaccd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 1e5]);
legend('Contínuo','Discreto');
grid on;
title('[WTE - Corrente] Malha aberta com compensador')
MCCId.TFmf = feedback((MCCId.TFg*MCCId.TFc),MCCId.TFh);
MCCId.TFmfd = feedback((MCCId.TFgd*MCCId.TFcd),MCCId.TFh);
```

Boot.m

```
[MCCId.num,MCCId.den]=tfdata(tf(Cid));
MCCId.num{1,1}=MCCId.num{1,1}/(Chopper.Hi*Chopper.Hadc);
MCCId.c1 = ((2^15-1)*MCCId.num{1,1}(1,1)/MCCId.den{1,1}(1,1));
MCCId.c2 = ((2^15-1)*MCCId.num{1,1}(1,2)/MCCId.den{1,1}(1,1));
MCCId.c3 = -(2^15-1)*MCCId.den{1,1}(1,2)/MCCId.den{1,1}(1,1)
figure('position',[100 100 1000 400])
h = bodeplot(MCCId.TFmf,'black',MCCId.TFmfd,'black--');
setoptions(h,'FreqUnits','Hz','Xlim',[1 1e4]);
legend('Contínuo','Discreto');
grid on;
title('[WTE - Corrente] Malha fechada')
figure('position',[100 100 1000 400])
step(MCCId.TFmf,'black',MCCId.TFmfd,'black--');
legend('Contínuo','Discreto');
grid on;
title('[WTE - Corrente] Degrau de referência')
clear Ci infoi opts

%% Algoritmo emulador
figure('position',[100 100 1000 400])
x=0.1:((13.65-0.1)/256):13.6;
cp=0.5*(116*(1./x - 0.035/1)-5).*exp(-21*(1./x - 0.035/1)) + 0.01*x;
EMU.cp_lut =
round((Chopper.Hi*Chopper.Hadc)*(60/(2*pi))*(1/MCC.Kt)*(WECS.ro*pi*(WECS.
raio^2)/2)*cp);
EMU.cp_lut(length(EMU.cp_lut)) = 0;
plot(x,EMU.cp_lut,'black')
figure('position',[100 100 1000 400])
x = 0:0.01:2000;
EMU.wind = (sin(2*pi*x/25)+sin(2*pi*x/125) + sin(2*pi*x/250) +
sin(2*pi*x/500))/4 + 6;
x = 0:2000;
EMU.wind_lut = (sin(2*pi*x/25)+sin(2*pi*x/125) + sin(2*pi*x/250) +
sin(2*pi*x/500))/4 + 6;
plot(x,EMU.wind_lut,'black')
EMU.wind_lut = round(EMU.wind_lut*(2^7));
clear x cp

%% Parametros do MPPT
MPPT.Wm_max=1000;
MPPT.Wm_min=400;
MPPT.period=1;
MPPT.step=8;
MPPT.Kw=1/(Rect.Kret*WECS.Ke)

%% Parametros de simulação
SIM.WECStime = 20;
SIM.WECSSample = 1e-3;
SIM.WECSplot = 1e-5;
SIM.WTEtime = 0.5;
SIM.WTESample = 1e-8;
SIM.Boosttime = 0.005; %Current loop=0.04 / Speed Loop=5.0s
SIM.Boostsample = 1e-6
```