UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE BACHARELADO EM ENGENHARIA ELÉTRICA

Eduardo Puhl

DESENVOLVIMENTO DE APLICAÇÃO PARA CONTROLE DA EMULAÇÃO EM TEMPO REAL DE TURBINAS EÓLICAS EM BANCADA

Florianópolis
2020

Eduardo Puhl

DESENVOLVIMENTO DE APLICAÇÃO PARA CONTROLE DA EMULAÇÃO EM TEMPO REAL DE TURBINAS EÓLICAS EM BANCADA

Florianópolis

2020

Eduardo Puhl

DESENVOLVIMENTO DE APLICAÇÃO PARA CONTROLE DA EMULAÇÃO EM TEMPO REAL DE TURBINAS EÓLICAS EM BANCADA

Trabalho de Conclusão do Curso de Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia Elétrica.
Orientador: Prof. Marcelo Lobo Heldwein, Dr. Sc. ETH

Florianópolis
2020

Puhl, Eduardo
DESENVOLVIMENTO DE APLICAÇÃO PARA CONTROLE DA EMULAÇÃO EM TEMPO REAL DE TURBINAS EÓLICAS EM BANCADA / Eduardo Puhl; orientador, Marcelo Lobo Heldwein, 2020.
111 p.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2020.

Inclui referências.

1. Engenharia Elétrica. 2. Turbinas Eólicas. 3. Power-hardware-in-the-loop. 4. Eletrônica de Potência. I. Heldwein, Marcelo Lobo. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. III. Título.

Eduardo Puhl

DESENVOLVIMENTO DE APLICAÇÃO PARA CONTROLE DA EMULAÇÃO EM TEMPO REAL DE TURBINAS EÓLICAS EM BANCADA

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Engenharia Elétrica" e aprovado em sua forma final pela Banca Examinadora.

Florianópolis, 26 de outubro de 2020.

Documento assinado digitalmente
Jean Vianei Leite
Data: 11/11/2020 11:15:53-0300
CPF: 003.474.909-80

Prof. Jean Vianei Leite, Dr. Sc.
Coordenador do Curso de Graduação em Engenharia Elétrica

Banca Examinadora:

Documento assinado digitalmente
Marcelo Lobo Heldwein
Data: 10/11/2020 21:26:04-0300
CPF: 018.327.389-30

Prof. Marcelo Lobo Heldwein, Dr. Sc. ETH
Orientador(a)
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Samir Ahmad Mussa
Data: 10/11/2020 20:22:15-0300
CPF: 401.044.430-49

Prof. Samir Ahmad Mussa, Dr. Sc.
Avaliador(a)
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Leandro Benhur Klinger Fisch
Data: 10/11/2020 20:14:32-0300
CPF: 032.502.580-08

Eng. Leandro Fisch, B.Sc.
Avaliador(a)
Universidade Federal de Santa Catarina

Dedico este trabalho aos familiares, professores, amigos e colegas, sem os quais certamente não teria chegado até aqui.

AGRADECIMENTOS

Agradeço primeiramente a Deus por sempre me guiar pelo meu caminho. Agradeço aos meus familiares, que nunca mediram esforços para que eu pudesse dar prosseguimento e me dedicar totalmente à minha formação acadêmica. Sem vocês, nada disso seria possível.

Agradeço a todos os professores que em algum momento passaram pela minha vida acadêmica e auxiliaram a construir o conhecimento que adquiri durante todos esses anos, especialmente agradeço ao professor Marcelo Lobo Heldwein, que não apenas me orientou neste presente trabalho como também dedicou parte de seu tempo destes últimos dois anos e meio à me orientar no desenvolvimento de iniciação científica no laboratório de microrredes do Instituto de Eletrônica de Potência. Agradeço a sua paciência, solicitude e por compartilhar parte do conhecimento ímpar que possui.

Agradeço aos meus amigos que me acompanharam nesta jornada, sempre agregando com o auxílio mútuo nas atividades acadêmicas e compartilhando experiências que levarei para toda vida.

“*Uma vida sem reflexão não vale a pena ser vivida*”
(Sócrates, c. IV a.C)

RESUMO

A energia eólica é indubitavelmente uma realidade na matriz energética brasileira. Sendo a terceira maior fonte de energia elétrica do país em capacidade instalada, cada vez vem ganhando mais espaço com a inerente preocupação ambiental e a consequente urgência em descarbonizar cada vez mais nossa matriz energética. Todos os dias diversos estudos são desenvolvidos com o objetivo de implementar novas tecnologias que permitam o aumento da eficiência e integração à matriz energética dos aerogeradores, seja para a aplicação na geração centralizada em parques eólicos ou em sistemas de geração distribuída. O desenvolvimento e os testes de novos equipamentos utilizados no processo de conversão da energia eólica se beneficiam do uso de sistemas capazes de recriar as condições da aplicação, uma vez que seria inviável utilizar o vento em um local para isto, o qual teria certamente características peculiares e possivelmente não geraria de maneira constante as variações e os limites de operação, condições necessárias para testar novos equipamentos. Para este fim são utilizados sistemas de emulação em tempo real, também denominados *Power-Hardware-in-the-Loop*. Tais sistemas são controlados por microcontroladores ou computadores que necessitam serem programados para tal função. Este trabalho visa implementar uma aplicação que permite realizar a emulação em tempo real de turbinas eólicas utilizando um sistema físico de inversor, motor e gerador, com o objetivo de integrá-la às fontes energéticas despacháveis de uma microrrede e possibilitar futuros estudos de análise de turbinas eólicas.

Palavras-Chave: 1. Aerogeradores. 2. Emulação. 3. Energia eólica. 4. *Real Time.*

ABSTRACT

Wind energy is undoubtedly a reality in the Brazilian energy matrix. As the third largest source of electric energy in the country in installed capacity, it is increasingly gaining ground with environmental concerns and the consequent urgency to increasingly decarbonize our energy matrix. Every day, several studies are developed with the objective of implementing new technologies that increase the efficiency and integration to the energy matrix of the wind turbines, whether for an application in centralized generation in wind farms or in distributed generation systems. The development and testing of new equipment used in wind energy conversion systems benefits from the use of systems capable of recreating the conditions of the application. It would be unfeasible to use the wind in a certain location for this, which would certainly present peculiar characteristics and possibly would not often generate typical variations and operating limits. These are necessary conditions to test new equipment. For this purpose, real-time emulation systems, also called *Power-Hardware-in-the-Loop*, are used. Such systems are controlled by microcontrollers or computers that need to be programmed for this function. This work aims to implement an application that allows real-time emulation of wind turbines using a physical inverter, motor and generator system, with the objective of integrating it with the dispatchable energy sources of a microgrid and enabling future wind turbine analysis.

Keywords: 1. Wind turbines. 2. Emulation. 3. Wind energy. 4. Real time.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

A/D	Conversor Analógico/Digital
ADU	*Application Data Unit*
ANEEL	Agência Nacional de Energia Elétrica
API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
CA	Corrente Alternada
CAN	*Controller Area Network*
CC	Corrente Contínua
CERTI	Centro de Referência em Tecnologias Inovadoras
CLP	Controlador Lógico Programável
CPU	*Central Process Unit*
D/A	Conversor Digital Analógico
DFIG	*Doubly Fed Induction Generator*
DOS	*Disk Operating System*
GUI	*Graphical User Interface*
I/O	*In-Out*
INEP	Instituto de Eletrônica de Potência
IP	*Internet Protocol*
MPPT	*Maximum Power Point Tracker*
OSI	*Open System Interconection*
PC	*Personal Computer*
PDU	*Protocol Data Unit*
RPM	Rotações por Minuto
TCP/IP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*

LISTA DE SÍMBOLOS

ω'_M	Velocidade mecânica lida pelo transdutor de torque [rad/s]
ω_{omega}	Velocidade angular da função senoidal do vento [rad/s]
C_P	Coeficiente de potência de um aerogerador
C_n	Coeficientes constantes da turbina, onde n ∈ {1, 2, 3, ..., 6}
E_{Ci}	Energia cinética inicial da massa de ar em movimento [J]
E_{Co}	Energia cinética restante após passagem pelo gradiente [J]
$J_{aerogerador}$	Constante de inércia do aerogerador [$kg.m^2$]
$M_{pá}$	Massa da pá [kg]
P_{MEC}	Potência mecânica da máquina [W]
$R_{pá}$	Raio da pá [m]
T_M	Torque mecânico de referência da turbina eólica [N.n]
T'_M	Torque mecânico lido pelo transdutor de torque [N.m]
$V_{\omega,avg}$	Velocidade média do vento [m/s]
v_o	Velocidade do vento após as pás [m/s]
$v_\omega(t)$	Velocidade de vento [m/s]
β	Posição angular das pás *pitch* [º]
ΔE_C	Variação da energia que realiza trabalho [J]
Δt	Variação de tempo [s]
λ	Velocidade de ponta da turbina (*Tip Speed Ratio)*
ρ	Densidade do ar [$kg.m^3$]
Ω	Resistência elétrica [Ohm]
ω	Velocidade angular da turbina [rad/s]
A	Área varrida pelas pás [m^2]
K	Constante de tensão da máquina síncrona [V/RPM]
Tm	Torque mecânico [N.m]
d	Distância percorrida pela massa de ar [m]
m	Massa do ar em movimento [kg]
r	Raio da turbina [m]

SUMÁRIO

1 INTRODUÇÃO

Com os avanços tecnológicos e industriais de nossa sociedade e a ampliação do acesso aos novos padrões de consumo estabelecidos, a demanda energética consequentemente evidencia a necessidade de uma expansão da geração de energia elétrica para continuar suprindo as necessidades energéticas da produção.

Como aponta Manwell *et al*. (2009, pg. 1), durante todo o século XIX e boa parte do século XX as principais fontes energéticas foram as derivadas dos combustíveis fósseis. Entretanto, no final da década de 60 com o desenvolvimento de novas tecnologias a energia com origem motriz dos ventos passou a ter uma nova exposição de possibilidades ao mundo, firmando-se a partir da década de 90, quando a conscientização sobre a finitude dos combustíveis de origem fóssil e os malefícios dos gases por eles gerados teve atenção global e a energia eólica passou a receber incentivos fiscais governamentais que a tornou financeiramente competitiva frente à outras fontes energéticas.

Desde a década de 90 então, com a profusão do desenvolvimento de tecnologias para aplicação na geração eólica, sobretudo na eletrônica de potência, os aerogeradores passaram da capacidade de alguns poucos quilowatts até chegarem à faixa da dezena de megawatts (MANWELL, 2009). Para isso, testes de novos modelos de geradores (síncrono, *DFIG,* síncrono à ímãs permanentes etc.), conversores de potência, perfis de pás e outras tecnologias foram desenvolvidas no intuito de melhorar a capacidade e performance dos aerogeradores modernos.

Com o viés de contribuir com o desenvolvimento de tecnologias que permitam auxiliar em uma geração de energia limpa, inclusive no panorama de incentivo às gerações elétricas distribuídas, em parceria com a Fundação CERTI e a Engie Brasil Energia, o Instituto de Eletrônica de Potência da Universidade Federal de Santa Catarina implementou um projeto piloto de uma microrrede inteligente conectada à rede de distribuição utilizando como fontes geradoras arranjos fotovoltaicos, sistema eólico via emulador em tempo real, turbina a gás e banco de baterias.

O sistema eólico, porém, devido às limitações locais de geração eólica por conta da inconstância de vento e sua baixa velocidade, foi implementado através de uma bancada de emulação eólica composta por: gerador, motor, inversor, torquímetro, módulo de leitura de dados elétricos e *software* de controle.

O presente trabalho trata-se da implementação de uma aplicação para controle da bancada de *power-hardware-in-the-loop* (P-HIL) da emulação desta bancada de conversão eólica, com o objetivo de aproximar o máximo possível as respostas e geração da bancada de emulação ao performático de uma turbina eólica real, observadas as limitações das ferramentas e métodos aqui empregados, para que tal sistema seja integrado a geração da microrrede implementada.

1.1 Objetivo geral

Realizar o estudo do funcionamento físico de turbinas eólicas e da utilização de sistemas *power-hardware-in-the-loop* e realizar a implementação de uma aplicação para controle de um sistema de emulação de conversão eólica de energia utilizando a ferramenta *Simulink Real Time* e a linguagem de programação C#.

1.2 Objetivos específicos

- Realizar a implementação de um *software* para o controle de um sistema de emulação *hardware-in-the-loop* que simula fisicamente uma turbina eólica de 10 kW.
- Analisar a resposta do sistema implementado e seu desempenho.

2 ENERGIA EÓLICA E AEROGERADORES

Dado o já não tão recente emprego da energia eólica como fonte energética pela humanidade, nesta seção objetiva-se retratar parte da história do desenvolvimento dos aerogeradores modernos e a sua modelagem matemática estática utilizada neste trabalho, bem como a especificação dos parâmetros da turbina simulada.

2.1 Energia eólica na história

A utilização da energia eólica como força motriz remonta aos de moinhos de vento utilizados há mais de 3000 mil anos. Como relembra Burton *et al.* (2001, pg. 1), a energia eólica foi utilizada desde os primórdios auxiliando os povos antigos como força motriz na moagem de grãos, bombeamento de água e nas navegações oceânicas. Entretanto, a utilização dos ventos como força eletromotriz data-se apenas a partir do início do século XIX, com a construção do primeiro gerador eólico 12 kW. Apesar disso, durante a maior parte do século XX a geração de energia elétrica através de fonte eólica foi renegada à insignificância em detrimento do uso de combustíveis fósseis para geração elétrica. Segundo Manwell (2009, pg. 1), os primeiros sinais de mudança no paradigma de geração de energia elétrica surgiram no final dos anos 60 e foram confirmados a partir dos anos 90, quando o mundo notou um forte ressurgimento da indústria de energia eólica com o desenvolvimento e produção em larga escala de turbinas eólicas na faixa dos megawatts, redução de custos na produção das turbinas eólicas e o desenvolvimento de tecnologias para produção de energia *offshore.*

Ainda segundo Manwell (2009), existiram cinco fatores para o ressurgimento da energia eólica como uma fonte energética competitiva. Primeiramente a conscientização comum sobre a finitude dos combustíveis fósseis e os consequentes efeitos adversos advindos do seu uso. Em segundo lugar, o potencial eólico presente em quase todo o planeta, inclusive com lugares com uma considerável densidade de energia disponível. Em terceiro lugar, a capacidade tecnológica que permitiu que os desenvolvimentos em outras áreas pudessem revolucionar a utilização das turbinas eólicas como fontes eletro energéticas. Em quarto lugar, uma nova visão de meios de utilização do vento como fonte energética, e por fim, em quinto lugar, os apoios e

incentivos financeiros governamentais para que a energia eólica pudesse se tornar economicamente competitiva frente às matrizes energéticas de origem fóssil.

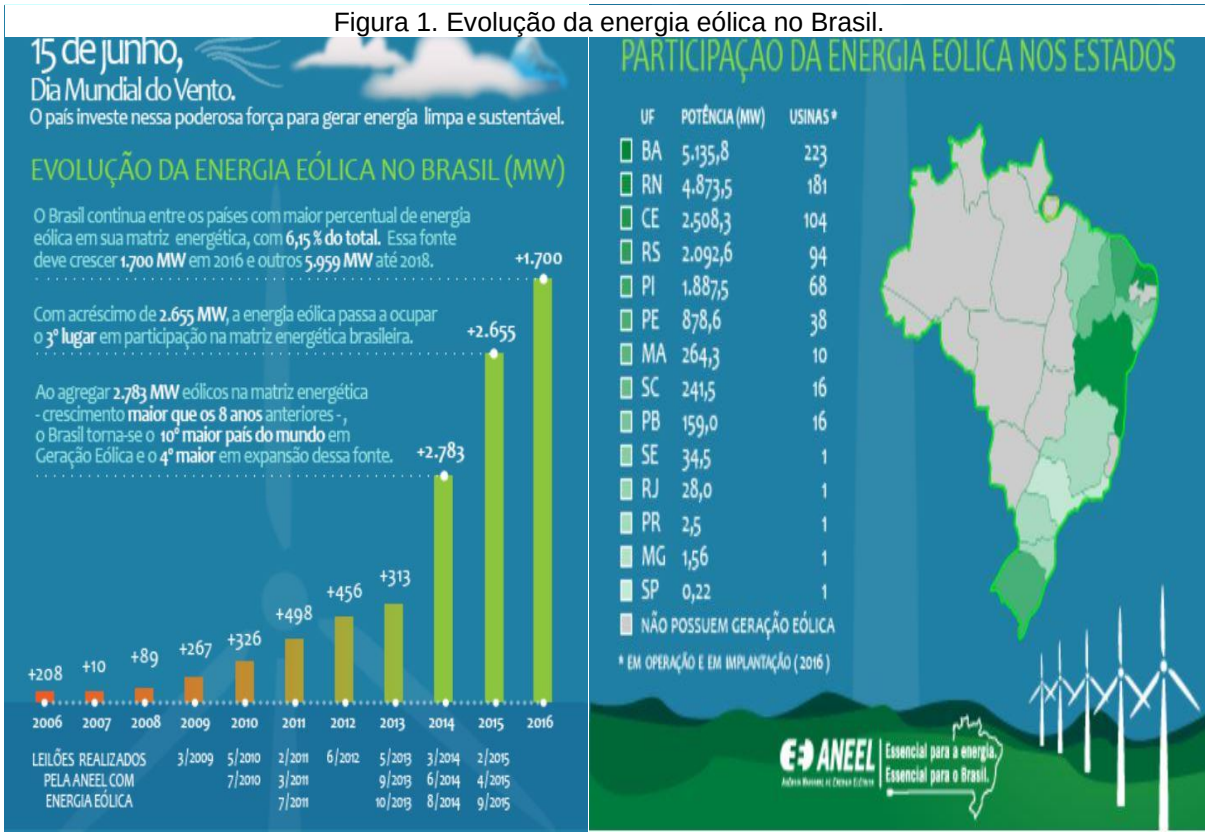
2.2 Panorama atual da energia eólica

Tratando-se de sistemas de energia, as matrizes energéticas de fontes renováveis vêm ganhando destaque nas últimas décadas com a crescente preocupação para a questão da sustentabilidade e a consequente busca por meios alternativos de geração de energia elétrica limpa.

Uma das principais fontes energéticas em ascensão é a energia eólica. Segundo o Boletim de Energia Eólica Brasil e Mundo - 2016 produzido pelo Ministério de Minas e Energia, até 2026 as fontes eólicas representarão 12,5% da capacidade energética instalada, incluindo geração distribuída. Com a crescente demanda pela implementação de novos sistemas eólicos, torna-se inerente a necessidade de novas tecnologias que permitam o aprimoramento destes sistemas através de testes, emulações e simulações.

Acompanhando essa tendência de expansão de energias renováveis, uma das possíveis soluções para suprir a demanda energética que vem sendo pesquisada e implementada é a geração distribuída de energia elétrica. A abordagem neste conceito de geração é a produção de energia próxima ao seu local de consumo através de, majoritariamente, fontes renováveis. As vantagens da geração distribuída são inúmeras, como: economia ao consumidor, adiamento de investimentos em redes de transmissão e distribuição, redução no carregamento das redes, minimização de perdas, diversificação da matriz energética e autossustentabilidade.

Figura 1. Evolução da energia eólica no Brasil.



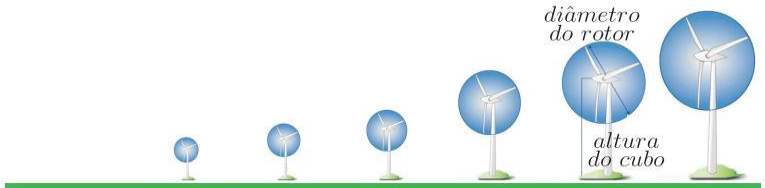
Fonte: ANEEL, 2017.

2.3 Descrição dos aerogeradores

Os aerogeradores são, em suma, equipamentos elétricos que convertem a energia cinética do vento em energia elétrica. Esses equipamentos são então conectados a redes elétricas compostas por circuitos industriais, residenciais, públicos, ilhados ou não para auxiliar no suprimento das demandas energéticas elétricas.

Com o passar das décadas, os aerogeradores foram aumentando gradativamente a sua potência nominal e consequentemente o seu tamanho. Atualmente, existem aerogeradores com potência nominal na faixa de alguns kW com aplicação geralmente em sistemas de baixa potência como geração distribuída, a até aerogeradores com potência nominal na faixa de vários MW, empregados na geração centralizada de energia elétrica em parques eólicos compostos por dezenas de aerogeradores.

Figura 2. Evolução da potência dos aerogeradores de 1980 à 2005.



	1980	1985	1990	1995	2000	2005
potência nominal	30 kW	80 kW	250 kW	600 kW	1,5 MW	5 MW
diâmetro do rotor	15 m	20 m	30 m	46 m	70 m	115 m
altura do cubo	30 m	40 m	50 m	78 m	100 m	120 m
produção de energia	35 GWh	95 GWh	400 GWh	1,25 TWh	3,5 TWh	17 TWh

Fonte: apud COLLIER, 2011.

Apesar dos mais variados *designs* e aspectos construtivos, como os aerogeradores de eixo vertical (Figura 3) utilizados geralmente em aplicações urbanas de baixa potência onde há um perfil de vento turbulento e de direção variável, aerogeradores de eixo horizontal de uma e duas pás (Figura 3), aerogeradores *downwind* (Figura 3) que não necessitam de controle ativo de *yaw* para posicionar o aerogerador em relação à direção do vento predominante, os aerogeradores mais

comuns encontrados no mercado e principalmente em parques eólicos de geração centralizada são os aerogeradores *upwind* com três pás. Estes aerogeradores possuem controle ativo de *yaw* que os permitem, através de um sensor *wind vane*, aferir a direção de vento predominante e posicionar o aerogerador para obtenção da máxima potência de vento disponível.

Figura 3. Aerogeradores de eixo vertical, uma, duas e três pás.

a)

b)

c)

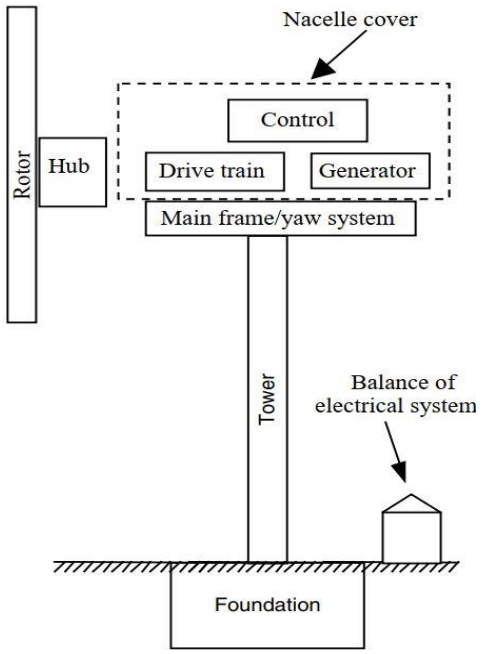
d)

Fontes: a) Aerogerador eixo vertical [https://www.eolicafacil.com.br/eixo-vertical]; b) Aerogerador uma pá [http://www.wind-works.org/]; c) Aerogerador duas pás [https://www.123rf.com/photo_23986078_twin-blade-wind-turbine-close-up.html]; d) Aerogerador três pás [Telles, 2018]. Acessos em 03-10-2020.

Segundo MANWELL (2009, pg. 3), as principais partes e características que compõem (resumidamente) a estrutura um aerogerador são:

- Rotor, que consiste nas pás do aerogerador e no cubo de suporte, além dos sistemas de controle de ângulos da pás;
- *Drive Train,* que inclui as partes girantes do aerogerador como eixo, gearbox, acoplamento, freio mecânico e gerador;
- Nacele e *main frame,* que inclui a carcaça e placa base da turbina eólica e o sistema de *yaw*, responsável pelo ajuste de posicionamento da nacele;
- Torre e fundação;
- Controles da máquina;
- Casa de máquinas, formada por cabos, transformadores e opcionalmente, conversores de eletrônica de potência.

Figura 4. Partes componentes de um aerogerador.



Fonte: MANWELL, 2009.

2.4 Modelagem matemática dos aerogeradores

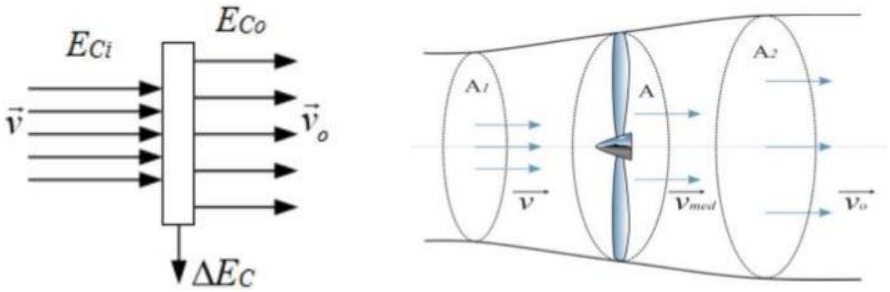
O equacionamento da energia extraída por uma turbina eólica nesta seção baseia-se no trabalho de TIBOLA (2009). A energia cinética contida em uma massa de vento é dada por:

$$E_{Ci} = E_{Co} + \Delta E_C \qquad (2.1)$$

Onde

E_{Ci} = Energia cinética inicial da massa de ar em movimento;
E_{Co} = Energia cinética restante após passagem pelo gradiente;
ΔE_C= Parte da energia que realiza trabalho;

Figura 5. Fluxo de ar sobre a área do rotor de um aerogerador.



Fonte: TELLES, 2018.

A energia ΔE_C é a energia eólica convertida em mecânica nas pás. A variação de energia cinética pode ser descrita por:

$$\Delta E_C = E_{Co} - E_{Ci} = \frac{m(v^2 - v_o^2)}{2} \quad (2.2)$$

m = Massa do ar em movimento;
v = Velocidade do vento antes das pás;
v_o= Velocidade do vento após as pás.

Mas,

$$m = \rho A d \quad (2.3)$$

ρ = Densidade do ar;
A = Área varrida pelas pás;
d = Distância percorrida pela massa de ar.

Então, a potência associada à energia cinética massa de ar é dada por:

$$P = \frac{\Delta E_C}{\Delta t} = \frac{1}{2}\frac{m(v^2 - v_o^2)}{\Delta t} \quad (2.4)$$

Substituindo a equação *(2.3)* na equação *(2.4)*, tem-se:

$$P = \frac{1}{2}\frac{\rho A d(v^2 - v_o^2)}{\Delta t} \quad (2.5)$$

Mas,

$$v_{méd} = \frac{d}{\Delta t} = \frac{v + v_o}{2} \quad (2.6)$$

Então,

$$P = \frac{\rho A(v + v_o(v^2 - v_o^2)}{4} = \frac{\rho A v^3}{2}\frac{1 + \frac{v_o}{v}\left[1 - \left(\frac{v_o}{v}\right)^2\right]}{2} \quad (2.7)$$

Separando o coeficiente de potência e a potência disponível no vento, temos que a potência extraída pela turbina eólica é dada por:

$$P = \frac{\rho A v^3 C_P}{2} \quad (2.8)$$

Onde,

$$C_P = \frac{\left(1 + \frac{v_o}{v}\right)\left[1 - \left(\frac{v_o}{v}\right)^2\right]}{2} \quad (2.9)$$

A equação *(2.9)* refere-se ao coeficiente de potência da turbina eólica. É um parâmetro adimensional e representa o aproveitamento energético que turbina eólica faz com base na energia cinética disponível do vento.

Através da equação *(2.8)*, pode-se observar a dependência linear que a potência possui em relação à área varrida pelas pás e a dependência cúbica da potência em relação a velocidade do vento.

2.4.1 Características das turbinas eólicas

Conforme COLLIER (2011), a expressão da extração de potência em uma turbina eólica caracterizada pelo coeficiente de potência C_P não pode ser realizada através de métodos ou expressões analíticas, dado que depende de aspectos construtivos do aerogerador, majoritariamente das pás. A expressão do coeficiente de potência utilizado neste trabalho é empírica e dada pela equação:

$$C_P = C_1(C_2\lambda_1 - C_3\beta - C_4\beta^4 - C_5)e^{-C_6\lambda_1} \quad (2.10)$$

Onde

$$\lambda_1 = \left(\frac{1}{\lambda + 0.08\beta} - \frac{0.035}{\beta^3 + 1}\right) \quad (2.11)$$

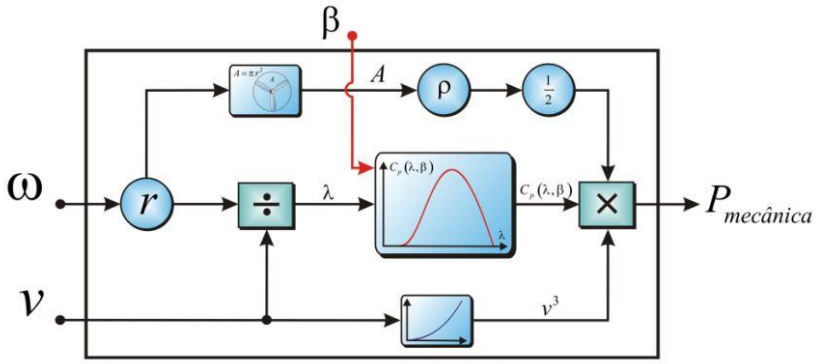
$$\lambda = \frac{r\omega}{v} \quad (2.12)$$

C_n = Coeficientes constantes da turbina, onde n ∈ {1, 2, 3, ..., 6};
λ = Velocidade de ponta da turbina (*Tip Speed Ratio);*
r = Raio da turbina [m];
ω = Velocidade angular da turbina [rad/s];
β = Posição angular das pás *pitch* [º].

Pode-se observar nas equações *(2.11)* e *(2.12)* que além das constantes inerentes aos aspectos construtivos do aerogerador, o coeficiente de potência também é dependente do comprimento das pás, da angulação de *pitch* das pás e da velocidade angular da turbina eólica.

Um digrama de blocos do equacionamento de potência mecânica da turbina pode ser observado na Figura 6.

Figura 6. Diagrama do cálculo de potência mecânica de uma turbina eólica.



Fonte: TIBOLA, 2009.

Na Figura 6 pode-se observar as curvas de potência mecânica por velocidade angular do aerogerador para diferentes velocidades discretizadas de vento utilizando os coeficientes dados no Quadro 1 (HEIER, 2006), além do ângulo de *pitch* e comprimento das pás indicados no Quadro 2. Estes coeficientes serão os coeficientes considerados neste trabalho na modelagem do aerogerador. Como o esperado, assim

como o coeficiente C_P, a potência mecânica disponível pela turbina varia conforme a velocidade angular mecânica do aerogerador e a velocidade de vento imposta.

Quadro 1. Coeficientes da turbina eólica modelada.

C_1	C_2	C_3	C_4	C_5	C_6
0.5	116	0.4	0	5	21

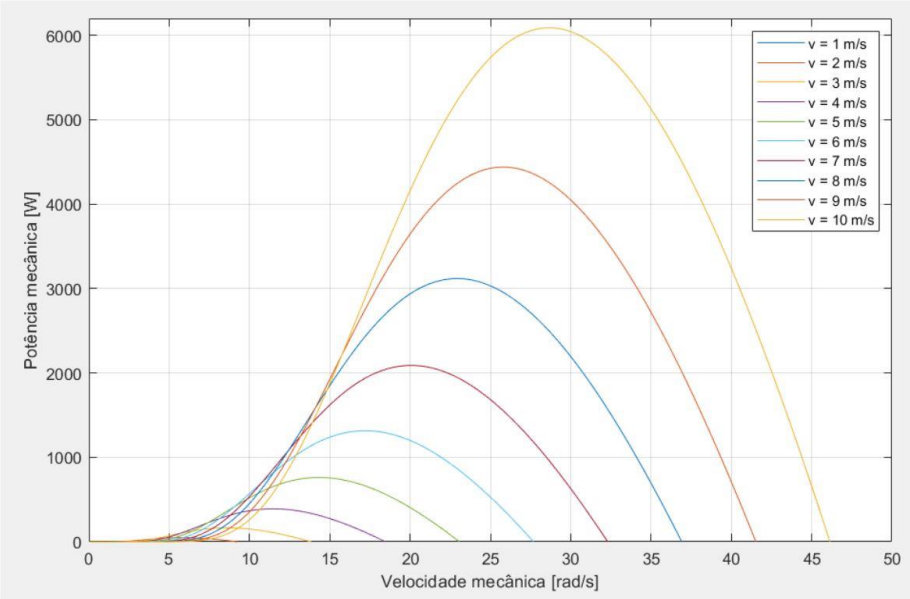
Fonte: HEIER, 2006.

Quadro 2. Características da turbina eólica modelada.

β	r
0º	2.775 m

Fonte: Autor.

Figura 7. Potência mecânica da turbina eólica modelada pela velocidade mecânica no eixo.



Fonte: Autor.

3 PROCESSAMENTO *HARDWARE-IN-THE-LOOP*

A implementação do sistema de emulação de turbina eólica se deu através de processamento *hardware-in-the-loop* (HIL).

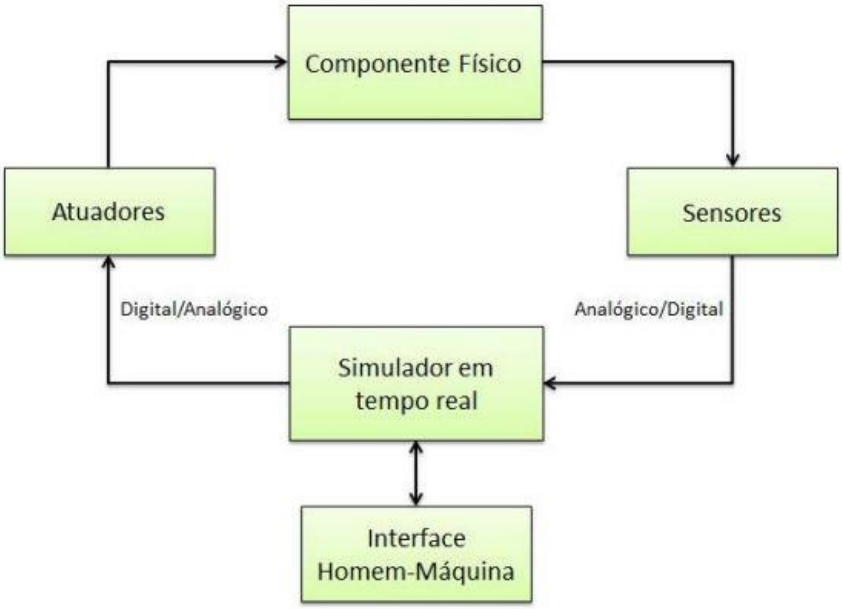
Segundo LOPES (2017), as simulações utilizando *hardware-in-the-loop* são técnicas recentes que apareceram a partir do final da década de 80 com o desenvolvimento de sistemas e *hardwares* computacionais que permitiram processar grandes quantidades de informação em um pequeno intervalo de tempo.

Ainda segundo LOPES (2017), *hardware-in-the-loop* pode ser definido como uma técnica onde o *hardware* dedicado de controle e parte da planta que se deseja testar é utilizado no experimento, enquanto a outra parte da planta é substituída por um modelo matemático de simulação em tempo real.

De acordo com PASSOS (2008), uma das principais vantagens deste sistema de prototipagem é a facilidade da implementação do *hardware* após a validação dos modelos virtuais do sistema, simplificando as etapas de concepção, validação e testes, mitigando assim os custos de desenvolvimento de um novo produto/tecnologia.

O conceito de *hardware-in-the-loop* também foi desenvolvido por GREGA (1999), que o define como o uso de modelos de simulação de um processo conjuntamente com um *target hardware* no controle de uma aplicação de emulação. O modelo de simulação fornece todos os sinais de processamento em tempo real que são então convertidos pelos conversores D/A para o controle dos *hardwares* integrantes do processo. As respostas do sistema físico são então lidas por sensores e retornadas ao *target hardware* via conversores A/D.

Figura 8. Componentes de um sistema *hardware-in-the-loop*.



Fonte: LOPES, 2017.

Os principais componentes de um sistema *hardware-in-the-loop* podem ser definidos em blocos como mostra a Figura 8, que são:

- Componente físico: componente de *hardware* que será controlado pelos sinais enviados pelo modelo matemático implementado através da planta de simulação;
- Sensores: dispositivos que irão ler a realimentação das respostas dos componentes físicos do sistema e os retornarão ao simulador;
- Atuadores: dispositivos que irão acionar os componentes físicos do sistema com base nos sinais produzidos pelo modelo de simulação;
- Simulador em tempo real: responsável pela aquisição, processamento e controle dos componentes físicos do sistema. Através do simulador em tempo real, pode-se implementar o modelo matemático de simulação do componente do sistema a ser analisado.
- Interface homem-máquina: responsável pela comunicação e controle entre o usuário e o sistema emulado. Através da interface homem-máquina, é possível verificar visualmente as respostas e sinais do sistema emulado e realizar o controle através da alteração de parâmetros da simulação em tempo real.

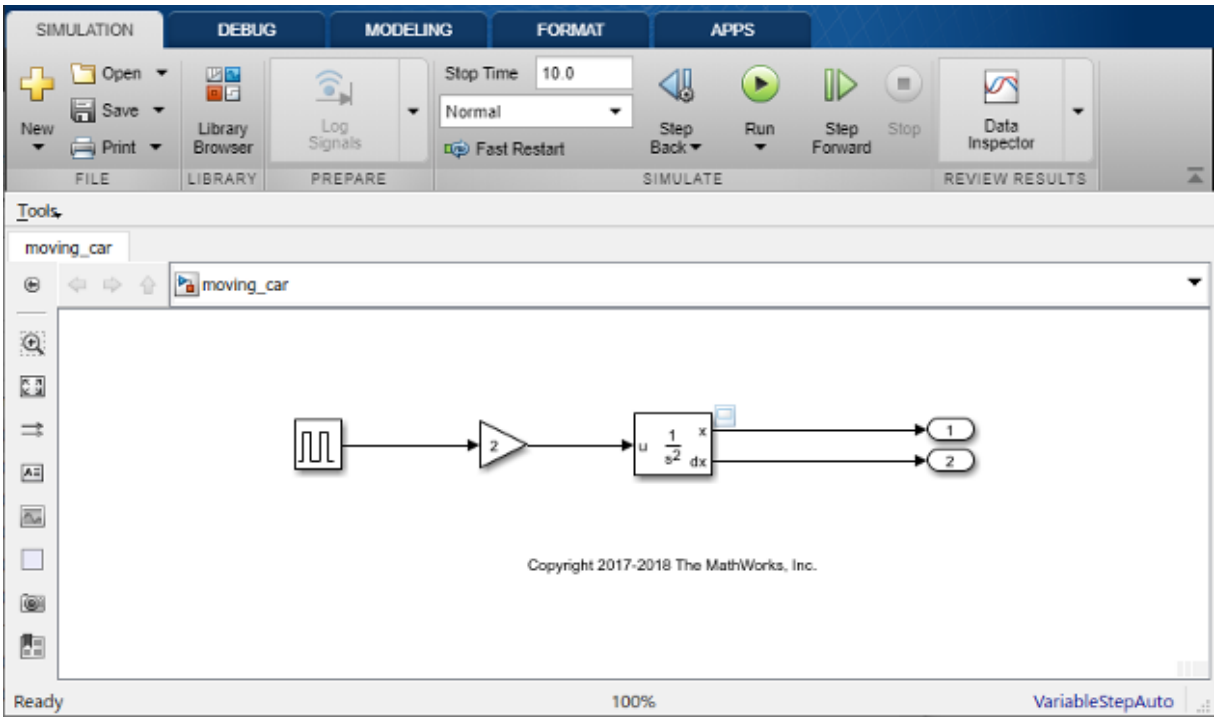
Existem diferentes empresas que fornecem *softwares* e *hardwares* para aplicação em sistemas *hardware-in-the-loop.* São exemplos: OPAL-RT, *Mathworks Simulink Real Time/Speedgoat, National Instruments, Altera, dSPACE*, entre outras.

Devido a utilização prévia do *software* em outros trabalhos e pesquisas desenvolvidas no Instituto de Eletrônica de Potência, este trabalho foi desenvolvido em sua integralidade através da utilização dos pacotes e *toolboxes* do *Mathworks Matlab Simulink Real Time*, que permitiu ao laboratório a utilização de recursos já existentes para a realização do controle da bancada de emulação eólica.

3.1 Simulink

O *Simulink* é uma ferramenta computacional desenvolvida pela *Mathworks* e faz parte do *software Matlab*. É um sistema utilizado em aplicações de modelagem em blocos para simulação e análise de sistemas dinâmicos. Através do *Simulink* é possível realizar a simulação de sistemas físicos e dinâmicos e avaliar o seu comportamento e suas saídas graficamente. A sua interface pode ser observada na Figura 9.

Figura 9. Tela do *software Simulink.*



Fonte: https://www.mathworks.com/help/simulink/gs/create-a-simple-model.html.

Acesso: 03/10/2020

A possibilidade de realizar a modelagem matemática em blocos de sistemas físicos reais com suas características e dinâmicas, associada com a ferramenta *Simulink Real Time* que permite que esses modelos sejam compilados para arquivos executáveis que rodam em máquinas com processamento dedicado para a implementação de simulações em tempo real foi a motivação da escolha do *Simulink* para a modelagem do aerogerador implementado na emulação da bancada eólica.

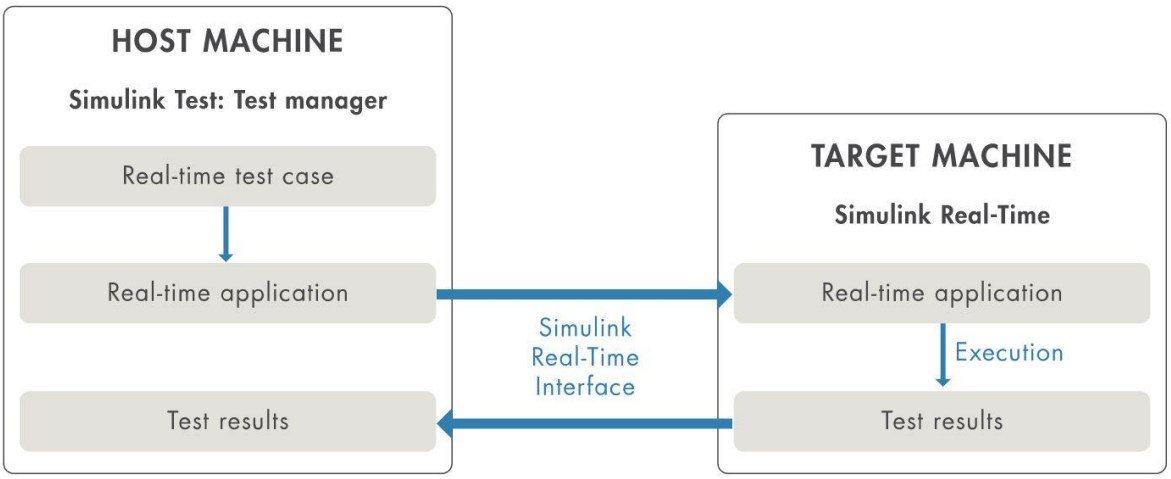
3.1.1 *Simulink Real Time*

O *Simulink Real Time* é uma ferramenta computacional desenvolvida e fornecida pela *Mathworks* como uma *toolbox* do *software Matlab*/Simulink, e por isso requer a sua instalação para desenvolvimento das aplicações para simulação em tempo real. O *Simulink Real Time* permite o desenvolvimento de simulações em tempo real e tarefas de teste como *Rapid Control Prototyping* (RCP) e simulações *Hardware-in-the-Loop* (HIL). O diagrama básico de operação pode ser visualizado na Figura 10.

Os modelos de simulação do *Simulink Real Time* devem rodar em um *target machine* à parte da máquina de desenvolvimento (*host machine*), que pode ser conectado diretamente aos componentes do sistema físico emulado através de blocos de driver I/O (In/Out) disponíveis que permitem a utilização de protocolos de comunicação para a transferência de dados no sistema, tais como UDP, TCP/IP, Serial, CAN, XCP, MIL-STD1553, EtherCAT, entre outros. Para isso, o *Simulink Real Time* conta com compiladores que permitem que os modelos de simulação criados em *Simulink* sejam compilados e executados na *target machine* através do *kernel real time* instalado pelo *Simulink*.

A *target machine* é uma máquina que irá rodar apenas o modelo de simulação da aplicação. Para isso, a máquina necessita de uso dedicado e deve contar com CPU multi-core e placa de rede compatível com os drivers suportados pelo *Simulink Real Time*. Com a utilização de unidades de processamento dedicadas, o desempenho do *Simulink Real Time* é ideal para aplicações onde deseja-se obter um baixo valor de *jitter* e altas frequências de processamento, o que permite que o modelo de simulação na máquina forneça respostas ao sistema emulado em tempo real.

Figura 10. Diagrama de blocos do sistema de simulação do *Simulink Real Time.*



Fonte: https://www.mathworks.com/products/simulink-real-time.html. Acesso: 03/10/2020.

O *Simulink Real Time* também oferece a possibilidade de utilizar blocos de instrumentação para visualização de dados, implementar aplicações interativas e/ou automatizadas e utilizar protocolos de interface.

3.2 API .NET

Juntamente com o *Simulink Real Time*, o *software Matlab* também oferece uma API's (*Application Programing Interface)* permitem a integração dos modelos de simulação desenvolvidos no *Simulink Real Time* com aplicações desenvolvidas em linguagens de programação. Atualmente, a Mathworks disponibiliza duas API's distintas para a integração com linguagens de programação, sendo possível a integração dos modelos *Simulink Real Time* com aplicações em C e .NET.

A API desenvolvida para integração a .NET é disponibilizada através de componentes .NET frameworks que permitem que a simulação seja controlada e visualizada através da comunicação direta entre o *host computer* e o *target computer* por aplicações customizáveis criadas em .NET através do ambiente do Microsoft Visual Studio.

Através da integração do *Simulink Real Time* e aplicações .NET, é possível criar interfaces de usuário que permitem controlar a simulação em tempo real de forma simplificada com controles de instrumentação visual, além de possibilitar visualizar, coletar e arquivar dados da simulação, realizar o *debug* da aplicação, analisar dados de tempo real e implementar o controle da simulação em tempo real de forma *stand-alone*, o que permite que a simulação em tempo real não necessite do *Matlab*/Simulink para ser executada na *target machine* e nem controlada na *host machine,* podendo qualquer computador com os arquivos de instalação da aplicação ser capaz de iniciar a simulação e controlá-la.

3.3 Protocolos de comunicação de rede

Protocolos de comunicação podem ser definidos como um conjunto de regras e padrões estabelecidos para que dispositivos eletrônicos que nem sempre utilizam da mesma linguagem se comuniquem entre si (CASTELUCCI, 2011).

Para permitir a comunicação entre diferentes componentes e dispositivos em uma mesma topologia, as redes de comunicação digital são formadas essencialmente por uma arquitetura em camadas que implementam pilhas de protocolos para diferentes funções (CASTELUCCI, 2011). O modelo teórico mais difundido é o OSI (*Open System Interconnection)*, criado em 1978 para definir exemplos de padrões e

protocolos na conexão entre dispositivos. O modelo OSI prevê as seguintes camadas de rede (CASTELUCCI, 2011).:

- Camada de aplicação: comunicação direta com o usuário que fornece serviços básicos de comunicação.
- Camada de apresentação: define o formato de troca de dados atuando como um tradutor para protocolos, criptografias, compressão de dados etc.
- Camada de sessão: canal de comunicação entre duas aplicações que estão sendo executadas por dispositivos diferentes. Nesta camada são executadas as funções de reconhecimento de nomes e segurança.
- Camada de transporte: é responsável pela integridade dos pacotes de informação, garantindo comunicação confiável.
- Camada de rede: responsável por identificar os endereços dos sistemas na rede e transmitir o dado.
- Camada de link: utilizada para definir como a informação será transmitida pela camada física.
- Camada física: formada pelo *hardware* utilizado para conexão como cabos. A informação nesta camada está contida em sinais físicos como elétricos ou óticos.

Existem diversos protocolos de comunicação que podem ser utilizados nas diferentes camadas de rede do modelo OSI como TCP, ASCII, UDP, IP etc (CASTELUCCI, 2011). Estes protocolos são regidos por elementos-chave que os definem:

- Sintaxe: representa o formato dos dados e a ordem a qual são apresentados. Em suma, definem a ordem e a função dos bytes de uma mensagem.
- Semântica: significado de cada conjunto sintático.
- *Timing:* definição da velocidade de transmissão dos pacotes de mensagem entre as entidades que estão se comunicando

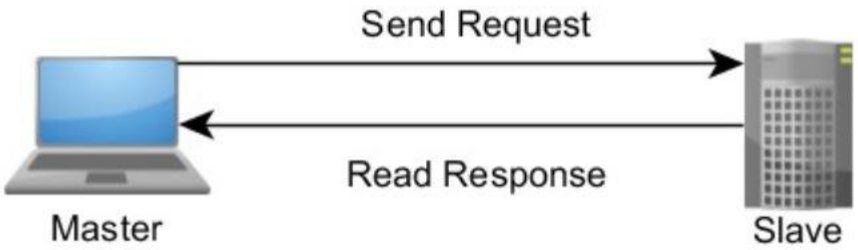
No sistema desenvolvido para implementação da bancada de emulação eólica foram utilizados, além do protocolo proprietário do torquímetro que será discutido posteriormente, dois protocolos universalmente conhecidos: *Modbus* e UDP.

3.3.1 *Modbus*

O protocolo *Modbus* é um protocolo industrial de nível de aplicação (modelo OSI) desenvolvido em 1979 com o objetivo de possibilitar a comunicação entre dispositivos de automação industrial (National Instruments, 2019). Foi concebido inicialmente para aplicações em transferências de dados por camada serial, porém posteriormente foi ampliado para incluir comunicações TCP/IP e UDP.

Essencialmente o *Modbus* é um protocolo que utiliza requisição-resposta em um relacionamento mestre-escravo. Neste tipo de relacionamento, a comunicação ocorre em pares onde o mestre envia uma requisição e aguarda a resposta do escravo (Figura 11). O mestre é geralmente uma interface homem-máquina ou um servidor SCADA, enquanto o escravo é um dispositivo sensor, CLP ou CPA (National Instruments, 2019).

Figura 11. Requisições com protocolo de comunicação *Modbus*.



Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020.

Inicialmente o protocolo *Modbus* era um protocolo simples para comunicação serial, mas posteriormente outras unidades de dados de aplicação foram introduzidos ao protocolo para permitir a utilização de TCP/IP e UDP. Diante disso, o protocolo básico foi separado em unidade de dados de protocolo (PDU) e a camada de rede que define a unidade de dados de aplicação (ADU) (National Instruments, 2019).

3.3.1.1 Unidade de dados de protocolo

O *Modbus* adota a PDU como a base para a especificação da aplicação. É formada por um código de função seguida por um conjunto de dados associados, sendo que as dimensões e conteúdos dos dados são definidos pela função, embora o tamanho máximo da PDU (código de função e dados) não possa ultrapassar 253 bytes totais. A estrutura da PDU é formada por 1 byte de código de função seguido por até 252 bytes de dados (Figura 12).

Os dados acessados e armazenados pelo protocolo *Modbus* estão contidos em quatro faixas de endereços: coils, entradas discretas, registradores holding e registradores de entrada, embora essas denominações possam variar. Cada faixa de endereço possui um tipo e direito de acesso dos dados contidos. As especificações dos endereçamentos *Modbus* são definidos no Quadro 3:

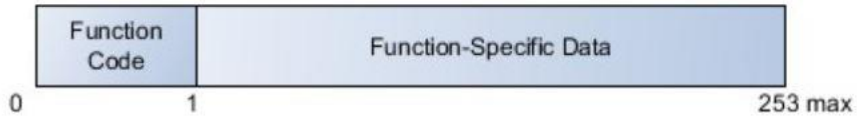
Quadro 3. Tipo de dados e permissões utilizados pelo protocolo *Modbus*.

Bloco de memória	**Tipo de dados**	**Acesso ao mestre**	**Acesso ao escravo**
Coils	Booleano	Leitura/escrita	Leitura/escrita
Entradas discretas	Booleano	Somente leitura	Leitura/escrita
Registradores holding	Palavra não sinalizada	Leitura/escrita	Leitura/escrita
Registradores de entrada	Palavra não sinalizada	Somente leitura	Leitura/escrita

Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020

A especificação do *Modbus* define que cada bloco contém espaço de endereçamento de 65536 (2^{16}) elementos, indo do elemento de endereço 0 até 65535.

Figura 12. PDU do protocolo *Modbus*.



Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020.

O protocolo *Modbus* também especifica que as faixas de endereçamento de dados utilizados pelo protocolo tenham um esquema de numeração que inclui prefixos ao endereço do dado para especificar à qual faixa de endereçamento o dado pertence.

A faixa de registradores holding, por exemplo, possui o prefixo 4 seguido pelo endereço do elemento específico (4xx.xxx). O registrador holding 13 nesta especificação seria representado então por 40.013. Os prefixos de endereçamento de dados são definidos no Quadro 4:

Quadro 4. Prefixos utilizados por tipo de dados no protocolo *Modbus*.

Bloco de dados	**Prefixo**
Coils	0
Entradas discretas	1
Registradores de entrada	3
Registradores holding	4

Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020.

Os códigos de função padrão que compõe a PDU são especificados pelo protocolo *Modbus* e definem a operação a ser realizada. Os principais códigos utilizados na comunicação pelo protocolo *Modbus* são definidos no Quadro 5:

Quadro 5. Código das principais funções do protocolo *Modbus*.

Código	**Descrição**
1	Read Coils
2	Read Discrete Inputs
3	Read Multiple Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Register
7	Read Exception Status
16	Write Multiple Registers

Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020

3.3.1.2 Unidade de dados de aplicação

As funções e dados que formam a PDU juntamente com a utilização de protocolos de rede (ADU) formam a mensagem *Modbus*. Os protocolos de rede mais utilizados para comunicação pelo protocolo *Modbus* são o TCP/IP, Serial e UDP. Para cada protocolo de rede suportado, o *Modbus* inclui uma variante do ADU que permite a sua utilização (National Instrument, 2019).

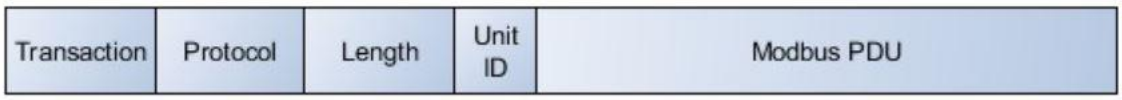
Entre as variantes de ADU's disponíveis pelo protocolo *Modbus*, existem recursos que são comuns à todas, sendo:

- PDU: cada ADU possui uma PDU que compõe a mensagem;
- Unit ID/Adress: fornece informações de roteamento à camada de aplicação;
- Verificação de erro;
- Mecanismo de determinação de início e fim da mensagem;

3.3.1.3 *Modbus* TCP/IP

As ADU's para TCP/IP são formadas por um cabeçalho do *Modbus* Application Protocol (MBAP) concatenado com a PDU do *Modbus*, sendo que o MBAP é um cabeçalho de uso geral e portanto, depende de uma camada de rede confiável (NATIONAL INSTRUMENTS, 2019).

Figura 13. ADU protocolo *Modbus*



Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020.

- *Transaction*: responsável por rotular as mensagens para que o mestre e o escravo possam processar e responder as requisições/respostas assincronicamente;
- *Protocol*: identificador do protocolo, geralmente possui valor nulo, mas pode ser utilizado para expandir o comportamento do protocolo;
- *Lenght*: explicita o comprimento do restante do pacote;

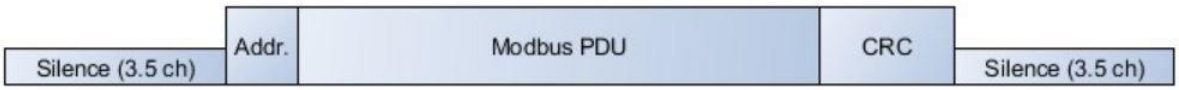
- *Unit ID*: não é utilizado por dispositivos TCP/IP, porém permite que o pacote possa ser convertido por gateways para conexão com dispositivos seriais;
- PDU: unidade de dados de aplicação padrão.

3.3.1.4 *Modbus* RTU

A ADU para aplicações seriais RTU são mais simples que as ADU utilizadas em protocolos de rede TCP/IP. Apesar da simplicidade, os pacotes transmitidos via protocolo de rede serial RTU necessitam de um par de intervalos de silêncio de 3.5 caracteres para definir o início e fim de um pacote, o que causa inerentemente problemas de desempenho dada a necessidade de espera do tempo de silêncio para iniciar o processamento do pacote, e de confiabilidade ao utilizar *baud rates* muito altas que acabam introduzindo lacunas de tamanhos variáveis e confundindo o código que recebe o pacote e o processa, causando problemas de interpretação e corrompimento do pacote (NATIONAL INSTRUMENTS, 2019).

A estrutura da ADU para aplicações seriais RTU pode ser observada na Figura 14:

Figura 14. ADU da comunicação serial *Modbus*.



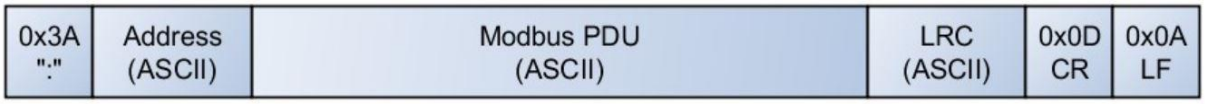
Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020

- *Address*: define para qual escravo o pacote é destinado. Se é definido como endereço 0, a mensagem é difundida por toda a rede e todos os escravos processarão a requisição;
- *Modbus* PDU: unidade de dados de aplicação padrão;
- CRC: responsável por garantir a integridade dos dados;

3.3.1.5 *Modbus* ASCII

A ADU do protocolo de rede ASCII é mais complexa que a RTU para evitar os problemas advindos dos pacotes RTU. A ADU do protocolo ASCII prevê a utilização de caracteres marcadores de início e fim do pacote, justamente para garantir a confiabilidade da mensagem mesmo na utilização de *baud rates* altos. A sua desvantagem é de que todos os dados são transferidos como caracteres ASCII, o que duplica os dados a serem transferidos pela rede serial (NATIONAL INSTRUMENTS, 2019). A sua estrutura pode ser observada na Figura 15:

Figura 15. Estrutura da ADU *Modbus* ASCII.



Fonte: https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html. Acesso: 03/10/2020.

- “:” (0x3A) : caractere marcador do início da mensagem;
- *Address*: define para qual escravo o pacote é destinado. Se é definido como endereço 0, a mensagem é difundida por toda a rede e todos os escravos processarão a requisição;
- *Modbus* PDU: unidade de dados de aplicação padrão;
- LRC: checagem de redundância para garantir a integridade dos dados;
- CR + LF: caracteres marcadores que definem o final da mensagem;

3.3.2 Protocolo UDP

O protocolo UDP é um protocolo não orientado a conexão correspondente à camada 4 do modelo OSI (transporte) assim como o protocolo TCP. Porém, diferentemente do protocolo TCP, o protocolo UDP não possui verificação de integridade de dados, verificação do recebimento do pacote pelo destino e não possui controle de fluxo de pacotes e ordem de mensagens. Mediante estas características, apesar do protocolo UDP ser simplificado e mais veloz em relação ao TCP, trata-se

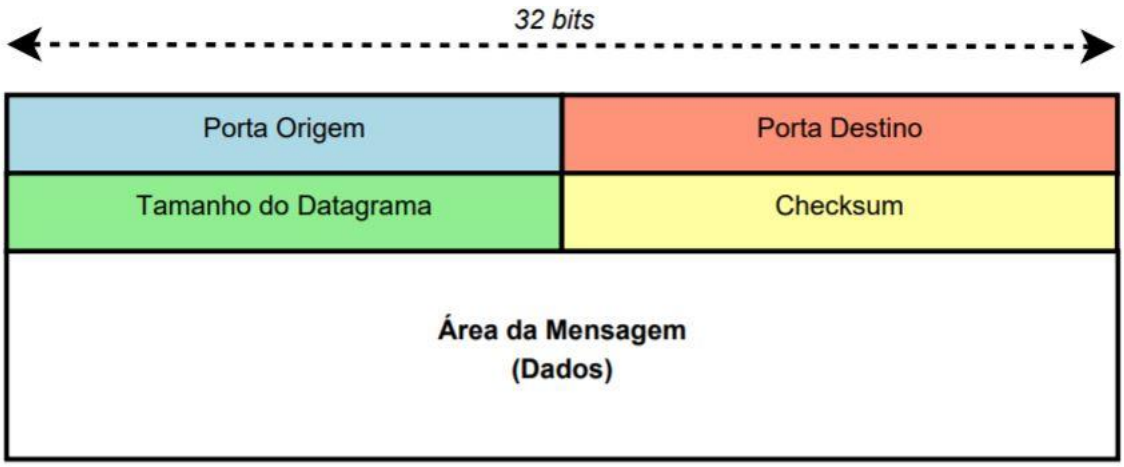
de um protocolo menos confiável, onde a integridade dos dados do pacote enviado não é garantida (CBPF, 201?).

Entretanto, a sua característica de ser um protocolo simples e consequentemente menor o faz ser o protocolo preferencial em relação ao TCP quando a velocidade da comunicação é mais importante que a sua integridade.

O datagrama do protocolo UDP é representado na Figura 16. Neste, pode-se perceber a simplicidade de implementação do pacote de informações no protocolo UDP, cujo cabeçalho possui apenas as informações de porta de origem do pacote enviado (16 bits), porta de destino (16 bits), tamanho do datagrama com base na mensagem e no cabeçalho (16 bits), *checksum* para conferência de integridade do pacote (16 bits) e mensagem de dados (MICHEL, 2010). É importante salientar que apesar da existência do *checksum* para a conferência da integridade dos dados por parte do dispositivo de destino, caso haja discrepâncias entre o *checksum* e o datagrama recebido o pacote é descartado sem solicitar o reenvio da mensagem.

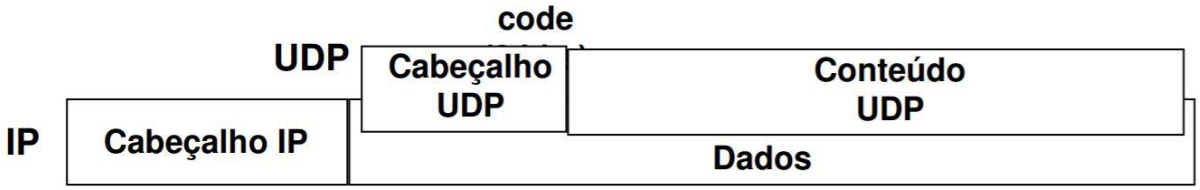
O protocolo UDP utiliza *sockets* para definir as portas para comunicação com dispositivos e para isso também utiliza o protocolo IP para empacotamento das mensagens UDP definindo o endereçamento do dispositivo destinatário do pacote enviado. O datagrama resumido da mensagem UDP/IP é apresentada na Figura 16.

Figura 16. Datagrama das mensagens do protocolo UDP.



Fonte: MICHEL, 2010.

Figura 17. Estrutura mensagem UDP.



Fonte: BERNAL, 2016

4 DESENVOLVIMENTO DA APLICAÇÃO

A aplicação para emulação da turbina eólica foi desenvolvida utilizando as ferramentas: *Simulink Real Time* para modelagem matemática da turbina eólica e do sistema de controle, a plataforma .NET para desenvolvimento da aplicação em C# que realiza a visualização e controle no *host computer* da emulação implementada pelo modelo matemático gerado e executado no *target computer,* e a API disponibilizada pela Mathworks que permite a chamada e execução de métodos para interação da aplicação de controle desenvolvida em .NET com a aplicação em *Simulink Real Time* rodando na *target machine*.

O desenvolvimento do sistema de emulação consistiu então na modelagem matemática da turbina eólica e da malha de controle para acionamento do motor que emula a turbina eólica e o desenvolvimento da aplicação de visualização/instrumentação no *host computer* que realiza a comunicação com o modelo de simulação que roda na *target machine* e com os demais periféricos do sistema.

Os materiais utilizados na composição da bancada de emulação da turbina eólica foram:

- Motor de Ímãs Permanentes com fluxo radial
    - Tensão: 380 V;
    - Potência: 11 kW;
    - Velocidade máxima: 1800 RPM;
    - Torque máximo: 5,9 kg.f.m;
    - Peso: 50 kg;
    - Número de polos: 6;

- Gerador de Ímãs Permanentes com fluxo radial
    - Tensão: 380 V;
    - Potência: 15 kW;
    - Velocidade máxima: 1800 RPM;
    - Torque máximo: 8,12 kg.f.m;
    - Peso: 80 kg;
    - Número de polos: 6;

- Inversor CFW 11 WEG
    - Tensão de entrada: 380 - 480 V;
    - Potência: 2 - 600 cv;
    - Interface: Interface de Operação Remota;
    - Protocolo para acionamento: *Modbus-RTU*;
    - Módulo de comunicação serial: Kit RS232-01

- Transdutor de Torque
    - Taxa de erro máximo: ± 0.1;
    - Tensão de alimentação: 12 a 28 V;
    - Corrente de alimentação: < 60 mA;
    - Tensão de saída: ± 5 V;
    - Sample rate: 200 amostras/segundo;
    - Conexão com o PC: USB;

- *Simulink Real Time*
    - *Software* utilizado para modelagem da simulação da turbina eólica e sistema de controle segundo os parâmetros coletados pelos sensores e dados de entrada do sistema.

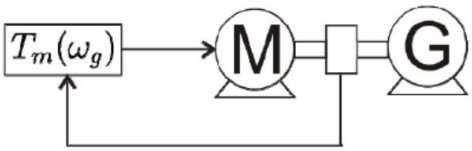
- Aplicação de controle
    - Aplicação com GUI (Interface Gráfica do Utilizador) desenvolvida na plataforma .NET através de linguagem de programação C# para controle da emulação e comunicação com periféricos.

4.1 Diagrama de blocos

A Figura 18 representa o modelo simplificado do funcionamento da emulação de um aerogerador. O motor M é a força motriz que emula o torque/potência gerada por uma turbina eólica dada uma velocidade de vento qualquer. O motor aciona o gerador representado pelo bloco G, responsável pela conversão eletromecânica de energia. O torque/velocidade gerado pelo motor M é controlado através do bloco $Tm(\omega g)$ que recebe os parâmetros de realimentação velocidade mecânica no eixo e

torque gerado de um sensor torquímetro e os utiliza para recalcular a velocidade que será imposta ao motor com base no modelo matemático da turbina eólica estudada.

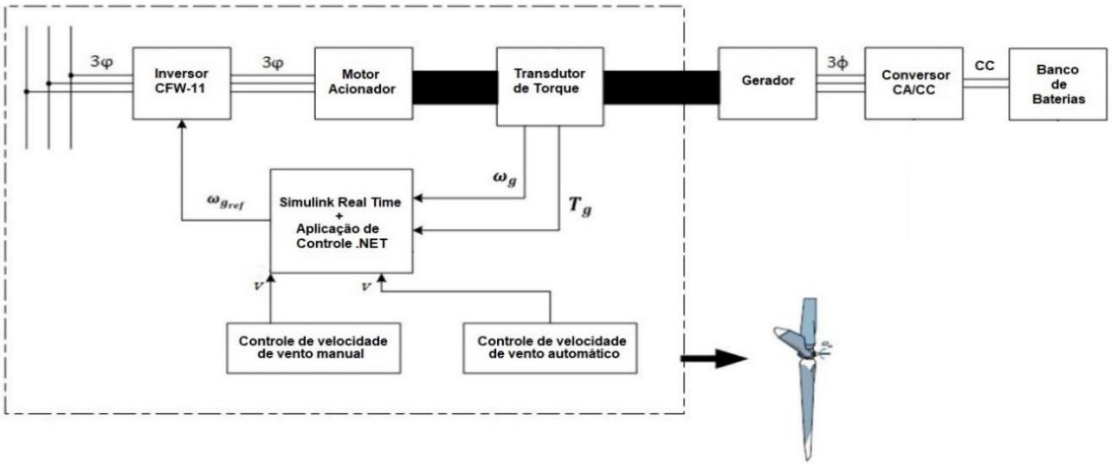
Figura 18. Modelo de emulação com um motor M, um sensor de torque, um gerador G e um bloco que realimenta e controla todo o sistema.



Fonte: CALLENGARO, 2010.

A bancada de emulação eólica possui o intuito de emular o funcionamento de uma turbina eólica para integração junto as demais fontes despacháveis da microrrede na qual foi implementada. O diagrama de blocos da bancada de emulação contemplando a sua aplicação final na microrrede é visualizada na Figura 19.

Figura 19. Diagrama de blocos e fluxograma da bancada de emulação eólica em sua aplicação final na microrrede.



Fonte: Adaptado de Righi, 2015.

O diagrama de blocos sintetiza de forma resumida o sistema de emulação implementado, e é composto pelos seguintes elementos:

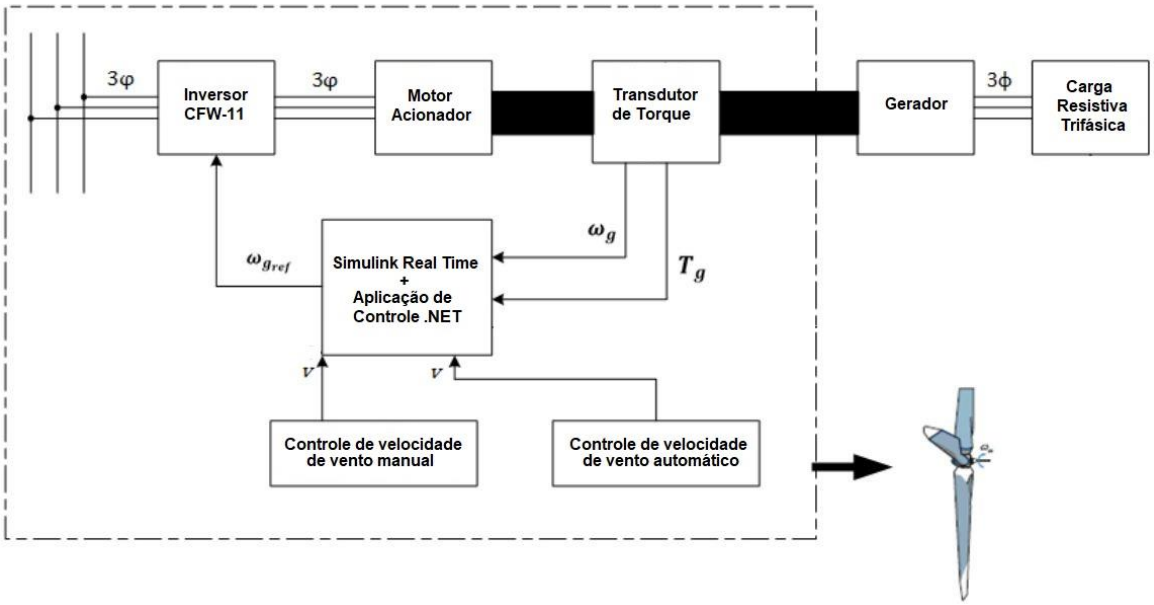
- Aplicação de controle .NET + *Simulink Real Time*: compreende o modelo matemático de simulação implementado na *target machine* e a aplicação de interface e controle desenvolvida na plataforma .NET implementada na *host machine*. É responsável por executar a simulação e realizar a comunicação para envio e recebimento de dados no sistema emulado. Seu principal

parâmetro de entrada definido pelo usuário é a velocidade de vento imposta ao aerogerador na altura do cubo;

- Inversor: Recebe do bloco de aplicação de controle a referência de velocidade mecânica calculada e a utiliza como referência para acionamento do motor da emulação;
- Motor: ao ser acionado pelo inversor, imprime a velocidade de referência gerando um torque no seu eixo acoplado ao gerador. É responsável por emular o funcionamento mecânico da turbina eólica;
- Transdutor de Torque: acoplado entre o eixo do motor e o eixo do gerador, faz a leitura dos dados de torque mecânico, velocidade mecânica e os envia à aplicação de controle para realimentação da malha;
- Gerador: converte a velocidade e o torque mecânicos em seu eixo acionado pelo motor em potência elétrica;
- Conversor CA/CC: responsável por converter as tensões e correntes AC em tensões e correntes DC, além de executar em seu sistema de controle algoritmos para extração da máxima potência disponível (MPPT).
- Banco de baterias: responsável por armazenar a energia gerada através do sistema de emulação motor-gerador.

Entretanto, devido à indisponibilidade das baterias e do conversor CA/CC previamente instalados na microrrede, o sistema de emulação implementado baseou-se em uma topologia alternativa, que se utilizou da substituição do conversor CA/CC e do banco de baterias por uma carga resistiva trifásica no intuito de simplificar e dar prosseguimento ao desenvolvimento do trabalho. O diagrama de blocos do sistema de emulação eólica implementado pode ser observado na Figura 20.

Figura 20. Diagrama de blocos e fluxograma da bancada de emulação eólica implementado neste trabalho.



Fonte: Adaptado de RIGHI, 2015.

Nesta topologia a bancada de emulação eólica não possui o comportamento real de um sistema eólico utilizado na maioria das aplicações, uma vez que não é possível realizar a rastreabilidade do ponto de máxima potência (MPPT) que é implementado através do conversor CA/CC, e, portanto, não há otimização da geração. Entretanto, ainda é possível avaliar o comportamento da turbina eólica utilizando-se de uma carga estática trifásica resistiva.

4.2 Transdutor de torque

O transdutor de torque utilizado para realizar a medição dos valores de torque e velocidade mecânica angular no eixo do motor acionador foi o T25 *High Speed Rotary Torque Transducer* da Interface. As características de aquisição e comunicação deste transdutor podem ser observadas no Quadro 6.

Quadro 6. Características do transdutor de torque T25 *High Speed Rotary Torque* da Interface.

Transdutor de Torque T25	**Características**
Variação máxima (erro)	± 0,1 N.m
Tensão de alimentação	12 a 28 V
Corrente de alimentação	≤ 60 mA
Tensão de saída	± 5 V
Sampling rate	10 kHz
Comunicação	USB

Fonte: Adaptado de RIGHI, 2015.

O transdutor de torque T25 *High Speed Rotary Torque Transducer* possui conexão direta com a *host machine* através de comunicação serial USB, utilizando-se de protocolo proprietário da Interface. Este protocolo proprietário possui suas próprias configurações e mensagens de comando.

4.2.1 Protocolo de comunicação

O protocolo de comunicação (INTERFACE, [200-?]) utilizado pelo T25 *High Speed Rotary Torque Transducer* é um protocolo proprietário e utiliza as seguintes características de comunicação:

- Baud rate de 230400 bits/s;
- Mensagem com 8 bits de dados;
- Sem paridade;
- 1 stop bit;
- 10µs de silêncio entre mensagens.

O formato das mensagens enviadas através do protocolo proprietário da Interface pode ser observado na Quadro 7.

Quadro 7. Formato das mensagens enviadas pelo protocolo proprietário do transdutor de torque.

STX	Command Byte	RX	TX	Number of Parameter bytes	Parameters	Checksum	Weight Checksum

Fonte: Interface, 200?

- STX: Na primeira parte da mensagem, o byte STX é definido por padrão pelo protocolo proprietário como o valor em hexadecimal 0x02. Este valor sinaliza o início da mensagem.

- *Command Byte*: o *Command Byte* (ou byte de comando) é um valor de dois bytes em hexadecimal que sinaliza qual é o comando a ser executado pela mensagem.

- RX e TX: Os bytes RX e TX indicam o dispositivo que receberá e o dispositivo que está enviando, respectivamente, a mensagem ou comando. É representado por um valor de um byte em hexadecimal, e possui disponibilidade de endereços de 1 à 249, sendo o endereço 0 dedicado ao *broadcast.*

- *Number of Parameter Bytes*: Indica previamente quantos *bytes* os parâmetros da mensagem enviada/lida têm.

- *Parameters*: Contém a informação efetiva da mensagem. Pode ser referente a valores de torque e velocidade a serem lidos, bem como parâmetros de execução do comando da mensagem enviada. Possui tamanho variado conforme a aplicação.

- *Checksum e Checksum Weighted*: Ambos possuem a finalidade de realizar a verificação da integridade da mensagem enviada e recebida. Realizam o cálculo do número de bytes anteriores ao *Checksum* e posteriores ao *byte* de inicialização da mensagem STX.
  O *checksum* é realizado segundo a seguinte equação:

$$Checksum(i) = X_1 + X_2 + X_3 + \cdots + X_i \tag{4.1}$$

  O *checksum weighted* é realizado segundo a seguinte equação:

$$WeightedChecksum(i) = iX_1 + (i-1)X_2 + \cdots + X_i \tag{4.2}$$

4.2.2 Comandos e interpretação dos dados

O protocolo proprietário de comunicação do transdutor T25 da Interface prevê vários comandos de operação, entretanto a aplicação foi desenvolvida utilizando apenas dois comandos: *Speed Optimized Polling Mode (SOPM)* e *Restart.*

- *Speed Optimized Polling Mode*

O comando *Speed Optimized Polling Mode* faz parte de um comando mais geral chamado *Goto Special Mode* e é utilizado para requisitar ao transdutor de torque o envio de infinitas amostras de valores de torque e velocidade.

A taxa de medidas por segundo enviadas através deste modo de leitura pode chegar a 5000 medidas por segundo utilizando a *baud rate* de 230400 bit/s, entretanto para a aplicação desenvolvida, dada a baixa taxa de atualização do inversor requerida nesta aplicação, não é necessário mais do que a taxa mínima de 200 medidas por segundo disponível neste protocolo.

O modelo de mensagem enviado pelo *host machine* ao transdutor de torque para iniciar o *polling* de envio de medidas com um *sample rate* de 200 medidas por segundo pode ser visto na Quadro 8.

Quadro 8. Exemplo de comando enviado ao transdutor de torque Interface.

0x02	0x5A	0x01	0xFF	0x05	0x03	0x19	0x00	0x00	0x43	0xBE	0x03

Fonte: RIGHI, 2015

Um exemplo de resposta a esse comando é a mensagem do Quadro 9:

Quadro 9. Resposta ao comando.

0x41	0xFF	0x94	0x00	0x64

Fonte: Adaptado de RIGHI, 2015.

O primeiro byte da mensagem, 0x41, refere-se ao contador. A cada nova mensagem enviada pelo transdutor o contador é acrescido de uma unidade até atingir *overflow.* Os dois bytes conseguintes, 0xFF e 0x94, concatenados formam o valor de torque serial, que precisa passar por uma conversão que será analisada na próxima seção. Os dois últimos bytes, 0x00 e 0x64, concatenados formam a velocidade mecânica em rotações por minuto (RPM), bastando converter de hexadecimal para decimal. Neste caso, o valor 0x0064 representa a velocidade de 100 RPM.

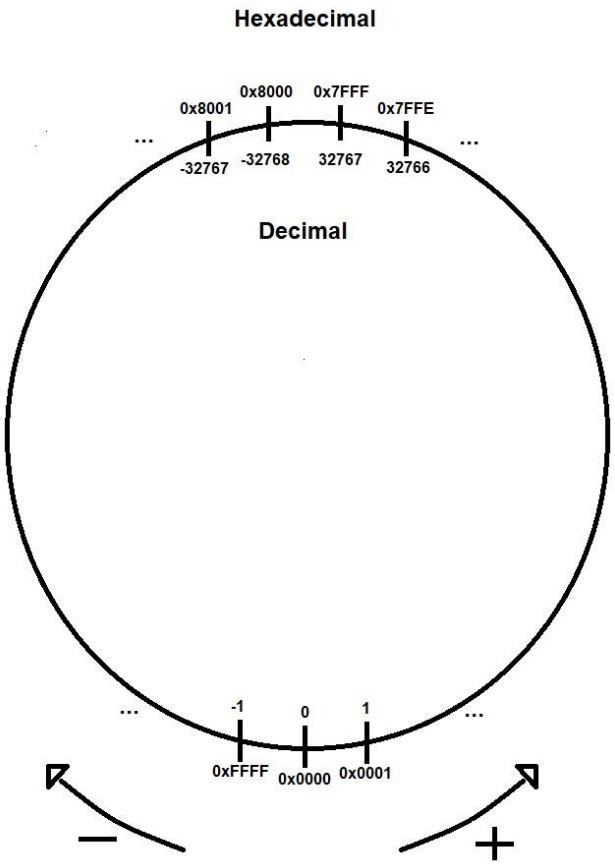
4.2.3 Interpretação dos dados de torque

Utilizando como exemplo a mensagem de resposta ao comando *Speed Optimized Polling Mode* do Quadro 9, o dado de torque contido na mensagem é a concatenação do segundo e terceiro byte da mensagem, 0xFF94. Segundo (INTERFACE, 2020), o torque dentro do protocolo proprietário de comunicação é, portanto, um valor em hexadecimal de dois bytes que possui um *range* que vai de 0x8000 = -32768 decimal até 0x7FFF = 32767 decimal. Os sensores da Interface são calibrados para funcionar com precisão entre 0x9E58 = -25000 decimal à esquerda e 0x61A8 = 25000 decimal à direita. Valores abaixo de -25000 até -32768 e acima de 25000 até 32767 são considerados *overload* e não possuem garantia de precisão. Um melhor entendimento pode ser obtido observando-se a Figura 21.

Os torques seriais decimais positivos e negativos indicam o sentido de rotação do eixo do transdutor de torque. Valores negativos indicam um sentido de rotação anti-horária, enquanto valores positivos indicam sentido de rotação horária.

Para calcular o valor real de torque em N·m da medida de torque serial recebida, o valor de torque serial convertido para decimal deve ser dividido pelo fundo de escala 25000 (25000 para valores positivos e -25000 para valores negativos), e multiplicado pelo valor nominal da resolução de torque utilizado pelo transdutor.

Figura 21. Ilustração da contagem de torque hexadecimal utilizada pelo transdutor de torque da Interface.



Fonte: Autor.

Tomando como exemplo o valor de torque serial em hexadecimal contido na mensagem do Quadro 9, 0xFF94, por ser um valor acima de 0x7FFF concluímos que se trata de um valor negativo (sentido de rotação anti-horária). Sendo um valor negativo, encontra-se o torque serial decimal da seguinte forma:

$$TorqueDecimal_{serial} = TorqueHex - 0xFFFF - 1 \quad (4.3)$$

$$TorqueDecimal_{serial} = 0xFF94 - 0xFFFF - 1 = -108 \quad (4.4)$$

Com o $TorqueDecimal_{serial}$, considerando o transdutor de torque operando com uma resolução de 2000 N·m, calcula-se o torque real em N.m:

$$Torque_{real} = \frac{TorqueDecimal_{serial} \times Resolução_{torquímetro}}{\pm FundoEscala_{torquímetro}} \quad (4.5)$$

$$Torque_{real} = \frac{-108 \times 2000}{-25000} = 8.64\ N.m \quad (4.6)$$

- *Restart*

O comando *restart* é utilizado para limpar os buffers de comunicação do transdutor de torque, bem como reiniciar a comunicação entre o *host computer* e o transdutor. É utilizado previamente toda vez em que a aplicação é iniciada ou reiniciada no *host computer*. A mensagem deste comando é simples, sendo constituída unicamente pelo byte 0x02.

4.3 Inversor de Acionamento

O inversor de frequência utilizado no sistema de emulação de turbina eólica é o modelo CFW-11 da WEG. As suas características básicas estão apresentadas no Quadro 10.

Quadro 10. Características do inversor WEG CFW-11.

Inversor CFW-11 WEG	**Características**
Tensão de entrada	380 – 480 V
Potência	2 – 600 CV
Interface	Interface de Operação Remota
Protocolo de comunicação	*Modbus-RTU*
Módulo de comunicação serial	RS232-01

Fonte WEG, 2010.

Como pode ser analisado, o protocolo de comunicação utilizado pelo Inversor CFW-11 é o *Modbus-RTU* baseado no módulo de comunicação serial RS232-01, uma interface física de comunicação serial.

As propriedades do protocolo de comunicação serial do Inverso CFW-11 são utilizadas nesta aplicação foram:

- *Baud rate* de 9600 bits/s;
- Mensagem com 8 bits de dados;
- Sem paridade;

- 1 stop bit;
- 4,001ms de silêncio entre mensagens.

O protocolo de comunicação utilizado pelo Inversor CFW-11 possui as mesmas características e estruturas de mensagem do protocolo *Modbus-RTU* discutido previamente na seção 3.3.1.4. Para todas as funções disponíveis pelo protocolo de comunicação do CFW-11, seja leitura ou escrita de *coils*, entradas discretas, registradores *holding* ou registradores de entrada o tempo de resposta entre o final da transmissão da mensagem do dispositivo mestre e a resposta do dispositivo escravo é de 2 à 10 milissegundos (WEG, 2010).

4.3.1 Comandos

O Inversor CFW-11 possui diversos comandos de operação e configurações que podem ser modificadas, entretanto nesta aplicação as funções que são utilizadas para controle do motor de acionamento são as conseguintes.

Quadro 11. Comando para controle do inversor por porta serial.

Comando	Valor
P0222	9

Fonte: WEG, 2010.

O comando do Quadro 11 é utilizado para definir a porta USB/Serial do Inversor CFW-11 como porta referencial para controle do inversor (WEG, 2010). O comando P0222 refere-se ao registrador de *holding* na posição 0222, enquanto o valor refere-se ao dado decimal a ser escrito no registrador de *holding*. Este comando precisa ser executado apenas uma vez, somente quando o inversor estiver com a sua referência de controle remoto selecionada para outra fonte.

Quadro 12. Comando para definir a porta serial como referência no acionamento/parada do inversor.

Comando	Valor
P0227	2

Fonte: WEG, 2010.

O comando do Quadro 12 é utilizado para selecionar a porta USB/Serial como porta remota referencial para iniciar ou parar o acionamento do inversor. O comando P0227 refere-se ao registrador holding na posição 0227, enquanto o valor refere-se ao dado decimal a ser escrito no registrador de holding. Novamente, este comando precisa ser executado apenas uma única vez no caso de não estar selecionado previamente a porta USB/Serial como referencial.

Quadro 13. Comando para iniciar/interromper o acionamento do inversor.

Comando	Valor
P0682	0 / 3

Fonte: WEG, 2010.

O comando do Quadro 13 é utilizado para acionar o inversor remotamente (valor 3) ou interromper o acionamento remotamente (valor 0). O comando P0682 refere-se ao registrador *holding* na posição 0682, enquanto o valor refere-se ao dado decimal a ser escrito no registrador de *holding*.

Quadro 14. Comando para envio de referência de velocidade mecânica ao inversor

Comando	Valor
P0683	Ref. Velocidade

Fonte: WEG, 2010.

O comando do Quadro 14 é utilizado para alterar a referência de velocidade mecânica angular a qual o inversor acionará o motor de emulação da aplicação. O comando P0682 refere-se ao registrador *holding* na posição 0682, enquanto o valor refere-se ao dado decimal a ser escrito no registrador de *holding*.

A velocidade de referência considerada pelo inversor é dada em rotações por minuto (RPM), entretanto o valor de velocidade em RPM não é gravado diretamente no registrador sem antes ser tratado para que possa ser interpretado corretamente.

A interface de comunicação do inversor CFW-11 define o valor hexadecimal 0x2000 (8192 decimal) como a velocidade síncrona em RPM do motor que está acionando (fundo de escala), enquanto o valor 0x0000 é referenciado como velocidade nula (WEG, 2010). No caso de um motor síncrono de 6 polos (1800 RPM) como o que é utilizado para emulação da turbina eólica neste projeto, a velocidade de

referência deve primeiramente ser tratada antes de ser enviada ao inversor pela seguinte transformação (WEG, 2010):

$$Velocidade_{serial} = \frac{Velocidade_{RPM} \times 8192}{1800} \qquad (4.7)$$

Após ser tratada pela equação *(4.7)*, a $Velocidade_{serial}$ pode ser enviada ao inversor para que seja obtida a $Velocidade_{RPM}$ no eixo do motor. É importante ressaltar que o inversor aceita apenas valores inteiros de velocidade mecânica angular em RPM, logo valores de velocidades com casas decimais são arredondados antes de serem enviados ao inversor.

4.4 Motor de emulação

O motor de acionamento utilizado para emular o funcionamento da turbina eólica é um motor síncrono de ímãs permanentes WEG WMagnet. Os dados de suas principais características podem ser encontrados no Quadro 15.

Quadro 15. Características motor síncrono WEG WMagnet.

Motor de Simulação de Vento	**Motor WMagnet - WEG**
Tipo	Motor síncrono de ímãs permanentes
Tensão	380 V
Potência	11 kW
Velocidade Máxima	1800 RPM
Torque Máximo	5.9 kg.f.m
Peso	50 kg
Nº de polos	6
Acionamento	Inversor CFW-11 WEG

Fonte: dados de placa WEG.

4.5 Gerador

O gerador utilizado para realizar a conversão eletromecânica de energia advinda do motor de emulação é um motor síncrono de ímãs permanentes WEG WMagnet. Os dados com as principais características do motor encontram-se no Quadro 16.

Quadro 16. Características gerador síncrono WEG WMagnet.

Gerador	**Motor Wmagnet - WEG**
Tipo	Motor síncrono de ímãs permanentes
Tensão	380 V
Potência	15 kW
Velocidade Máxima	1800 RPM
Torque Máximo	8,12 kg.f.m
Peso	80 kg
Nº de Polos	6

Fonte: dados de placa WEG.

Com o objetivo de dimensionar a carga resistiva trifásica implementada nesta aplicação, foi realizado um teste simples para encontrar a constante de tensão K da máquina síncrona que relaciona a tensão nos terminais da trifásicos do estator máquina com a velocidade mecânica angular de acionamento do rotor em rotações por minuto.

O teste foi realizado alterando-se a velocidade mecânica angular do rotor entre 100 RPM à 600 RPM com espaçamentos de 50 RPM e medindo-se as tensões de linha em circuito aberto existente entre as fases do gerador para cada velocidade. Esta abordagem é uma aproximação que não considera as perdas indutivas e resistivas existentes no estator do gerador, mas que é válida dentro dos propósitos deste trabalho.

As velocidades e as tensões observadas podem ser observadas na Tabela 1.

Tabela 1. Dados de tensão por velocidade mecânica no eixo do gerador WEG WMagnet.

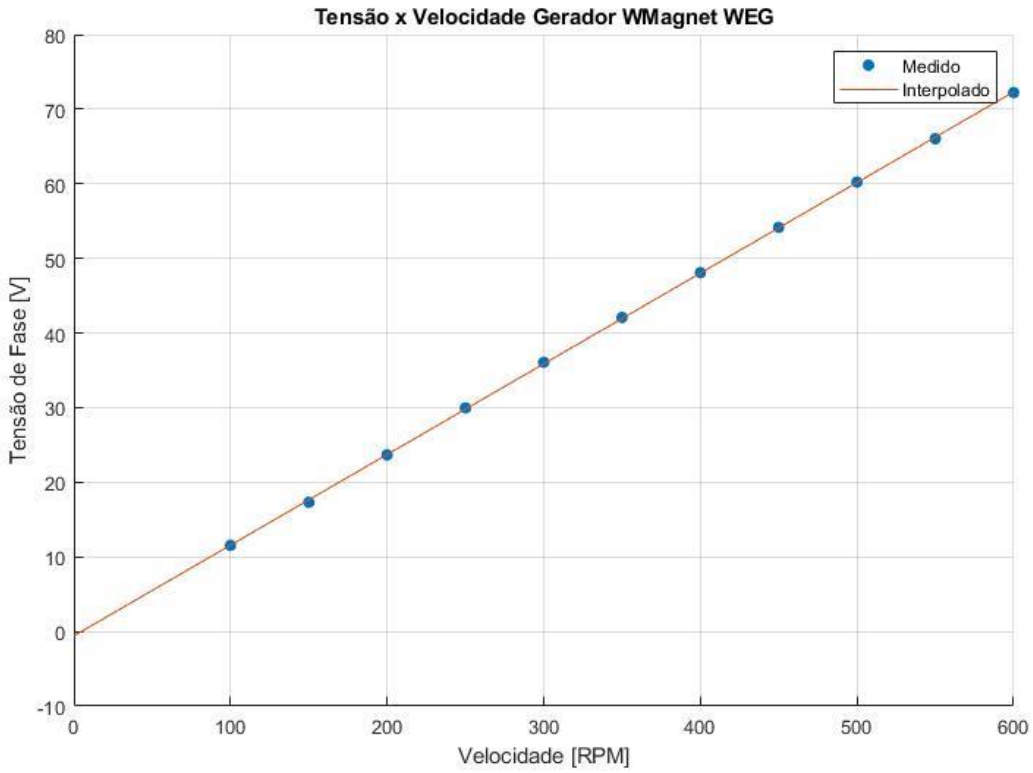
Velocidade (RPM)	**Tensão (linha)**	**Tensão (fase)**
100	20	11.55
150	30	17.32
200	41	23.67
250	51.9	29.96
300	62.5	36.08
350	72.9	42.09
400	83.3	48.09
450	93.8	54.16
500	104.3	60.22
550	114.4	66.05
600	125.1	72.23

Fonte: Autor.

Com base nos dados da Tabela 1, utilizando-se o *software Matlab* foi realizado uma interpolação numérica para encontrar a função de primeiro grau que melhor se adequasse aos dados relacionais entre a velocidade mecânica e a tensão de fase observada nos terminais da máquina. A constante de tensão encontrada e o gráfico da função interpolada pelos pontos medidos seguem abaixo.

$$K = 0.121516486\ [V/RPM] \tag{4.8}$$

Figura 22. Gráfico das medições e da curva interpolada da tensão de fase no estator pela velocidade mecânica no eixo do gerador.

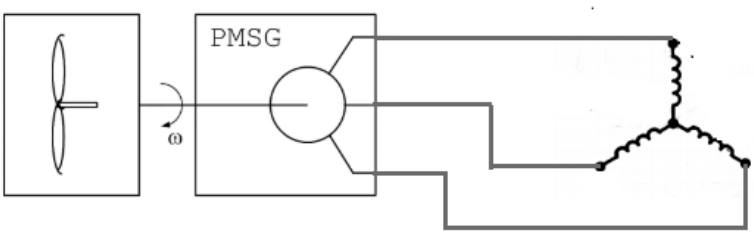


Fonte: Autor.

4.6 Carga resistiva trifásica e ponto de operação do sistema

Como citado anteriormente, devido a indisponibilidade do conversor CA/CC e do banco de baterias que compunham o sistema de despacho eólico da microrrede outrora, foi utilizado como carga para a realização dos testes da bancada de emulação eólica um banco resistivo trifásico conectado diretamente aos terminais do gerador, conforme a Figura 23.

Figura 23. Diagrama de blocos do sistema a ser emulado.



Fonte: Adaptação de TELLES, 2018.

Nesta topologia onde o gerador síncrono de ímãs permanentes é acionado com uma velocidade mecânica angular constante alimentando uma carga resistiva trifásica fixa, a constante de tensão dada em *(4.8)* pode ser utilizada para relacionar a velocidade mecânica angular com a potência elétrica consumida pela carga, uma vez que a variação de tensão possui um comportamento linear com a carga, logo pela lei de Ohm:

$$P = \frac{V^2}{R} = \frac{(K \times \omega)^2}{R} \qquad (4.9)$$

Onde:

$P = Potência\ [W]$;

$R = Resistência\ [\Omega]$;

$K = Constante\ de\ tensão\ [Volts/RPM]$;

$\omega = Velocidade\ mecânica\ angular\ [RPM]$.

O banco de resistores utilizado então foi dimensionado conforme as características da turbina emulada e da aplicação. Foi considerado que no sistema de emulação as velocidades de vento disponíveis para escolha manual variam de 3 a 12 metros por segundo. Neste caso, a excursão máxima de velocidade mecânica angular se encontra faixa de 0 à 600 RPM, como pode ser observado nas curvas de potência mecânica disponível na turbina eólica pela velocidade mecânica angular e pela velocidade de vento explicitadas na Figura 25, cujo calculo baseia-se nas equações *(2.8)*, *(2.10)*, *(2.11)* e *(2.12)*.

A carga resistiva trifásica que compõe o sistema foi então dimensionada conforme a capacidade de dissipação de potência máxima dos resistores disponíveis.

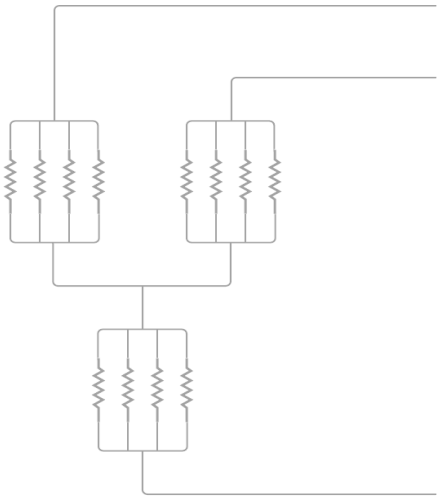
Foram utilizados doze resistores de 15 Ohms com capacidade máxima de dissipação potência de 600 Watts cada. Foram utilizados quatro resistores em paralelo

por fase, na topologia de carga trifásica em estrela. Através da equação *(4.10)* calcula-se a resistência equivalente por fase. A topologia da carga resistiva trifásica pode ser observada na Figura 24.

$$\frac{1}{R_{equivalente/fase}} = \frac{1}{R} + \frac{1}{R} + \frac{1}{R} + \frac{1}{R} \qquad (4.10)$$

$$R_{equivalente/fase} = 3.75\ [\Omega] \qquad (4.11)$$

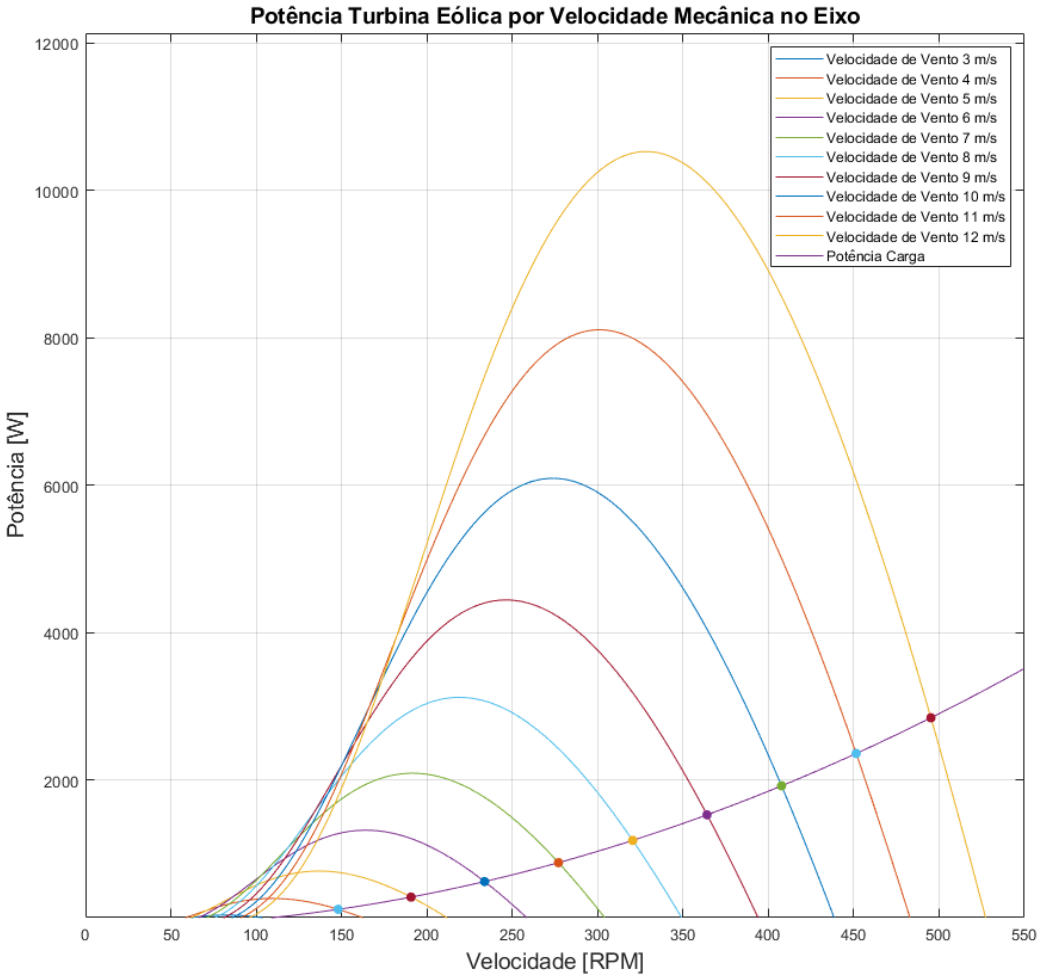
Figura 24 Configuração de carga trifásica em estrela utilizada na emulação eólica desenvolvida



Fonte: Autor.

Utilizando a equação *(4.9)* da potência dissipada pelos resistores pela velocidade mecânica de acionamento do gerador e as equações *(2.8)*, *(2.10)*, *(2.11)* e *(2.12)* que descrevem a potência mecânica disponível na turbina eólica pela velocidade mecânica no eixo, foram plotadas as curvas da Figura 25 no *software Matlab*.

Figura 25. Potência da turbina eólica e da carga conectada pela velocidade mecânica no eixo



Fonte: Autor.

Os pontos de intersecção demarcados na Figura 25 entre as curvas de potência da turbina eólica pelas velocidades de vento e a curva de potência dissipada na carga resistiva trifásica pela velocidade mecânica de acionamento são os pontos aproximados de operação do sistema emulação eólica. Os pontos de operação seguem na Tabela 2. Na Figura 27 podemos observar melhor as curvas de potência mecânica no eixo da turbina eólica para as velocidades de vento menores.

Tabela 2. Pontos de operação de velocidade mecânica e potência calculados para diferentes velocidades de vento.

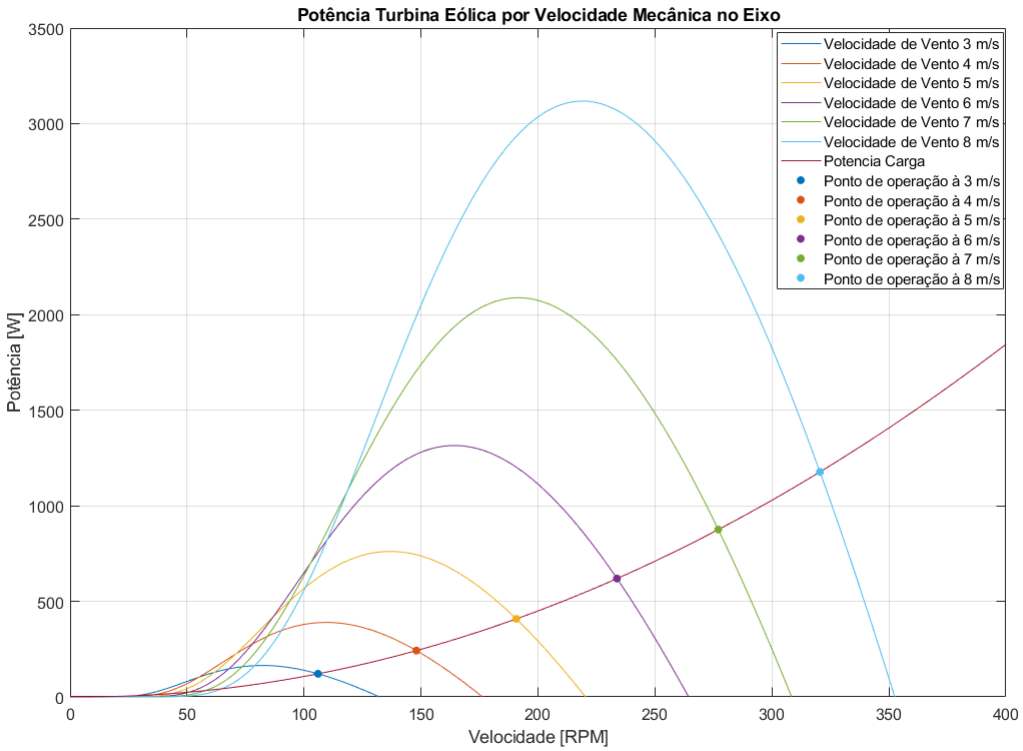
Pontos de Operação		
Velocidade de Vento [m/s]	Velocidade Mecânica [RPM]	Potência [W]
3	105.96	120.88
4	148.02	242.30
5	190.71	408.26
6	233.76	619.26
7	277.05	875.57
8	320.50	1177.36
9	364.07	1524.74
10	407.72	1917.77
11	451.44	2356.50
12	495.21	2840.96

Fonte: Autor.

Figura 26. Foto da carga resistiva trifásica em estrela utilizada na implementação da bancada de emulação eólica.

Fonte: Autor.

Figura 27. Potência da turbina eólica e da carga conectada pela velocidade mecânica no eixo para as velocidades de vento testadas na bancada.



Fonte: Autor.

4.7 Simulink Real Time e aplicação .NET

A solução criada para controlar a emulação da turbina eólica baseia-se na integração da simulação matemática desenvolvida para modelagem da turbina eólica através da *toolbox Simulink Real Time*, que faz parte da coleção de ferramentas do *software Matlab*-Simulink, e da aplicação criada através da plataforma .NET com a linguagem de programação C# e a IDE (*Integrated Development Environment*) Microsoft Visual Studio 2017 que tem o propósito de permitir ao usuário controlar a emulação através de uma interface gráfica.

A *target machine* é a máquina responsável por executar o modelo de simulação da turbina eólica. É nela que os dados da emulação são processados através do modelo matemático da turbina para gerar uma velocidade de referência que acionará o motor de emulação. Para que a máquina possa ser utilizada como *target machine,* ela deve conter uma placa de rede compatível com os *drivers* disponíveis para uso pelo *Simulink Real Time*, além de possuir um sistema operacional *DOS (Disk*

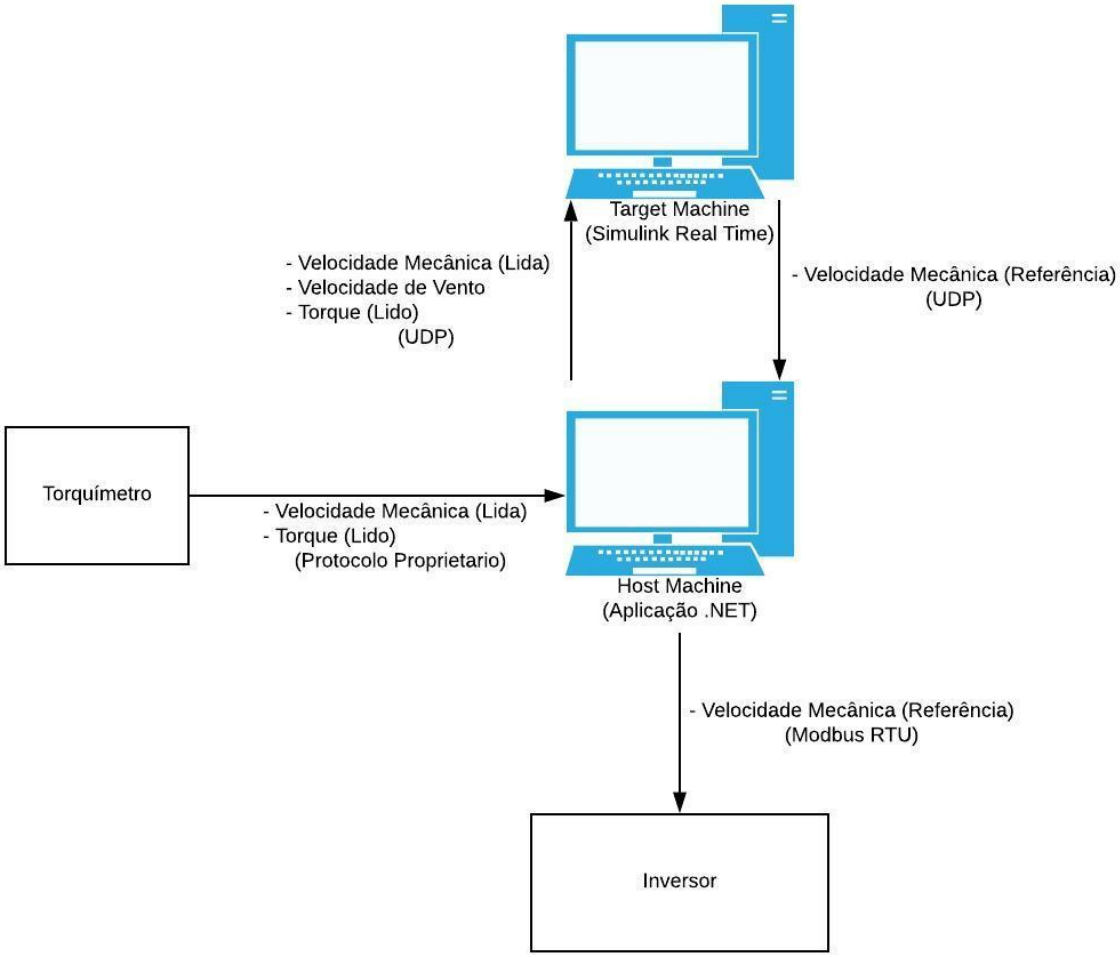
Operating System) para inicialização da máquina e carregamento do *Kernel* gerado pelo *Simulink Real Time*. Os passos para a criação do *Kernel* e para a utilização de *boot* por *pendrive* para carregamento do *Kernel* são discutidos no apêndice A.

A *host machine* é a máquina utilizada para controlar o sistema de emulação. Nela é implementada a aplicação .NET que realiza o controle da simulação que roda na *target machine* bem como o gerenciamento de envio e recebimento de dados do transdutor de torque, do inversor de acionamento e da própria *target machine*.

A comunicação entre os componentes do sistema de emulação (inversor de acionamento, transdutor de torque, *target machine* e *host machine)* ocorre exclusivamente através da aplicação .NET implementada na *host machine.* Embora o *Simulink Real Time* possua blocos para tratamento de comunicação serial, na versão do *Matlab* utilizada para a elaboração desta aplicação (*Matlab* 2018b) não há disponibilidade de blocos que utilizem o protocolo *Modbus,* tampouco há formas simples de tratamento de dados para utilização do protocolo proprietário que integra o sistema de comunicação do transdutor de torque Interface utilizado neste projeto. A *target machine* comunica-se apenas com a *host machine* utilizando comunicação UDP*,* recebendo os dados de torque e velocidade mecânica lidos pelo transdutor de torque, enviando os dados de velocidade mecânica de referência e recebendo os dados de velocidade de vento.

O diagrama de blocos do fluxo de comunicação do sistema de emulação pode ser observado na Figura 28. Neste diagrama estão explicitados os dados que são enviados e recebidos pelos componentes do sistema de emulação, bem como os protocolos que são utilizados nas comunicações entre os componentes.

Figura 28. Fluxograma de comunicação do sistema de emulação em tempo real.

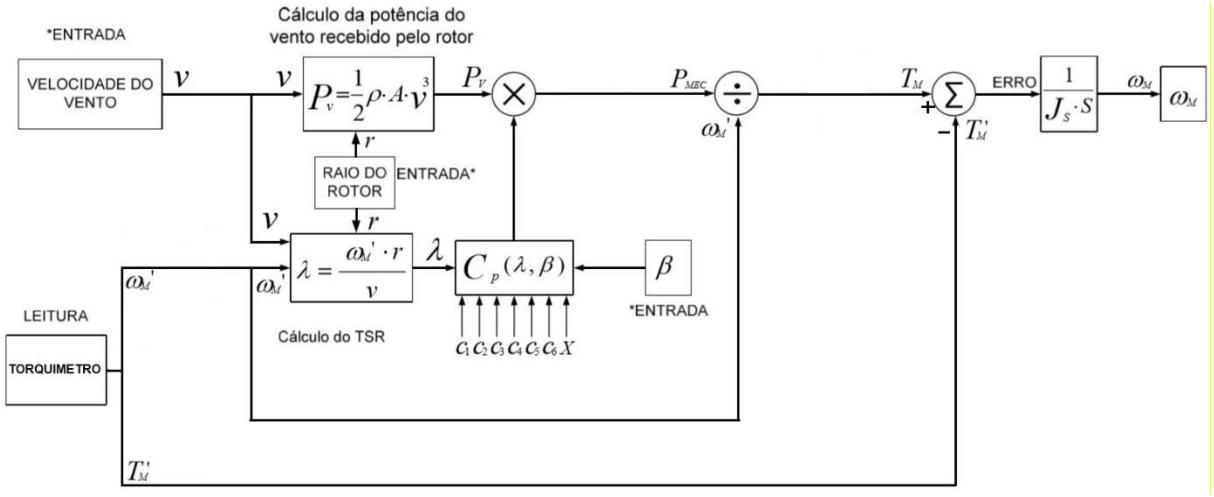


Fonte: Autor.

4.7.1 *Target machine* (*Simulink Real Time*)

O modelo matemático implementado na *target machine* para simulação da turbina eólica baseia-se nas equações *(2.8)* e *(2.10)* que calculam a potência extraída do vento e gerada pela turbina eólica com base na velocidade de vento escolhida pelo usuário da aplicação (ou carregada através do perfil de vento inserido na aplicação), no raio da turbina (que nesta aplicação é 2.775 metros), no ângulo de *pitch* das pás da turbina (que nesta aplicação é 0º) e nos dados do Quadro 1 (HEIER, 2006) referentes aos parâmetros empíricos para o cálculo do coeficiente de potência C_p da turbina eólica. A malha de controle do sistema de emulação pode ser observada na Figura 29.

Figura 29. Diagrama de blocos da malha de controle da emulação eólica desenvolvida.



Fonte: Adaptado de MUSSA et al, 2010.

Como é necessário ter a velocidade mecânica no eixo da turbina para calcular a potência extraída do vento pela turbina eólica, a velocidade é lida pelo torquímetro e enviada à *target machine* para realizar o cálculo da potência.

Obtendo-se a potência mecânica de referência calculada para a turbina eólica modelada, obtém-se o torque de referência dividindo a potência mecânica pela velocidade mecânica lida pelo transdutor de torque. Desse torque de referência é então subtraído o torque real lido pelo transdutor, gerando um sinal de erro que é utilizado como sinal de controle da malha, segundo as equações *(4.12)* e *(4.13)*.

$$T_M = \frac{P_{MEC}}{\omega'_M} \qquad (4.12)$$

$$erro = T_M - T'_M = \Delta T \qquad (4.13)$$

Obtendo-se o erro de torque da malha de controle, cujo representa uma variação de torque, calcula-se através da equação *(4.15)* a nova velocidade de referência que será enviada ao inversor de acionamento para controle do motor. O momento de inércia $J_{aerogerador}$ apesar de complicado de ser calculado analiticamente para turbinas eólicas de velocidade variável, pode ser aproximado pela equação *(4.16)* (MORREN; PIERIK, 2006). Considerando um peso total aproximado de 50 kg para as pás, o que é condizente com as turbinas deste porte disponíveis no mercado, temos o momento de inercia dado em *(4.17)*.

$$T = J_{aerogerador} \frac{d\omega}{dt} \tag{4.14}$$

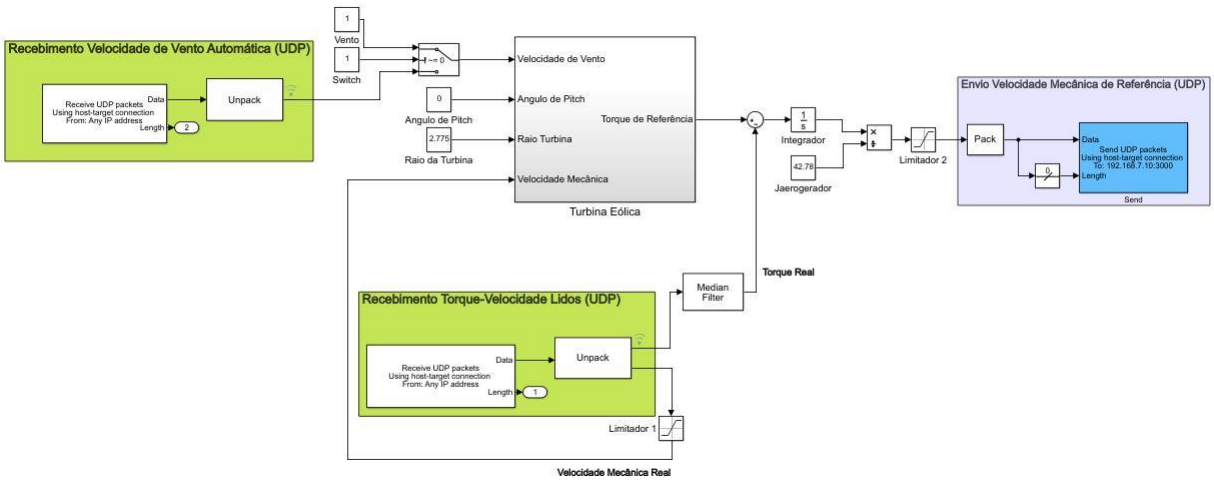
$$\omega = \frac{1}{J_{aerogerador}} \int \Delta T \tag{4.15}$$

$$J_{aerogerador} = \frac{M_{pá}.r^2}{9} \tag{4.16}$$

$$J_{aerogerador} = \frac{50 \times 2.775^2}{9} = 42.78 kg.m^2 \tag{4.17}$$

O modelo de simulação da turbina eólica implementado através do *Simulink Real Time* e inserido na *target machine* pode ser observado na Figura 30.

Figura 30. Diagrama de blocos da modelagem matemática da turbina eólica e da malha de controle desenvolvida no *software Simulink.*



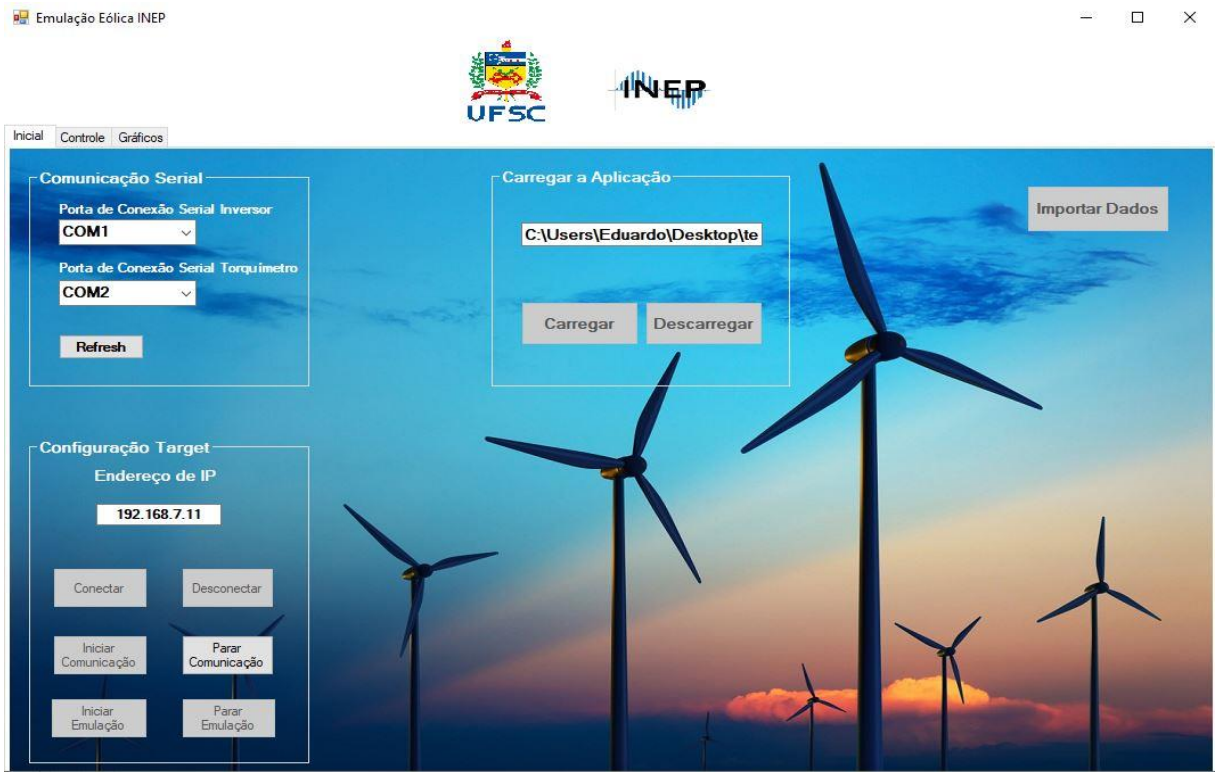
Fonte: Autor.

Após a implementação do modelo de blocos para a simulação da turbina eólica conforme a Figura 30, o *Simulink* ao construir o modelo realiza a criação de um arquivo .mldatx através de um compilador para linguagem C que permite que o modelo da turbina possa ser transferido para a *target machine.* O procedimento de criação do arquivo .mldatx é discutido no apêndice A.

4.7.2 *Host machine* (aplicação .NET)

A aplicação .NET foi desenvolvida para controlar todo o sistema de emulação. Ela utiliza uma API disponibilizada pela *Mathworks* que implementa através de bibliotecas para .NET Frameworks uma série de classes e funções que permitem a criação de aplicações *stand-alone* para a comunicação e controle dos modelos de simulação em tempo real implementados na *target machine* com a *toolbox Simulink Real Time*. Além do controle sobre a aplicação que roda na *target machine*, como a alteração de parâmetros à exemplo da alteração manual da velocidade de vento de referência, a aplicação .NET foi desenvolvida para realizar a comunicação com todos os componentes do sistema de emulação. É através da aplicação .NET que é realizada a comunicação serial com o inversor CFW-11 utilizando o protocolo *Modbus-RTU,* implementado através da biblioteca *EasyModbus* disponível gratuitamente para implementação do protocolo em desenvolvimento de aplicações .NET. Também na aplicação .NET que é realizado a comunicação e tratamento dos dados enviados e recebidos pelo transdutor de torque, seguindo o protocolo proprietário discutido na seção 4.2.2. Já a comunicação com a *target machine* para alteração de parâmetros e envio/recebimento de dados como velocidade mecânica de referência e lida, torque lido e velocidade de vento automática é realizada através de *sockets UDP,* discutidos na seção 3.3.2.

Figura 31. Tela inicial da aplicação desenvolvida em .NET para controle da emulação.



Fonte: Autor.

Na aba *Inicial* da aplicação desenvolvida (Figura 31) situam-se todos os botões e configurações necessárias para realizar a operação da aplicação e comunicação entre componentes. É nesta aba onde são definidas as respectivas portas seriais onde estão conectados o inversor CFW-11 e o transdutor de torque *Interface*. Tal informação pode ser obtida através da verificação na ferramenta *Gerenciador de Dispositivos* do *Windows* após os componentes terem sido conectados. Além das portas seriais, também é definido nesta aba o endereço IP ao qual a *host machine* irá se conectar à *target machine,* que depende da configuração na criação do *Kernel* pelo *Simulink Real Time*, discutido no apêndice A.

Ainda há seis botões que controlam o início da emulação, são eles:

- Conectar/desconectar: após a inicialização da *target machine*, são os botões responsáveis por conectar/desconectar a *host machine* à *target machine.*
- Iniciar/Parar emulação: após o carregamento da aplicação com o modelo da turbina eólica que será citado posteriormente, é responsável por iniciar os osciloscópios virtuais e a simulação na *target machine*, bem como limpar todos

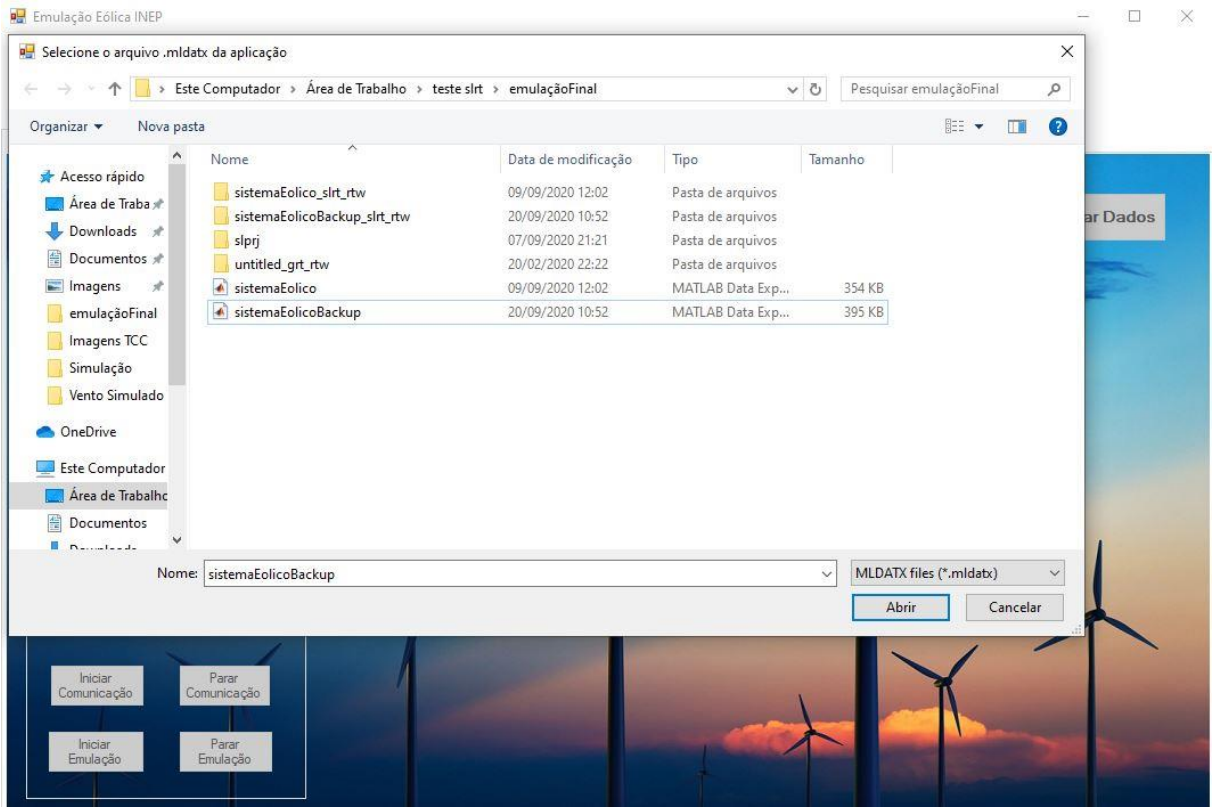
os *buffers* de comunicação da aplicação e iniciar e requisitar o início de aquisição de dados pelo transdutor de torque.

- Iniciar/Parar comunicação: responsável por iniciar a rotina de comunicação entre os componentes do sistema de emulação, descrita na seção 4.7.2.1.

A aplicação .NET foi desenvolvida para permitir que a bancada de emulação eólica rode o modelo de emulação de forma totalmente *stand-alone*, isto é, sem necessitar do *software Matlab* para realizar a comunicação entre a *host machine* e a *target machine*, tanto para inicialização e controle da aplicação quanto para a visualização dos dados da emulação em tempo real. Para isso, após a *target machine* ter sido inicializada com o *kernel* gerado pelo *Simulink Real Time* como discutido no apêndice A e a *host machine* estar conectada à *target machine*, o arquivo .mldatx gerado na compilação do modelo *Simulink* implementado é carregado pelo usuário para a *target machine* utilizando a *group box Carregar a Aplicação* (Figura 32).

Após a emulação ter sido executada e finalizada, o botão *Importar Dados* permite que o usuário possa transferir os dados da última execução de emulação que são armazenados em arquivos .DAT na *target machine* para qualquer pasta na *host machine*. Para a leitura dos arquivos de dados gravados é necessária a utilização do *software Matlab*, já que se trata de um tipo específico de dados deste *software*. A rotina para leitura dos dados pode ser verificada no apêndice B.

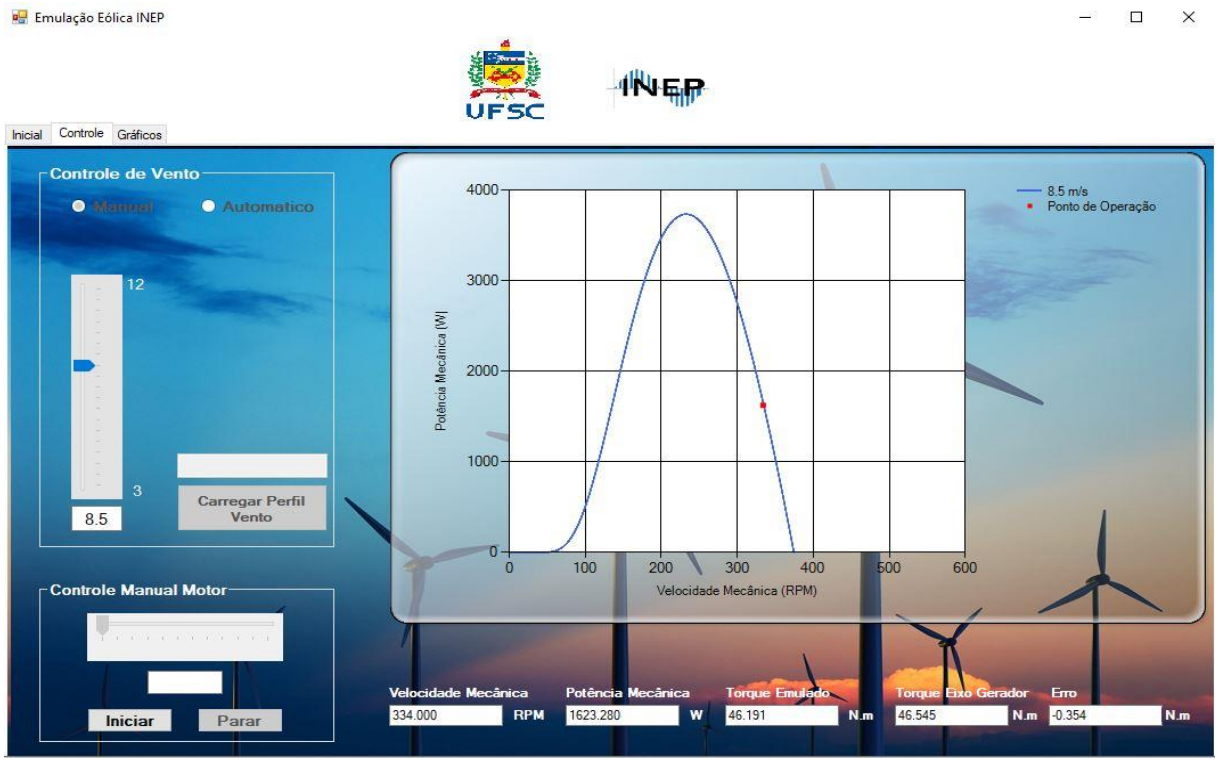
Figura 32. Carregamento do modelo desenvolvido em Simulink na *target machine* através da aplicação .NET.



Fonte: Autor.

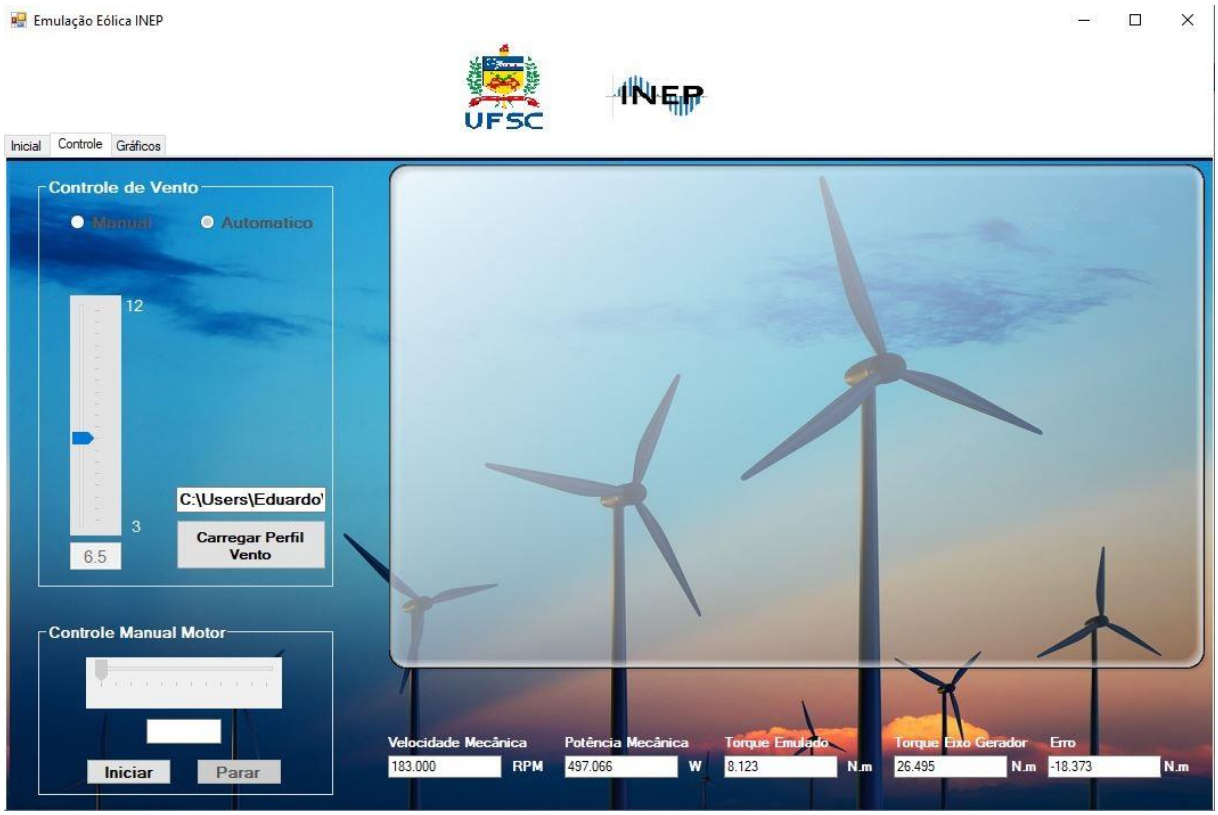
A aba *Controle* da aplicação desenvolvida (Figura 32) é responsável por controlar a velocidade de vento e visualizar os dados da emulação em tempo real. Na aplicação foram implementadas duas possibilidades para o controle do vento de referência para a aplicação. Utilizando a opção de vento manual, o usuário do sistema pode selecionar uma das velocidades de vento disponíveis que variam de 3 a 12 metros por segundo em intervalos de 0.5 metro por segundo. Para cada velocidade de vento selecionada, a aplicação carrega a respectiva curva de potência por velocidade angular mecânica da turbina calculada previamente e carregada antes da execução do programa, bem como indica o atual ponto de operação onde localiza-se o sistema no momento. Já optando-se pela velocidade de vento automática, o usuário tem a possibilidade de carregar na aplicação um arquivo .txt com velocidades de vento geradas por algum perfil de vento que serão enviadas da *host machine* para a *target machine* com o mesmo período da execução da rotina de comunicação da aplicação (aproximadamente 250 milissegundos).

Figura 33. Aba de controle da aplicação .NET operando em modo manual de velocidade de vento.



Fonte: Autor.

Figura 34. Aba de controle da aplicação .NET operando em modo automático de velocidade de vento.



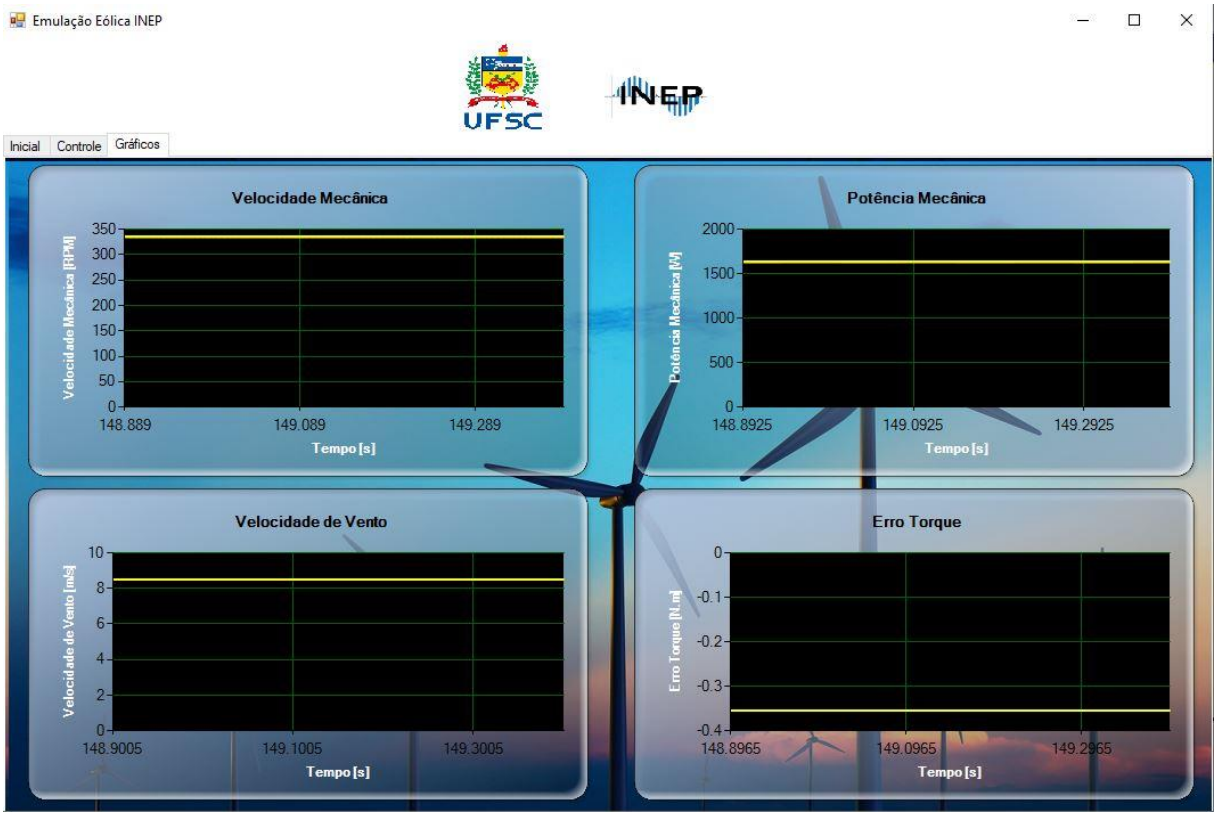
Fonte: Autor.

Nesta aba (Figura 33) também é possível conferir os valores da simulação em tempo real. São indicados os valores de velocidade mecânica no eixo do gerador, potência mecânica, torque emulado (torque de referência calculado pelo modelo matemático da turbina), torque no eixo do gerador (torque real lido pelo transdutor de toque) e erro (diferença de entre o torque emulado e o torque lido no eixo do gerador).

Por fim, nesta aba (Figura 33) também é possível controlar o acionamento do inversor/motor de forma manual, selecionando velocidades que vão de 50 a 600 RPM em intervalos de 50 RPM. Esta funcionalidade tem o objetivo de permitir realizar testes e *debugs* com o sistema de acionamento.

Na aba *Gráficos* é possível visualizar de forma gráfica os sinais de velocidade mecânica, potência mecânica, velocidade de vento e o erro de torque.

Figura 35. Aba de gráficos da aplicação .NET desenvolvida para controle da emulação eólica.



Fonte: Autor.

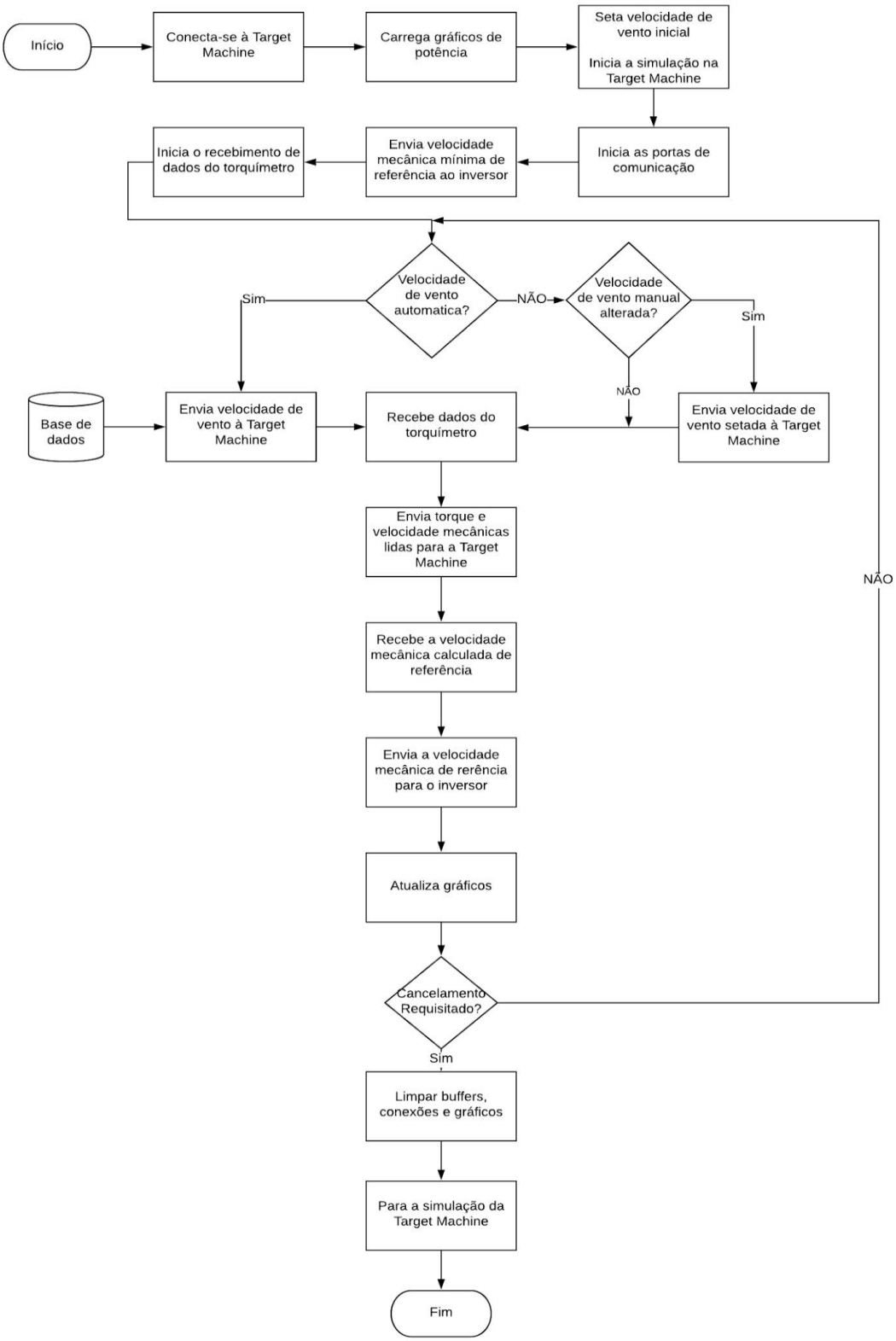
4.7.2.1 Rotina de comunicação e execução

A aplicação .NET desenvolvida possui duas *threads* de execução que rodam paralelamente durante a operação da aplicação. A primeira *thread* é responsável por permitir ao usuário a operação e controle do sistema de emulação através da interface gráfica, permitindo iniciar e parar a emulação, alterar parâmetros do modelo como a velocidade de vento e visualizar os sinais nos osciloscópios virtuais da aplicação. A segunda *thread* é responsável por realizar a comunicação do sistema de emulação. Trata-se se uma rotina que realiza a aquisição dos dados de torque e velocidade mecânica oriundas do transdutor de torque e trata os dados conforme discutido na seção 4.2, envia estes dados à *target machine* através de *sockets UDP* e recebe a velocidade mecânica de referência para enviá-la através do protocolo *Modbus-RTU* ao inversor de acionamento.

A rotina de comunicação repete-se em um *loop* infinito até que a interrupção do sistema de emulação seja solicitada, extinguindo a *thread.* Ela possui um período de execução de aproximadamente 250 milissegundos, período que compreende a somatória de todos os tempos de execução do envio/recebimento das mensagens no sistema de emulação incluindo um acréscimo proposital de 200 milissegundos de pausa na execução da *thread* para que a taxa de atualização da velocidade de referência do inversor não cause uma sobrecarga na comunicação serial *Modbus.*

O fluxograma de execução da aplicação desenvolvida em .NET pode ser acompanhado na Figura 36.

Figura 36. Fluxograma de execução da aplicação de controle .NET.



Fonte: Autor.

5 RESULTADOS E DISCUSSÕES

Previamente à realização dos testes de bancada, foi implementado um modelo matemático de simulação do sistema completo com gerador e carga resistiva trifásica (Figura 37) dentro do próprio modelo da turbina eólica no *Simulink Real Time* com o intuito de validar as modelagens e controles desenvolvidos até então.

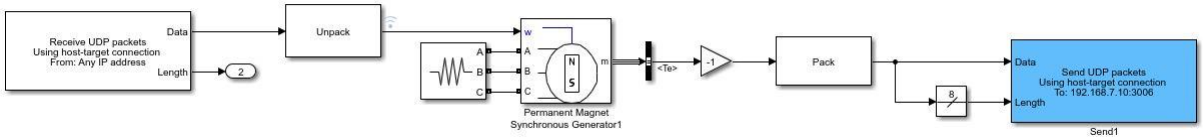
O modelo matemático utilizado para simulação do gerador foi o modelo do próprio *Simulink* para máquinas síncronas de ímãs permanentes. Os parâmetros da máquina utilizados na simulação foram extraídos do trabalho de NETO (2016), que caracteriza os parâmetros de uma máquina síncrona de ímãs permanentes semelhante, porém de potência nominal menor. Os dados da máquina utilizada na simulação encontram-se na Tabela 3.

Tabela 3. Parâmetros construtivos da máquina elétrica WEG WMagnet 11 kW.

Parâmetro	**Valor**
Potência Nominal	11 kW
Tensão Nominal	380 V
Velocidade Nominal	1800 RPM
Número de polos	6
Indutância de eixo direto	11,9 mH
Indutância de eixo em quadratura	23,1 mH
Resistência dos enrolamentos	0,46 Ω
Fluxo concatenado de pico dos ímãs	0,538 Wb

Fonte: Autor.

Figura 37. Diagrama de blocos da modelagem do gerador no *software Simulink*.



Fonte: Autor.

Foram realizados dois testes para a simulação e para a emulação: o primeiro teste consistiu em utilizar o controle de vento manual para alterar a velocidade de vento de 3 metros por segundo até 8 metros por segundo com variações de 1 metro por segundo e avaliar o comportamento do sistema. O segundo teste consistiu em utilizar o controle de vento automático, carregando na aplicação dados de velocidades

geradas pelo perfil de vento dado pela equação *(5.1)*, obtido no trabalho de COLLIER (2011).

$$v_{\omega}(t) = V_{\omega,avg} + \frac{V_{\omega,avg}}{10} sin(\omega_{\omega} t) + \frac{V_{\omega,avg}}{10} sin(3{,}5\omega_{\omega} t) + \frac{V_{\omega,avg}}{20} sin(12{,}35\omega_{\omega} t) + \frac{V_{\omega,avg}}{100} sin(35\omega_{\omega} t) \quad (5.1)$$

Onde:

$v_{\omega}(t) =$ Velocidade de vento (m/s);

$V_{\omega,avg} =$ Velocidade média (m/s).

Dado:

$$\omega_{omega} = \frac{2\pi}{T_{\omega}} \quad (5.2)$$

$T_{\omega} = 60s$;

$V_{\omega,avg} = 5\ m/s$.

Os dados adquiridos para o primeiro teste, com variação manual da velocidade de vento, foram a potência mecânica, velocidade mecânica no eixo do gerador, torques mecânicos (referência e real), potência mecânica e velocidade do vento de referência.

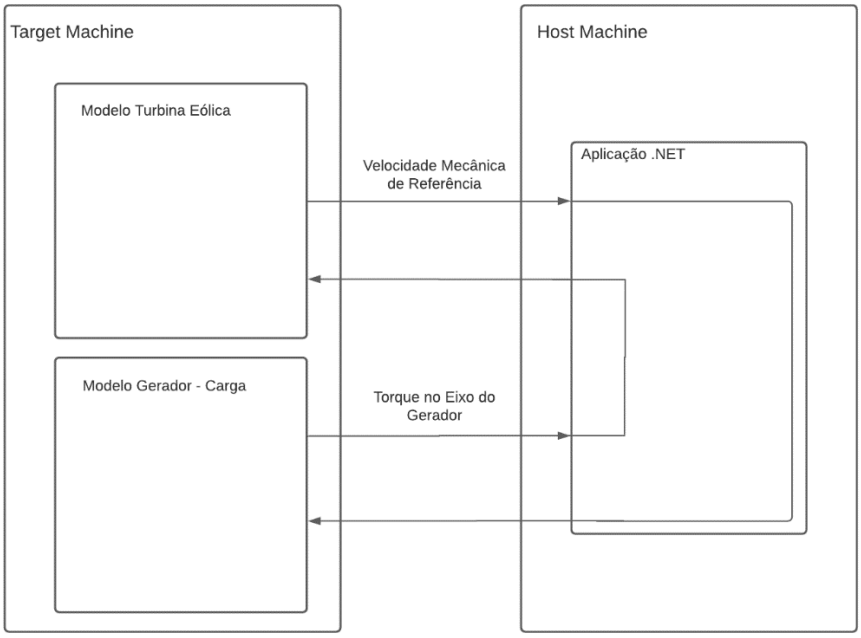
Para o segundo teste, com a variação automática da velocidade de vento baseada no perfil de vento gerado pela equação *(5.1)*, os dados adquiridos foram a velocidade de vento, potência mecânica, velocidade mecânica no eixo do gerador e torques mecânicos (referência e real).

Os dados coletados são oriundos dos arquivos .DAT armazenados na *target machine* para cada ciclo de emulação do sistema, e foram tratados e visualizados através do próprio *Matlab.*

5.1 Simulação

A simulação foi realizada através da simulação de todas as partes do sistema de emulação como discutido anteriormente. Na simulação não há interação com as partes físicas do sistema de emulação, como o inversor, motor, gerador, carga e o transdutor de torque. Ao invés disso, o modelo simulado de gerador-carga implementado juntamente no modelo da turbina eólica recebe os dados de velocidade mecânica de referência da *host machine* através de novos *sockets* de comunicação UDP e envia os dados de torque mecânico no eixo do gerador para a aplicação .NET na *host machine*, que recebe os dados de torque e os envia novamente para a *target machine,* porém endereçado ao *socket* referente ao recebimento dos dados de torque e velocidade da malha de controle de simulação da turbina eólica, como exemplificado na Figura 38.

Figura 38. Fluxograma de envio de dados da simulação entra *host machine* e *target machine.*

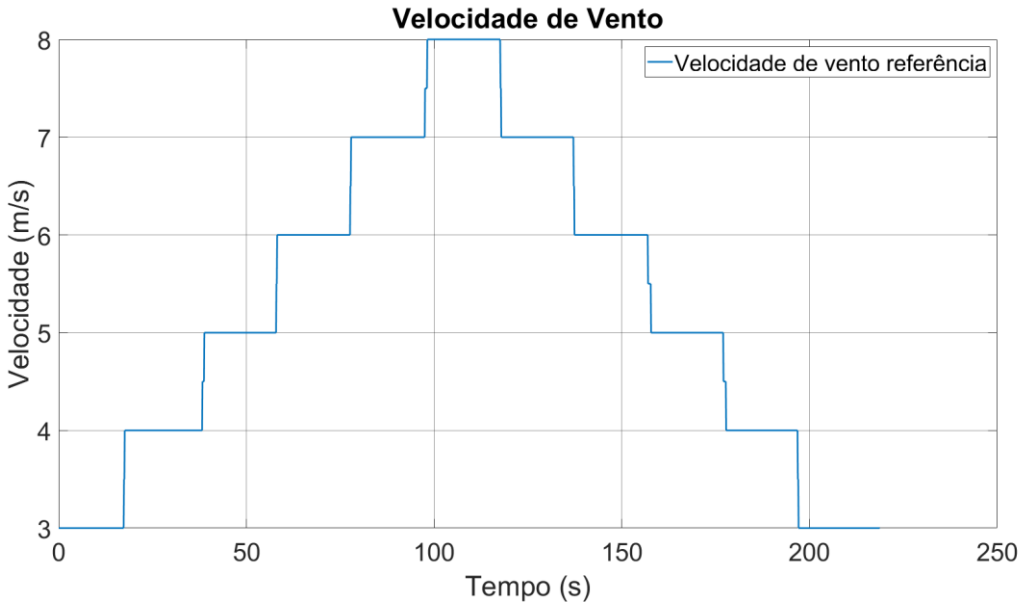


Fonte: Autor.

Velocidade de Vento Manual:

- Velocidade de Vento:

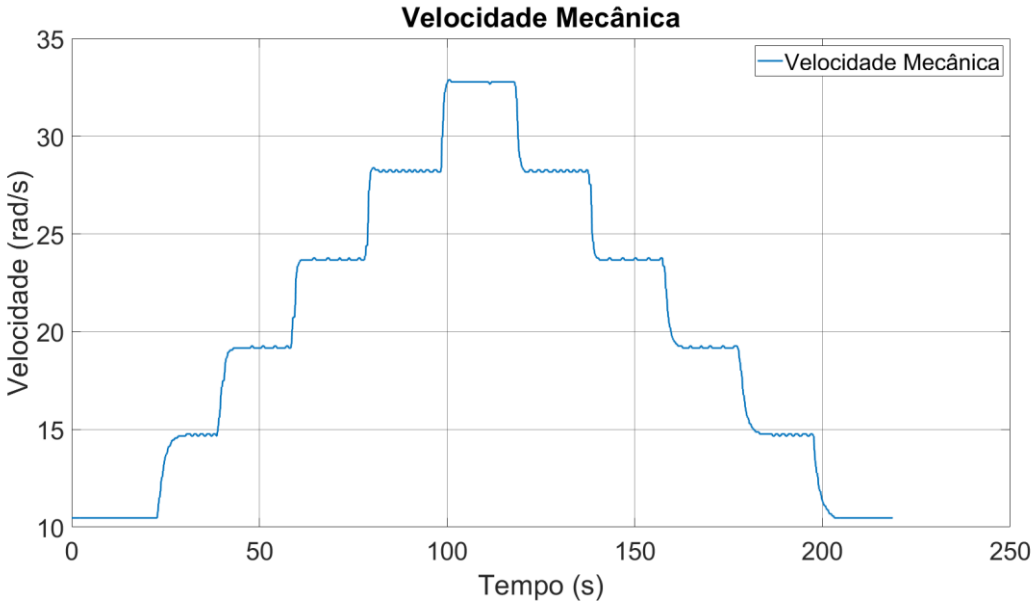
Figura 39. Velocidade de vento manual na simulação.



Fonte: Autor.

- Velocidade Mecânica no Eixo do Gerador

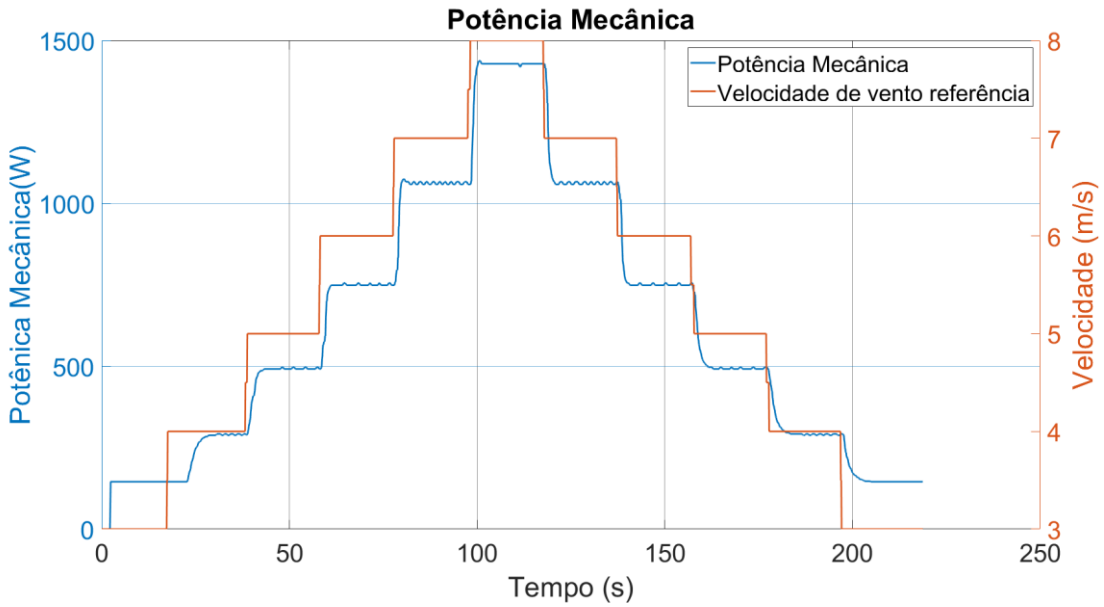
Figura 40. Velocidade mecânica no eixo do gerador na simulação.



Fonte: Autor.

- Potência Mecânica

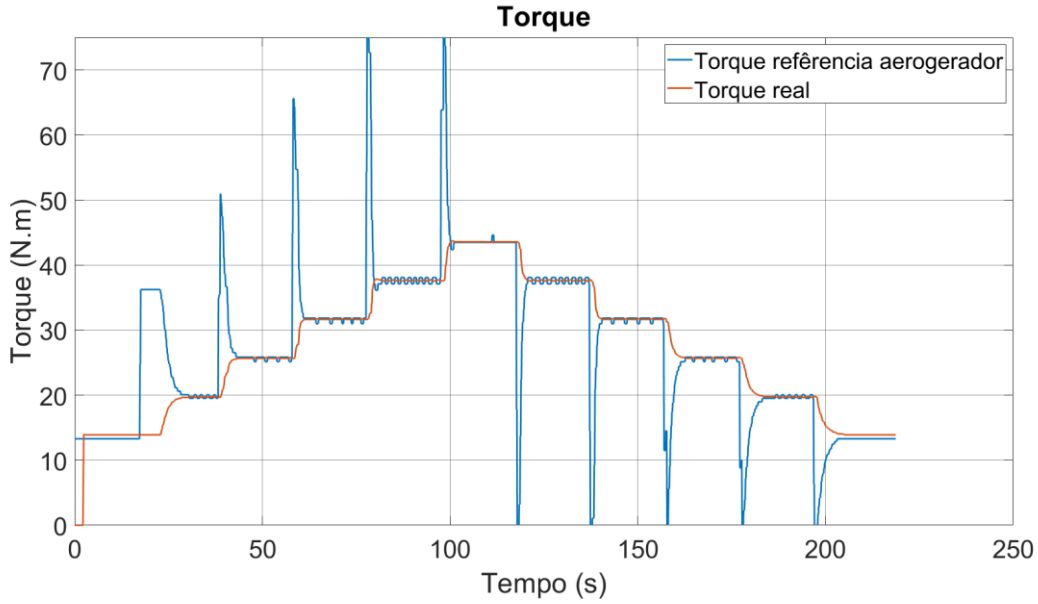
Figura 41. Potência mecânica e velocidade referencial de vento na simulação.



Fonte: Autor.

- Torques da simulação:

Figura 42. Torque de referência da turbina eólica e torque real medido pelo transdutor Interface na simulação.

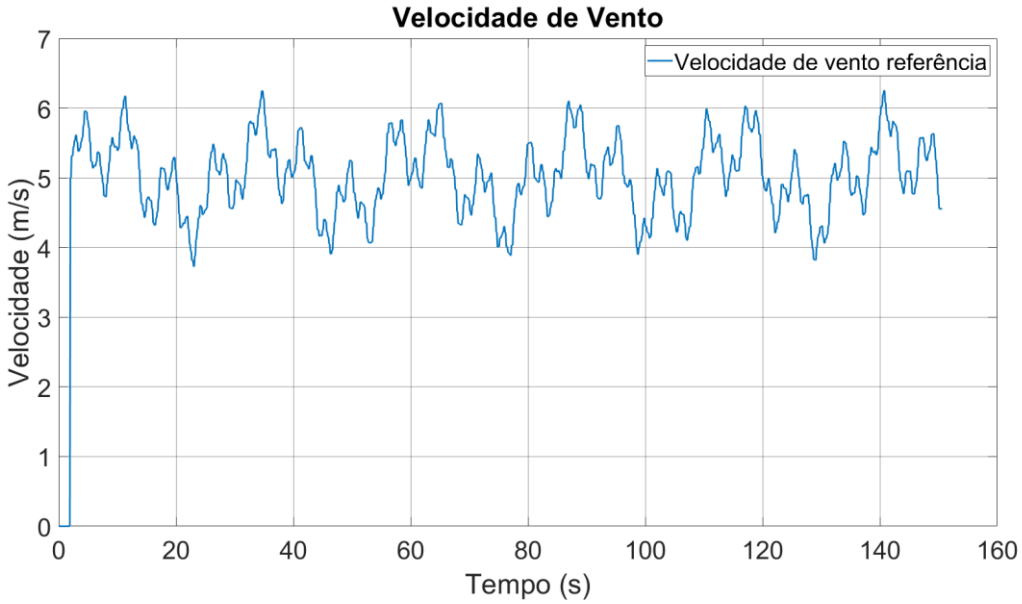


Fonte: Autor.

Perfil de Velocidade de Vento:

- Velocidade de Vento

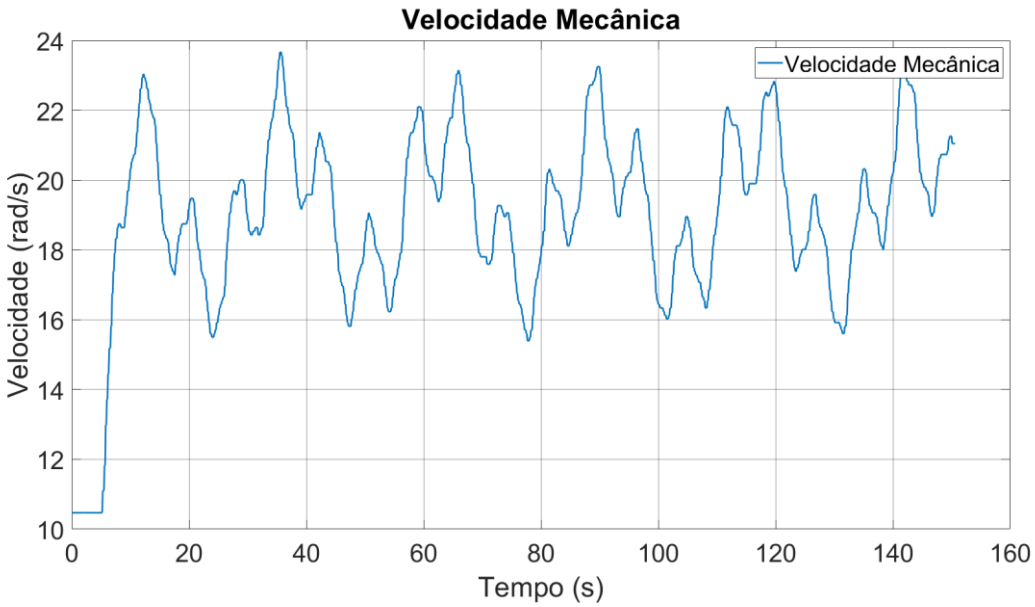
Figura 43. Perfil de velocidade de vento carregado na simulação.



Fonte: Autor.

- Velocidade Mecânica

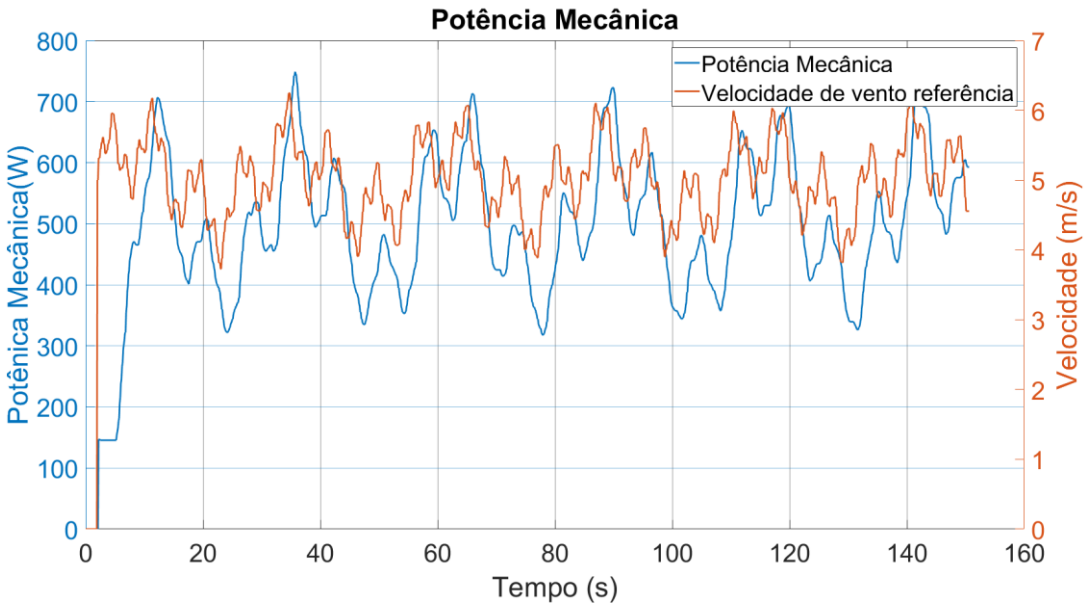
Figura 44. Velocidade mecânica no eixo do gerador com perfil de vento na simulação.



Fonte: Autor.

- Potência Mecânica:

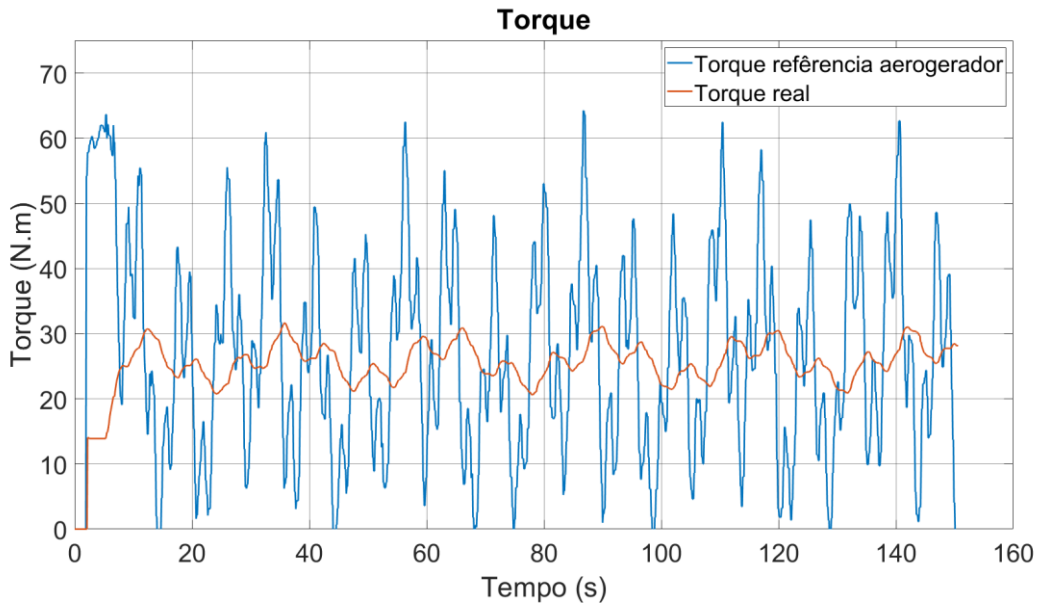
Figura 45. Potência mecânica da turbina eólica e velocidade de vento na simulação.



Fonte: Autor.

- Torques:

Figura 46. Torque de referência da turbina eólica e torque real no eixo do gerador da simulação.

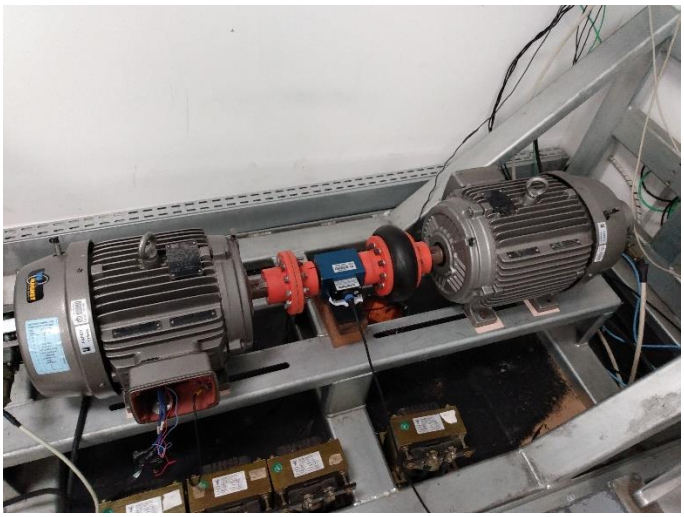


Fonte: Autor.

5.2 Emulação

A emulação foi realizada utilizando o sistema físico da bancada (Figura 47) com o acionamento inversor-motor-gerador-carga e com a aplicação final. Foram realizados os mesmos testes foram previamente realizados na simulação, sendo o primeiro teste com a mudança de velocidade de vento de forma manual e o segundo teste com a inserção de um perfil de vento de referência conforme a equação *(5.1)*.

Figura 47. Foto do sistema físico da bancada de emulação eólica.

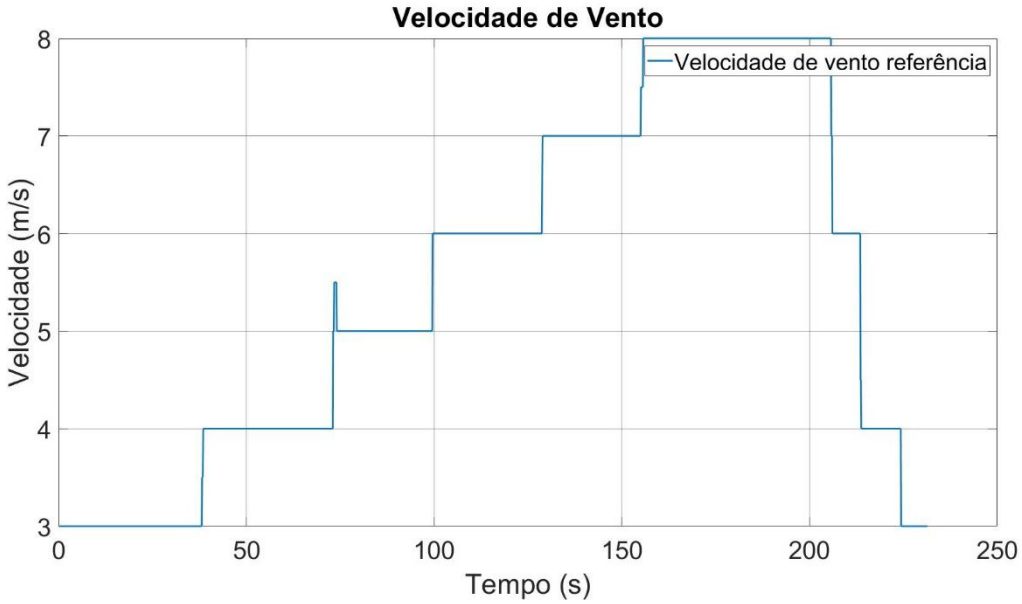


Fonte: Autor.

Velocidade de Vento Manual:

- Velocidade de Vento

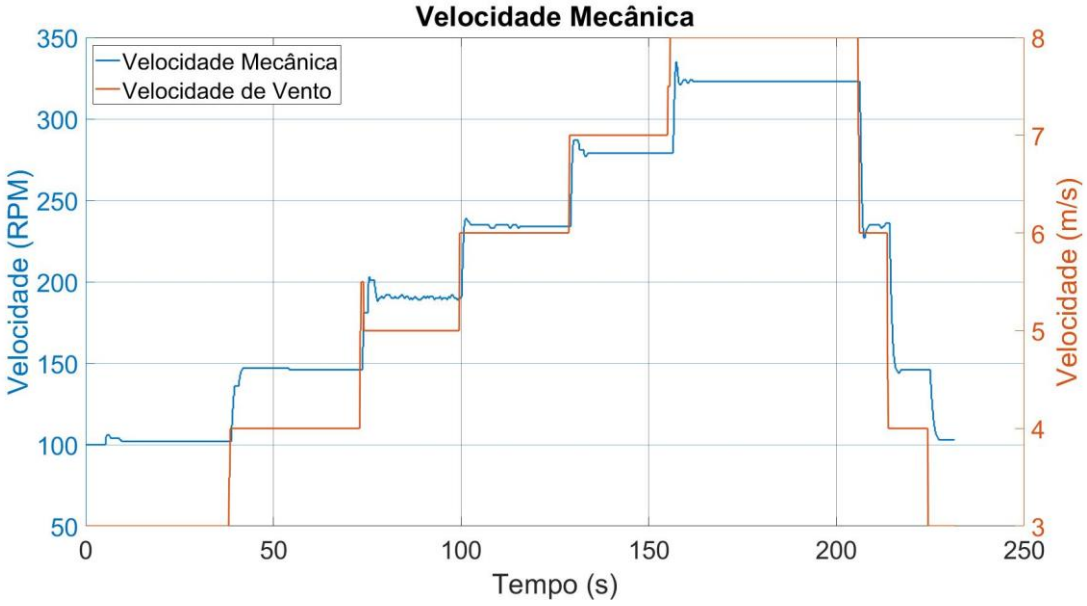
Figura 48. Velocidade de vento manual na emulação.



Fonte: Autor.

- Velocidade Mecânica

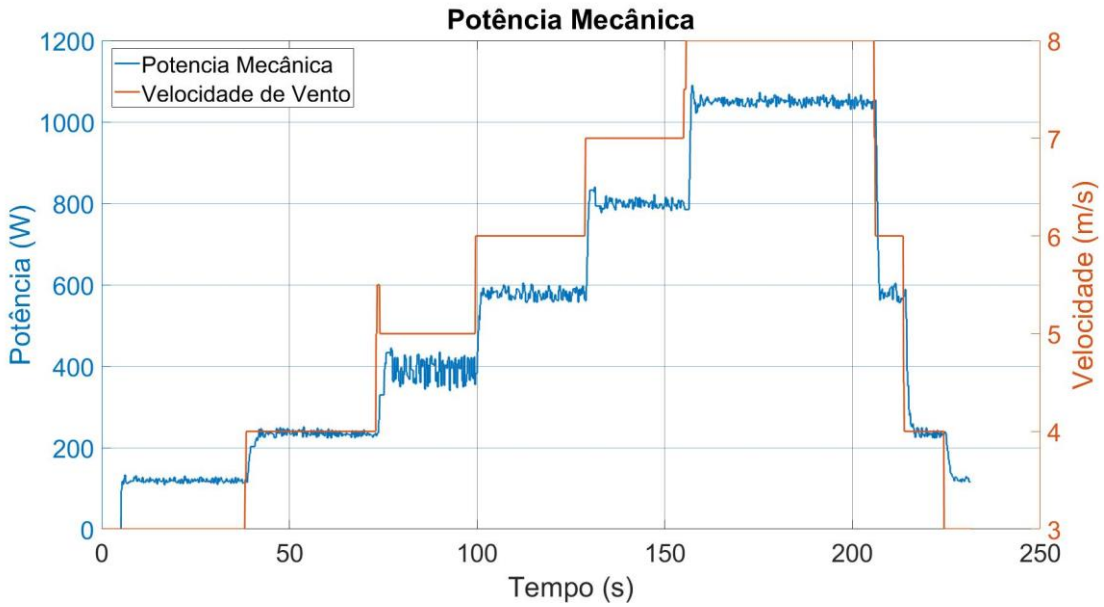
Figura 49. Velocidade mecânica durante emulação com velocidade de vento manual.



Fonte: Autor.

- Potência Mecânica

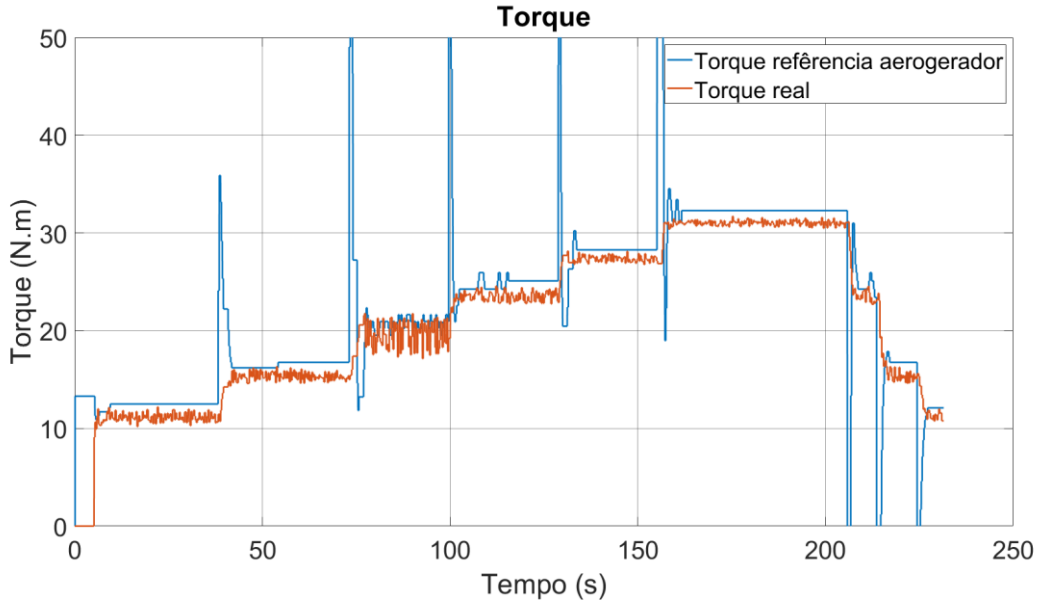
Figura 50. Potência mecânica e velocidade de vento durante emulação.



Fonte: Autor.

- Torques

Figura 51. Torque de referência da turbina eólica e torque mensurado pelo transdutor Interface durante emulação.

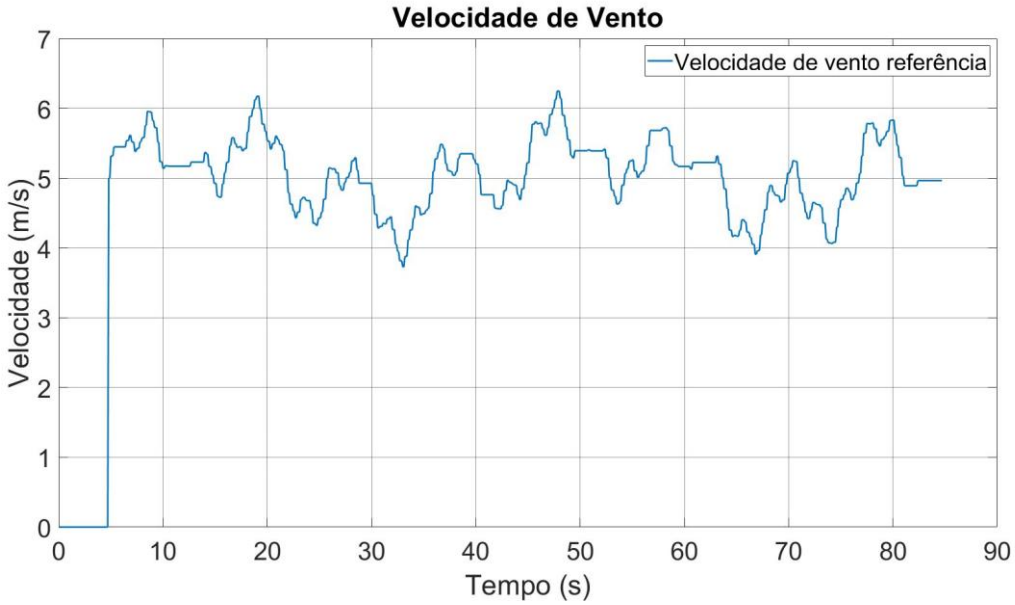


Fonte: Autor.

Perfil de Velocidade de Vento

- Velocidade de Vento:

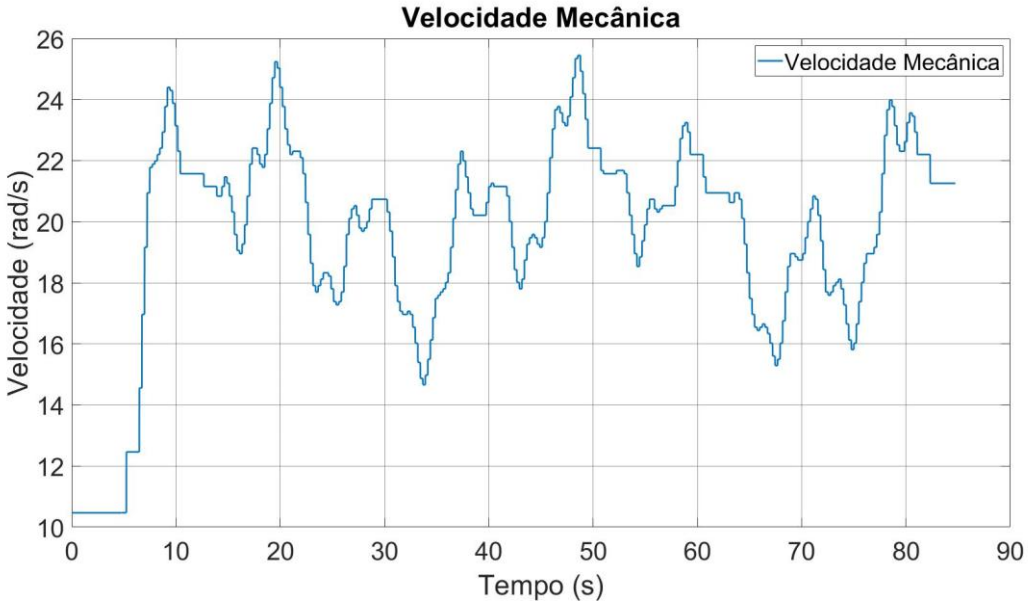
Figura 52. Perfil de velocidade de vento inserido na emulação.



Fonte: Autor.

- Velocidade Mecânica

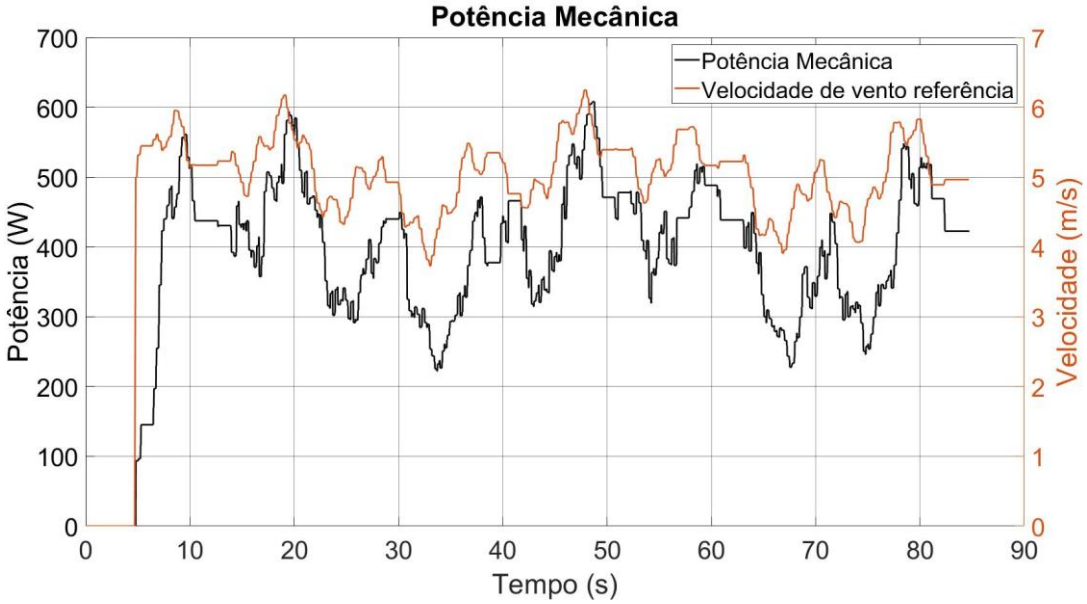
Figura 53. Velocidade mecânica no eixo do gerador na emulação.



Fonte: Autor.

- Potência Mecânica:

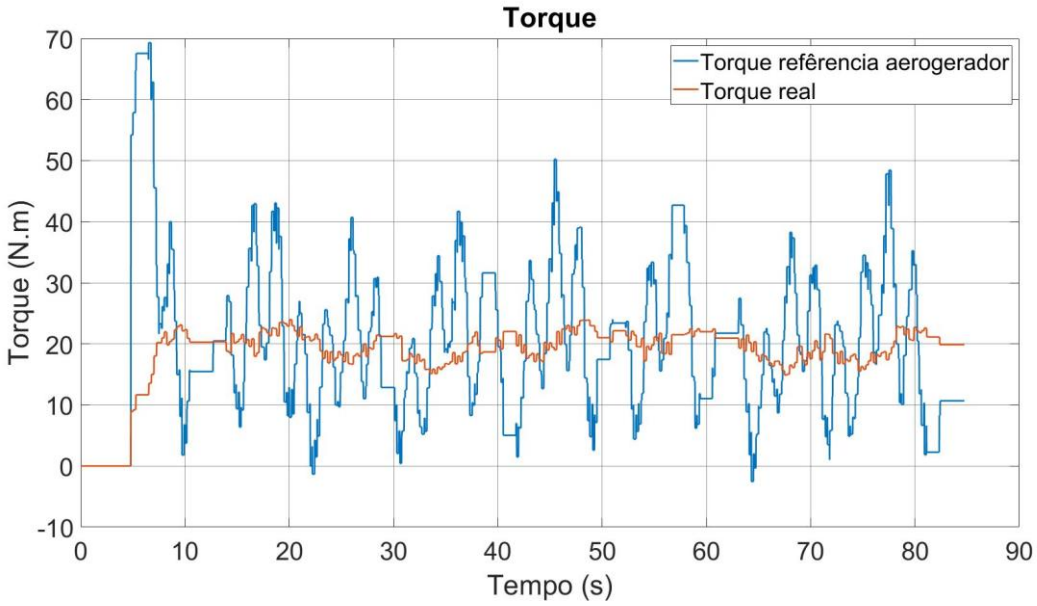
Figura 54. Potência mecânica e velocidade de vento de referência na emulação.



Fonte: Autor.

- Torques:

Figura 55. Torque de referência da turbina eólica e torque mensurado pelo transdutor Interface na emulação.

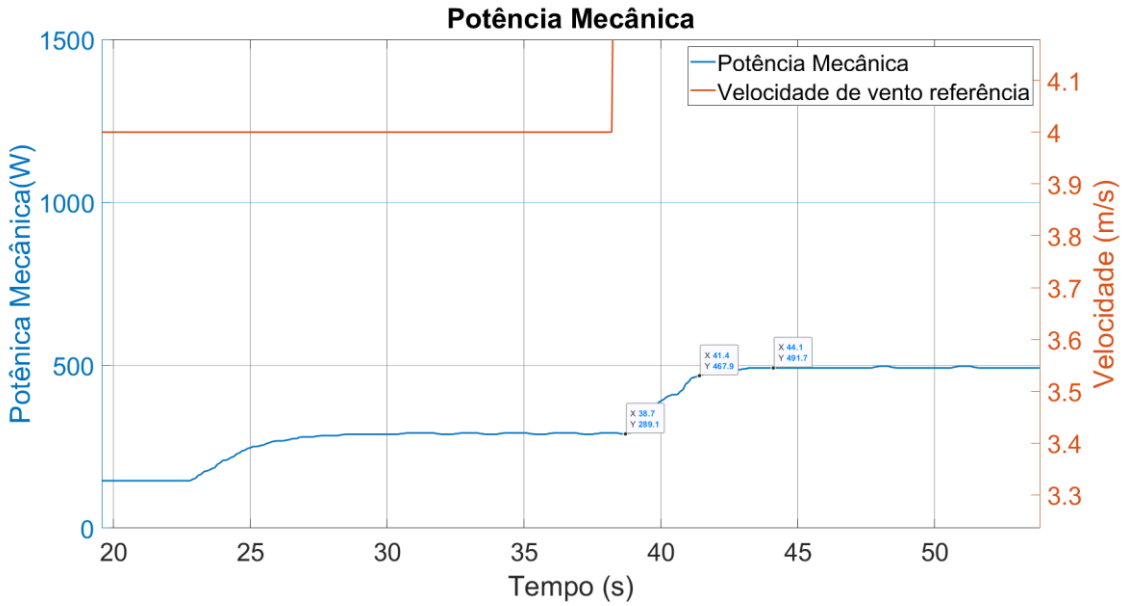


Fonte: Autor.

5.3 Discussões

Analisando os resultados obtidos nas simulações e emulações, percebe-se que o sistema de emulação eólica consegue cumprir com o seu objetivo de rastrear o comportamento dinâmico de uma turbina eólica à variação do vento, embora com algumas restrições. Foi possível verificar que de fato a modelagem da turbina eólica proporcionou uma boa resposta às variações de velocidade, com tempos de resposta à 5% máximos em operação próximos a 2.7 segundos (Figura 56).

Figura 56. Gráfico de potência mecânica durante emulação com tempos de resposta.



Fonte: Autor.

Tabela 4. Tempo de resposta para um degrau de 1 m/s.

Tempo de Resposta para 5 m/s		
T1 (s)	T2 (s)	$\Delta T\ (s)$
38.7	41.4	2.7

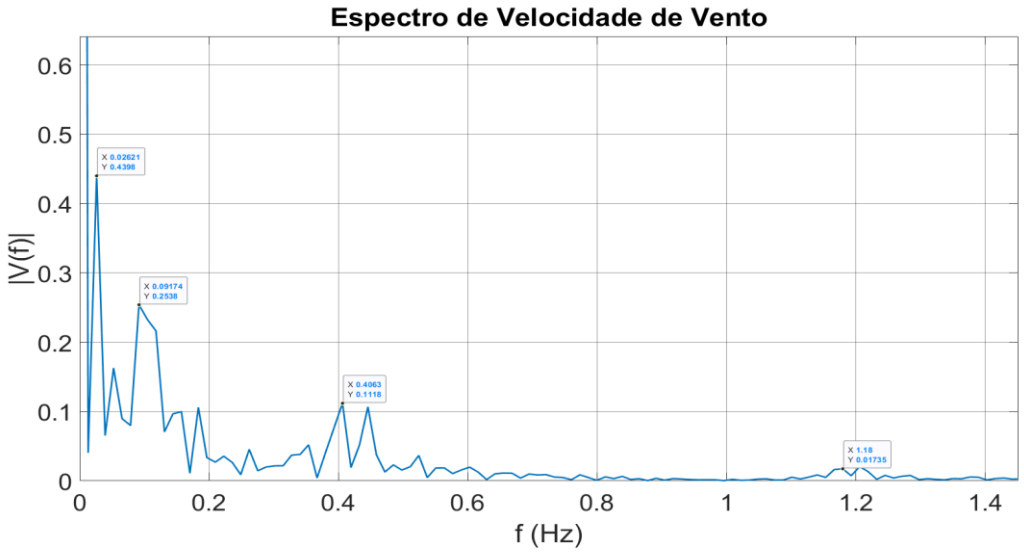
Fonte: Autor.

Na simulação, devido à idealidade dos componentes pode-se perceber que o rastreamento do torque de referência ocorre de forma satisfatória, resultando em boas resposta aos degraus de velocidade de vento impostos, assim como uma resposta satisfatória no seguimento do perfil de velocidade carregado na aplicação, que apresenta um perfil de segmento de potência semelhante ao perfil de vento imposto,

porém com uma característica de filtro passa baixa, eliminando as componentes de alta frequência que compõe o perfil de velocidade graças ao componente de inércia do aerogerador (Figura 57 e Figura 58).

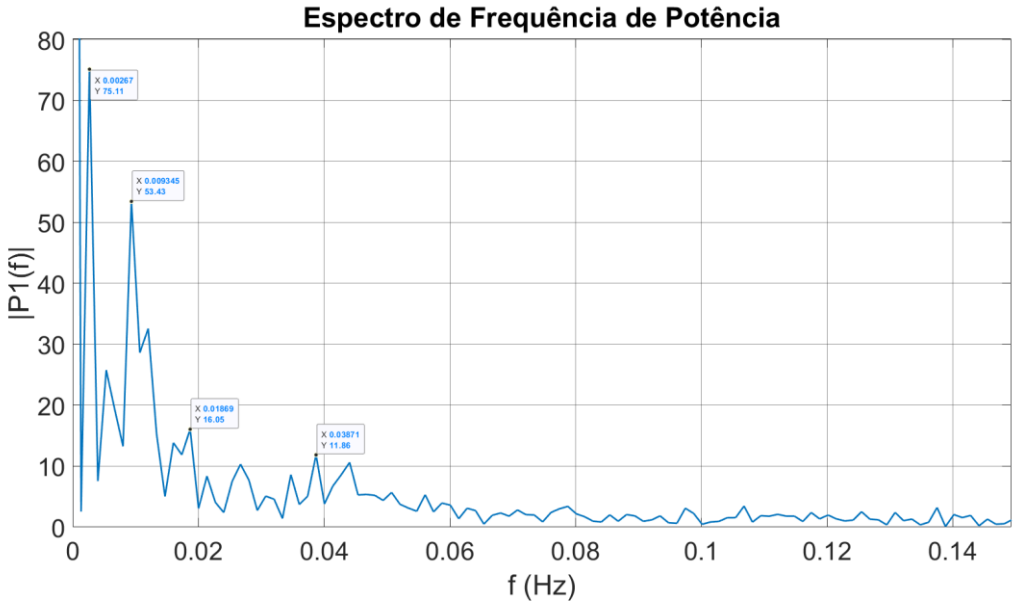
Os resultados obtidos na emulação se aproximam dos resultados obtidos na simulação com algumas ressalvas. Na emulação, há uma componente de alta frequência no sinal de torque lido que se propaga ao sinal de potência mecânica, causando distorções. Esta componente é resultante de um erro de leitura observado no transdutor de torque para os valores de torque real. Este erro de leitura foi observado e chega à ±1.5 N.m em torno do torque médio. Não foi possível indicar a origem deste erro de leitura, entretanto, a sua interferência não compromete a emulação dados os propósitos da aplicação. Também é verificado na Figura 51 que há um erro em regime permanente do torque real lido em relação ao torque de referência do aerogerador. Este erro em regime permanente deve-se ao integrador utilizado na ocasião da aquisição dos dados, que seguiu a modelagem do trabalho RIGHI (2015) utilizado como base para o desenvolvimento da bancada. Neste, o integrador utilizado tratava-se de um '*one step*', onde o valor atual é somado ao valor armazenado anterior. Esta abordagem permite atingir uma emulação que exige um menor processamento, mas que culmina em um erro em regime permanente semelhante à metodologia de controle por ganho proporcional. Posteriormente o integrador '*one step*' foi substituído por um integrador *Foward-Euler,* utilizado na simulação dos resultados apresentados anteriormente na seção 5.1.

Figura 57. Espectro de frequência do perfil de velocidade de vento na emulação



Fonte: autor.

Figura 58. Espectro de frequência da potência na emulação



Fonte: Autor.

A Tabela 5 contém as medidas de potência mecânica adquiridas nos testes de simulação e emulação da bancada eólica. Pode-se observar que as potências da simulação e da emulação apesar de seguirem um crescimento semelhante, diferem quanto valor para uma mesma velocidade de vento de forma razoável. Essa diferença deve-se ao fato de que os parâmetros da Tabela 3 utilizados para modelar o gerador da simulação tratam-se de um gerador de potência nominal maior (15 kW) do que o utilizado na emulação (11kW).

Tabela 5. Potências mecânicas verificadas.

Potências Mecânicas		
Velocidade de Vento [m/s]	Potência Simulação [W]	Potência Emulação [W]
3	145.6	119.01
4	290.2	234.93
5	491.7	391.89
6	748.8	577.24
7	1062	798.63
8	1428	1049.18

Fonte: autor.

Utilizando um osciloscópio de bancada, também foram adquiridas as medidas elétricas para o primeiro teste da emulação, onde degraus de velocidade de vento foram impostos de forma manual. As medidas elétricas de tensão e corrente de fase, bem como de potência trifásica encontram-se na Tabela 6.

Tabela 6. Medidas elétricas durante emulação.

Medidas Elétricas			
Velocidade de Vento [m/s]	Tensão de Fase [V]	Corrente de Fase [A]	Potência Trifásica [W]
3	10.63	2.57	81.96
4	15.29	3.73	171.09
5	19.59	5.41	317.95
6	23.9	6.65	476.80
7	28.3	7.82	663.92
8	32.71	8.08	792.89

Fonte: autor.

Utilizando os dados de potência mecânica estimada da Tabela 2 como referência, foram calculadas as variações percentuais das potências encontradas em relação a potência estimada para a respectiva velocidade de vento analisada. As variações encontram-se na Tabela 7.

Tabela 7. Variação das potências verificadas em relação as potências teóricas.

Variação em Relação a Potência Estimada			
Velocidade de Vento [m/s]	Potência Mecânica Simulação [%]	Potência Mecânica Emulação [%]	Potência Elétrica Emulação [%]
3	20.45%	-1.55%	-32.20%
4	19.77%	-3.04%	-29.39%
5	20.44%	-4.01%	-22.12%
6	20.92%	-6.79%	-23.00%
7	21.29%	-8.79%	-24.17%
8	21.29%	-10.89%	-32.66%

Fonte: autor.

Como é possível observar, os valores de potência mecânica encontrados para a simulação diferem em até 21.29% em relação aos valores estimados de potência mecânica. Esta diferença deve-se aos motivos supracitados de modelagem do gerador utilizado na simulação. As medidas de potência mecânica encontradas na emulação se aproximam de forma satisfatória dos valores de potência mecânica estimada, apresentando uma variação máxima para as velocidades de vento

analisadas de -10.89%. Já os valores de potência elétrica encontrados através das medidas elétricas de tensão e corrente variam até -32.66% das potências mecânicas estimadas. Isso deve-se ao fato de que as estimativas de potência elétrica da Tabela 7 não levavam em consideração as perdas do gerador, como por exemplo perdas elétricas resistivas e indutivas no estator da máquina e perdas nos condutores que conectam o gerador à carga resistiva.

5.4 Vantagens e desvantagens da solução

A utilização da solução que integra o *Simulink Real Time* e a aplicação .NET oferece diversas vantagens em relação à outras abordagens no controle de sistemas de emulação em tempo real como a bancada de emulação eólica. Algumas delas são:

- Infinidade de graus de liberdade para configurar a aplicação desenvolvida, uma vez que através da modelagem matemática de blocos utilizada pelo *Simulink* e sua ampla gama de opções é possível customizar de maneira simples a aplicação, incluindo filtros, diversas metodologias de integração, blocos de simulação de componentes elétricos e mecânicos, malhas de controle etc;
- Possibilidade de alterar parâmetros construtivos da modelagem em tempo real de forma simplificada;
- A utilização de uma segunda máquina (*target machine)* com a construção de um *kernel* próprio do *Simulink Real Time* para rodar de forma dedicada o modelo matemático da turbina eólica e da malha de controle permite que o desempenho da simulação seja amplamente aprimorado, podendo alcançar *sample rates* de até 20 kHz, além de contornar as interrupções dos sistemas operacionais que causam *jitter* e *lag* nos modelos de simulação.
- O emprego do *software Matlab*, que é amplamente utilizado no meio acadêmico e especialmente na área das engenharias, permite que a aplicação possa ser compreendida e aprimorada de forma simplificada por outros acadêmicos e futuros usuários da bancada de emulação eólica.
- A utilização de uma aplicação .NET com interface gráfica para realizar o controle sobre a emulação permite que o usuário tenha disponível um meio simples e intuitivo de controlar a emulação e visualizar os dados, além de

permitir a total customização e personalização para qualquer usuário que possua conhecimentos básicos/intermediários em programação C#.

- A utilização de um modelo matemático da turbina eólica e da malha de controle pré-compilado conjuntamente com o controle e visualização de dados através de uma aplicação .NET permite que a emulação eólica sequer necessite do *software Matlab* para a sua execução, sendo naturalmente uma solução que permite grande portabilidade aos usuários da bancada.

Entretanto algumas características do *Simulink Real Time*, especialmente relacionadas à integração da API .NET tornam-se desvantagens, como:

- Apesar dos diversificados blocos do *Simulink Real Time* que permitem comunicação TCP, UDP e Serial, não há disponibilidade gratuita de blocos para tratar de protocolos de comunicação industrial como o *Modbus.* Para ter acesso a estes blocos, é necessário realizar a aquisição da *target machine* vendida pela própria *Mathworks,* a Speedgoat.
- Pela alteração da estratégia de vendas da companhia ao incentivar a compra da *target machine*, a *Mathworks* dificultou o acesso à documentações e suporte técnico ao *Simulink Real Time,* e apesar da diversificada documentação existente no *site* da *Mathworks* sobre as classes, métodos e propriedades das bibliotecas que integram a API .NET, há poucas documentações sobre exemplos de utilização e informações específicas, bem como foram encontradas poucas informações na comunidade de desenvolvimento sobre a utilização desta solução.
- Devido a necessidade de realizar a comunicação com todos os componentes do sistema de emulação através da aplicação .NET na *host machine,* é necessário haver um cuidado especial na escolha dos tempos de execução da rotina de comunicação, sendo este um ponto a ser aprimorado na atual aplicação implementada.
- A utilização de padrões de protocolo serial como o RS 232 implementada para comunicação com o inversor traz problemas inerentes à esta modalidade de comunicação, como a comum perda de mensagens e a baixa velocidade na comunicação para este tipo de aplicação em tempo real.

6 CONCLUSÃO

O presente trabalho se dispôs a criar uma aplicação de simples utilização e grande portabilidade para implementar um método de emulação em tempo real de uma turbina eólica utilizando uma bancada com inversor, motor, transdutor de torque, gerador e carga. A solução baseou-se na integração de uma aplicação desenvolvida .NET que realiza o controle e a comunicação com os componentes do sistema com a modelagem matemática da turbina e sua simulação em tempo real provida pela *toolbox* do *software Matlab, Simulink Real Time.*

Foi constatado que o *Simulink Real Time* é uma solução eficiente e versátil para aplicação em simulações de tempo real, já que disponibiliza diversas ferramentas para modelagem de sistemas físicos, bem como oferece suporte para diversas modalidades de comunicação, além de produzir bons resultados ao utilizar um *kernel* próprio compilado em C para rodar os modelos de simulação de forma exclusiva em qualquer máquina que possua placa de rede compatível.

Apesar das limitações impostas pela modelagem utilizada para realizar a emulação e a imprecisão do transdutor de torque, as potências mecânicas médias mensuradas no teste em bancada se aproximaram de forma satisfatória das potências previstas utilizando os pontos de operação da turbina eólica com a carga resistiva trifásica expectados. Utilizado com um perfil de vento próximo à um perfil real, a aplicação desenvolvida foi capaz de reproduzir de forma satisfatória o comportamento observado em simulação. Entretanto melhorias podem ser implementadas em trabalhos futuros, onde há possibilidade de realização de um estudo de perdas nas máquinas e topologias utilizadas para compensação no modelo desenvolvido objetivando aproximar mais os valores das grandezas elétricas mensuradas. Um estudo com uma proposta de mitigação do erro de leitura de torque também se faz necessário para uma maior acurácia no modelo de emulação desenvolvido. Implementação de novas funcionalidades como aquisição de dados de vento diretamente de leituras anemométricas e adição de um módulo para leituras de grandezas elétricas diretamente na aplicação também serão bem-vindas.

A aplicação desenvolvida neste trabalho abre perspectivas sobre o desenvolvimento de sistemas de emulação em tempo real de turbinas eólicas utilizando a ferramenta *Simulink Real Time* integrada à API .NET para a bancada utilizada pelo Instituto de Eletrônica de Potência da Universidade Federal de Santa

Catarina, provendo um meio simples de modelagem e controle que certamente auxiliará trabalhos futuros que utilizarão a modelagem de turbinas eólicas.

7 BIBLIOGRAFIA

COLLIER, Daniel Augusto Figueirêdo – Modelagem e controle de retificadores PWM trifásicos conectados a geradores síncronos a ímãs permanentes em sistemas de conversão de energia eólica [dissertação] / Daniel Augusto Figueirêdo Collier; orientador, Marcelo Lobo Heldwein. – Florianópolis, SC, 2011.

LOPES, Afonso Martins Revidiego - Aplicação da simulação *Hardware* in the Loop para testes e desenvolvimento de suspensões veiculares / Afonso Martins Rividiego Lopes – Campinas, SP: [s.n], 2017.

NETO, Luis Juarez Castelo Branco Camurça – Sistema de Conversão de Energia Eólica de Alta Eficiência Utilizando o Conversor Delta Tipo-T e Minimização das Perdas Baseada em Modelo / Luis Juarez Castelo Braco Camurça; orientador, Marcelo Lobo Heldwein – Florianópolis, SC, 2017.

PASSOS, Wladimir de Almeida – Utilização de ferramentas de prototipagem rápida direcionada à concepção de sistemas embarcados baseados em computação reconfigurável / Wladimir de Almeida Passos – Campinas, SP: [s.n], 2008.

MORREN, J., PIERIK,J. - Inertial response of variable speed wind turbines. Electric Power Systems Research, 2006, p. 980-987.

MANWELL, J. F. et al. Wind energy explained: theory, design, and application. 2nd Ed. Inglaterra: Editora Wiley, 2009.

BURTON, T. et al. Wind energy handbook. 1nd Ed. Inglaterra: Editora Wiley, 2001.

WEG (2010). RS232 / RS485 Serial Communication Manual. Jaraguá do Sul, SC.

MUSSA, S. et al. – Conversor Para Interligação de um Gerador Eólico com a Rede Elétrica. Florianópolis, 2010.

GREGA, W. *Hardware* in the loop simulation and its application in control education. 29TH ASEE/IEEE Frontiers in Education Conference, 1999, Session 12b6, pp. 7-12.

CASTELUCCI, Daniella. Protocolos de Comunicação em Redes de Computadores**.** 2011. Disponível em: <https://daniellacastelucci.wordpress.com/2011/04/08/protocolos-de-comunicacao-em-redes-de-computadores/>. Acesso em 10 jun. 2020.

NATIONAL INSTRUMENTS. O protocolo *Modbus* em detalhes. 2019. Disponível em: <https://www.ni.com/pt-br/innovations/white-papers/14/the-*Modbus*-protocol-in-depth.html>. Acesso em 05 out. 2020.

INTERFACE [200-?]. A Flexible Command Set for Digital Sensors and Interfaces

MICHEL, Germano Veit – Estudo de Mecanismos FEC para Transmissão Confiável em UDP [Trabalho de Conclusão de Curso] / Germano Veit Michel: orientador, Prof. Dr. Sérgio Luis Cechin – Porto Alegre, RS, 2010.

TIBOLA, G. Sistema Eólico de Pequeno Porte Para Geração de Energia Elétrica com Rastreamento de Máxima Potência. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, mar. 2009.

TELLES, Brunelli Lazarin. Turbinas Eólicas. 2018. 27 slides.

HEIER, S. Grid Integration of Wind Energy Conversion Systems. Inglaterra: Wiley, 2006. v. 2a edição.

VOLNYS, Borges Bernal – Protocolo UDP. 15 slides, 2016. <https://edisciplinas.usp.br/pluginfile.php/4284866/mod_resource/content/1/61-Revisao-udp-v5.pdf>. Acesso em 08 nov. 2020.

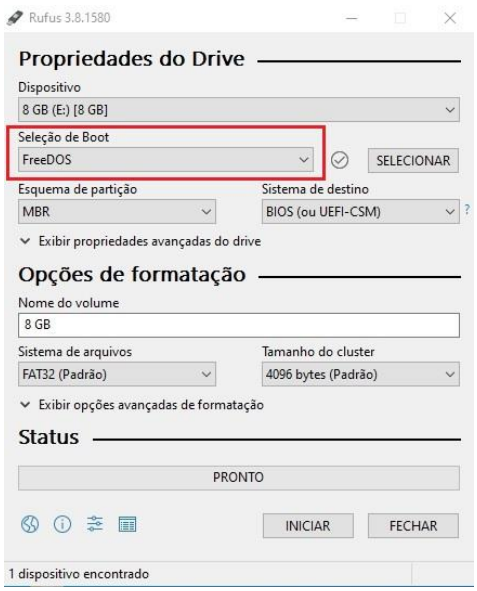
APÊNDICE A – PROCEDIMENTO PARA OPERAÇÃO *STAND ALONE*

Para utilizar a aplicação de controle da bancada de emulação eólica criada no *Simulink Real Time* de forma *stand-alone*, isto é, sem a utilização do *software Matlab* para inicializar a *target machine* e carregar o modelo .mldtax, uma das opções é utilizar um *pen drive* para realizar o *boot* do sistema e carregar o *kernel* do *Simulink Real Time*, para posteriormente realizar o carregamento do modelo .mldtax na *target machine* pela aplicação .NET.

O *kernel* gerado pelo *Simulink Real Time* é compatível com o sistema operacional DOS (*Disk Operating System)*, logo é necessário primeiramente criar um *pendrive bootável* com esse sistema operacional. Para realizar esse procedimento é recomendado a utilização do *software* de formatação de *pendrive Rufus*. Este *software* é um *freeware* que, além de realizar a formatação de dispositivos móveis, também oferece a funcionalidade de criar um *pendrive bootável* com o sistema *FreeDOS,* uma versão gratuita do sistema operacional *DOS* e também compatível com o *kernel* do *Simulink Real Time.*

Para realizar a criação do *pendrive bootável* com sistema *FreeDOS,* basta conectar o dispositivo de armazenamento removível no computador, abrir o executável do *Rufus,* selecionar o dispositivo desejado, escolher na *Seleção de Boot* a opção *FreeDOS* e iniciar a formatação do *pendrive* (Figura 59).

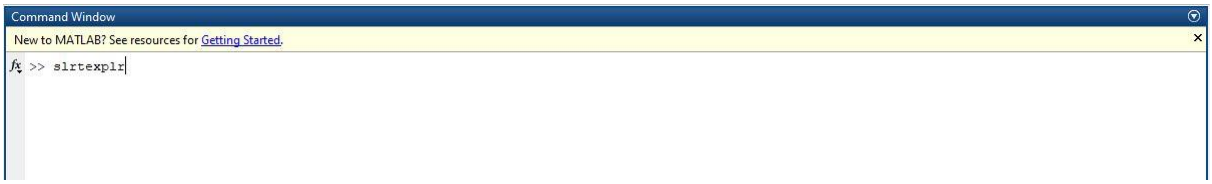
Figura 59. *Software* RUFUS.



Fonte: Autor

Após a criação do *pendrive bootável* com o sistema operacional *FreeDOS,* é necessário realizar a compilação e o carregamento no *pendrive* do *kernel* do *Simulink Real Time.* Para isso, primeiramente na janela de comando do *software* executa-se o comando *slrtexplr* para abrir o *Simulink Real Time Explorer* (Figura 60)

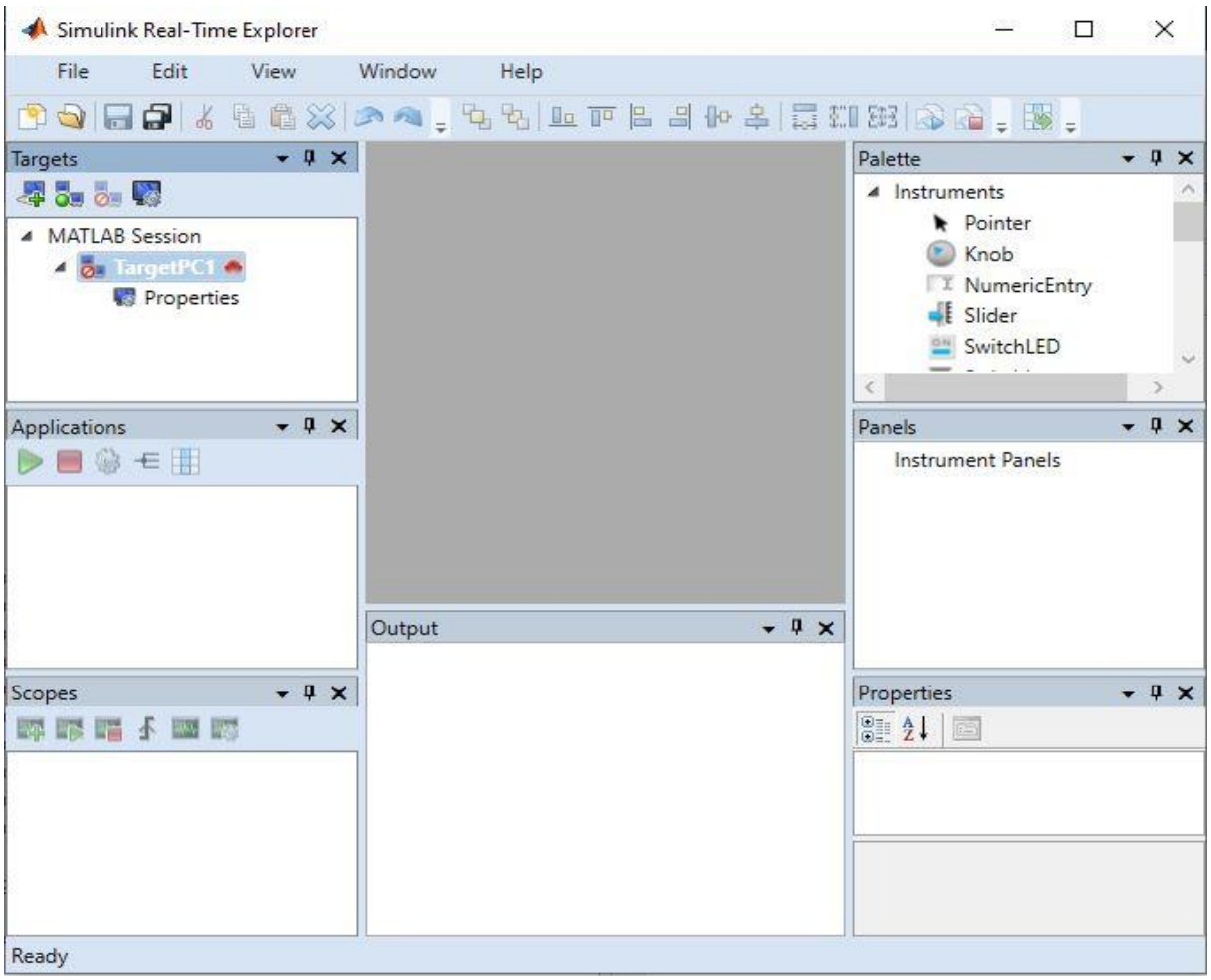
Figura 60. Janela de comando *Matlab.*



Fonte: autor.

Ao abrir a janela do *Simulink Real Time Explorer*, seleciona-se a opção *Properties* na área de *Targets* (Figura 61)

Figura 61. Janela do *Simulink Real Time* Explorer.

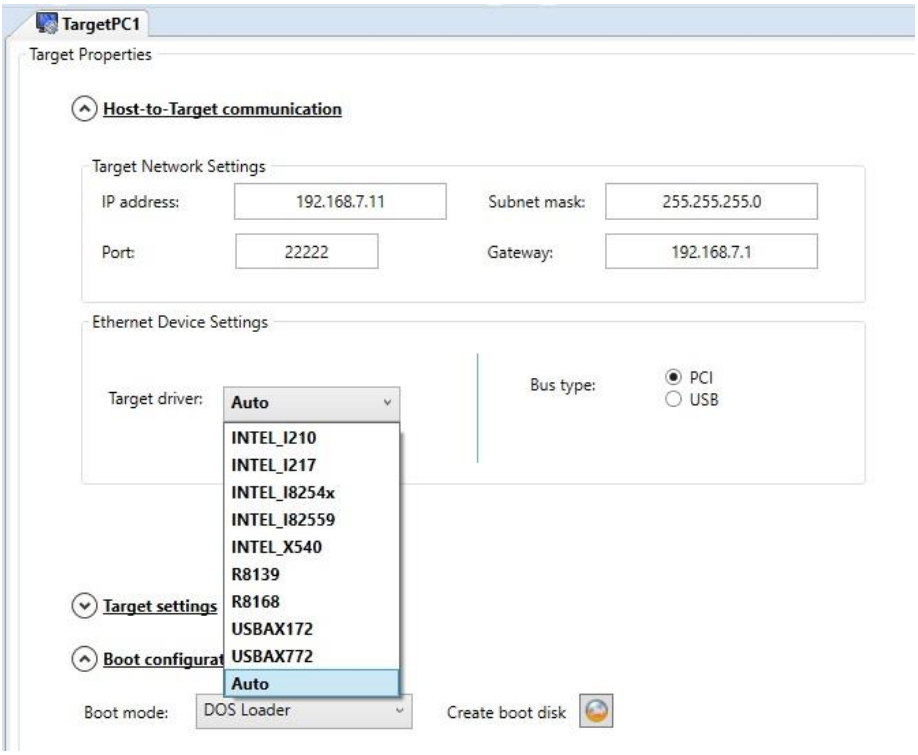


Fonte: autor.

Na aba *Host-to-Target communication* são configurados os parâmetros de comunicação entre a *host machine* e a *target machine.* Os parâmetros de comunicação que precisam ser definidos são:

- IP Adress: endereço de IP que será utilizado pela *target machine;*
- Port: porta de comunicação que será utilizada pela *target machine* para receber os dados de comunicação da *host machine*;
- *Subnet Mask:* máscara de subrede utilizada na comunicação;
- *Gateway:* escolha do *Gateway* comum utilizado pela comunicação *host-target;*
- *Target driver:* escolha do modelo de *driver ethernet* utilizado pela placa de rede da *target machine*. Os modelos de placa de rede suportados pelo *Simulink Real Time* estão indicados na Figura 62;
- Bus type: define qual o tipo de barramento utilizado pela *target machine.*

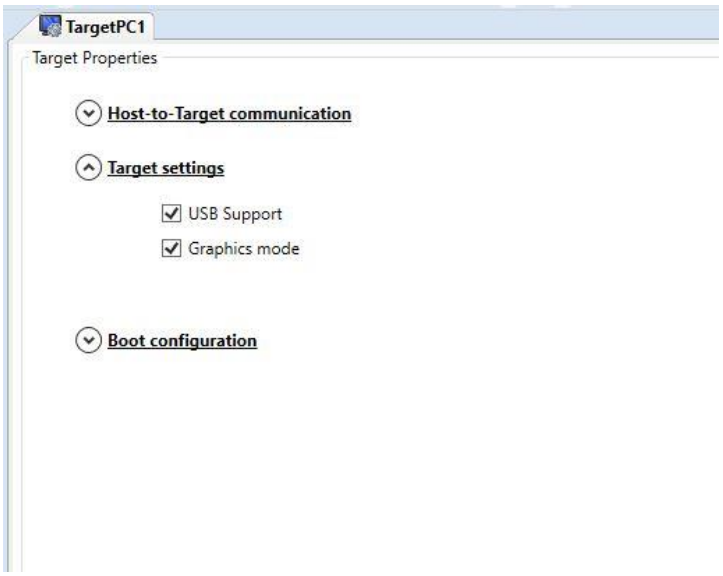
Figura 62. Janela de propriedades para a *target machine* com os respectivos drivers de comunicação ethernet compatíveis com o *Simulink Real Time.*



Fonte: autor.

Na aba *target settings,* marca-se as *check box USB Support* e *Graphics mode,* para que a *target machine* tenha suporte USB e realize a exibição dos sinais dos *Targets Scopes* em qualquer monitor que estiver conectado à *target machine,* respectivamente.

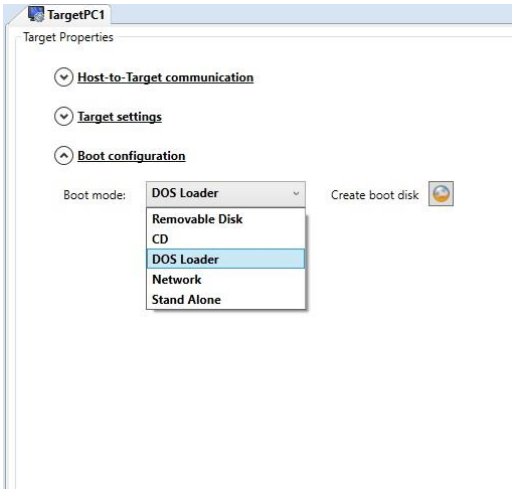
Figura 63. Aba de configuração da *target machine.*



Fonte: autor.

Após configurado os parâmetros de comunicação e suporte do *kernel* do *Simulink Real Time,* na aba *Boot configuration* escolhe-se na lista suspensa a opção de *boot* DOS *Loader.*

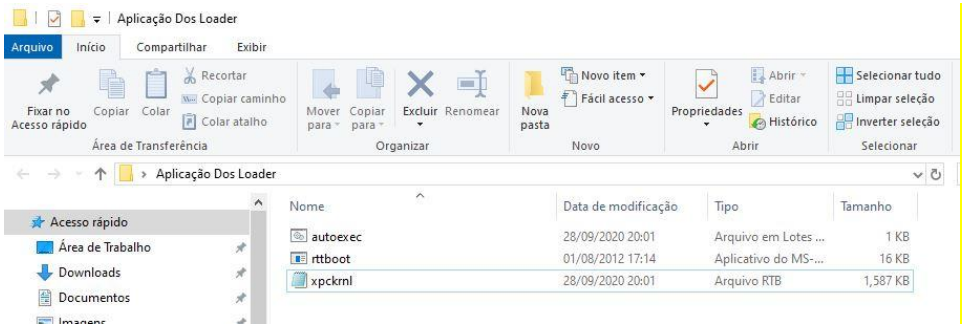
Figura 64. Aba de configuração do método de boot para a *target machine.*



Fonte: autor.

Após realizar as configurações, basta clicar no botão *Create boot disk* e selecionar a pasta de destino. O *Simulink Real Time* realizará a compilação do *kernel* e criara os arquivos da Figura 65.

Figura 65. Arquivos gerados durante a criação do *kernel* do *Simulink Real Time.*

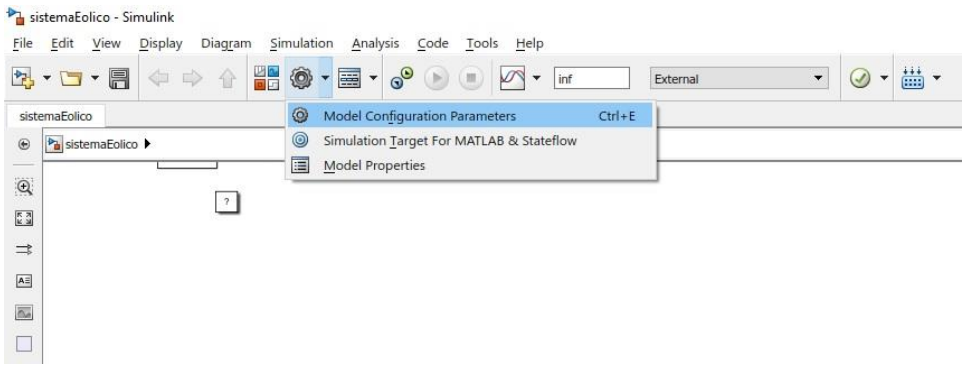


Fonte: autor.

Após a criação dos arquivos, basta copiá-los para o *pendrive bootável* criado anteriormente e o *pendrive* já estará pronto. Para executá-lo, basta conectá-lo à *target machine* desligada, iniciar a *target machine,* escolher na BIOS da máquina a opção de *boot* por *USB* e o sistema *FreeDOS* será carregado, executando automaticamente a inicialização do *kernel Simulink Real Time.*

Para criar o modelo de simulação .mldtax do *Simulink Real Time* que será carregado na *target machine* para a execução, é necessário configurar o *Simulink* para realizar a compilação do modelo. Para isso, após o modelo ter sido criado no *Simulink*, configura-se o modelo nas configurações *Model configuration parameters,* localizada na barra de ferramentas (Figura 66).

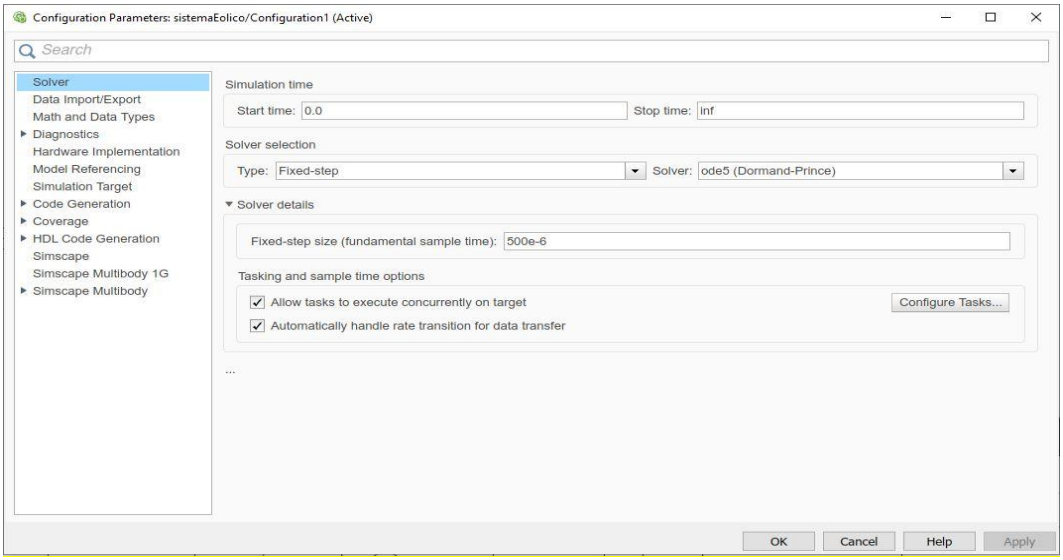
Figura 66. Model Configuration Parameters no *Simulink*.



Fonte: autor.

Na aba *Solver*, altera-se o *Stop Time* da execução do modelo para *INF*, o *Type* para *Fixed-Step* e escolhe-se o *Sample Time* de operação do modelo da *box Fixed Step Size* (Figura 67)

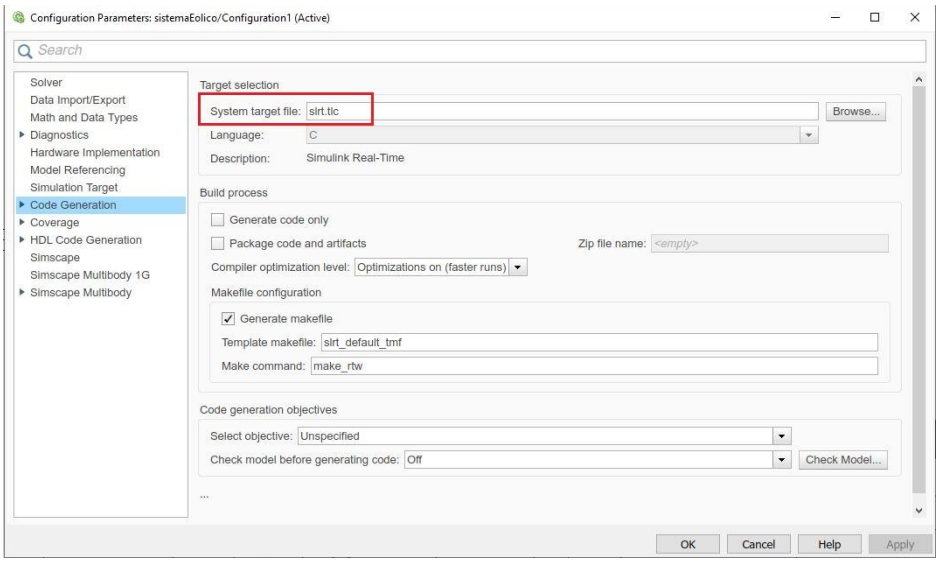
Figura 67. Aba de configuração do solver utilizado pelo *Simulink Real Time.*



Fonte: autor.

Na aba *Code Generation*, é selecionado no *System target file* a opção *slrt.tlc,* referente à criação de um modelo compilado em linguagem C para carregar na *target machine* (Figura 68).

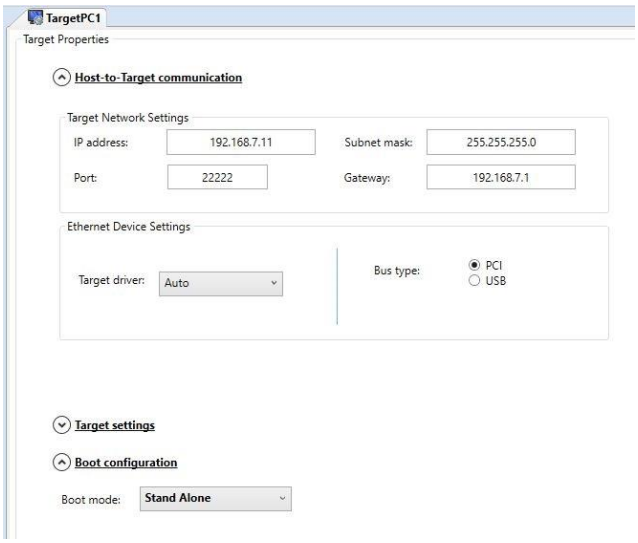
Figura 68. Aba de configuração da geração de código para os modelos do *Simulink Real Time.*



Fonte: autor.

Após a configuração do modelo, aplicam-se as alterações. O próximo passo é retornar ao *Simulink Real Time Explorer* (comando *slrtexplr)* e selecionar desta vez nas aba de *Boot configuration* a opção de modo de *boot Stand Alone* (Figura 69).

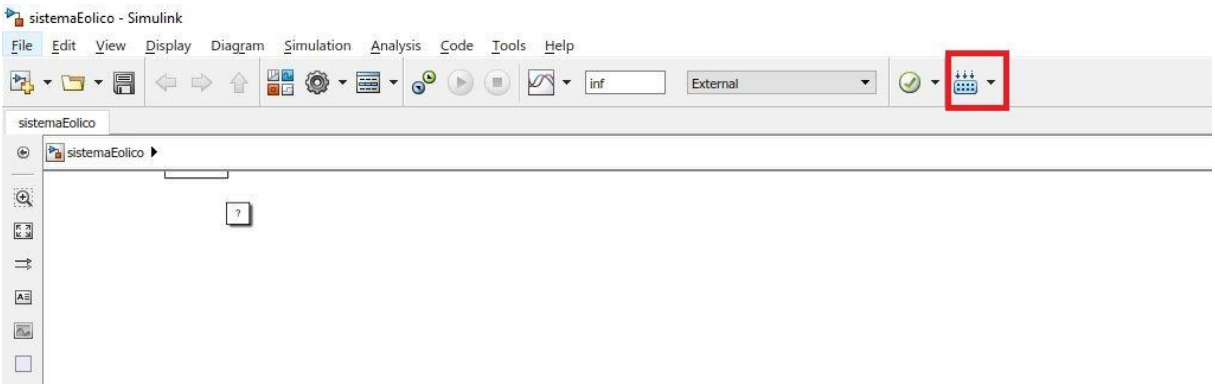
Figura 69. Janela para seleção do método de *boot Stand-Alone*.



Fonte: autor.

Retornando ao *Simulink,* o modelo já está pronto para ser construído e compilado. Basta clicar no botão *Build* da barra de ferramentas (Figura 70) e o modelo de blocos criado no *Simulink* será compilado e um arquivo .mldtax será criado na pasta raiz selecionada no *Matlab.* Após a realização do *boot* da *target machine* discutido anteriormente, o modelo .mldtax gerado pode ser carregado a qualquer momento através da aplicação .NET.

Figura 70. Botão de Build para gerar o arquivo .mldtax utilizado na *target machine.*



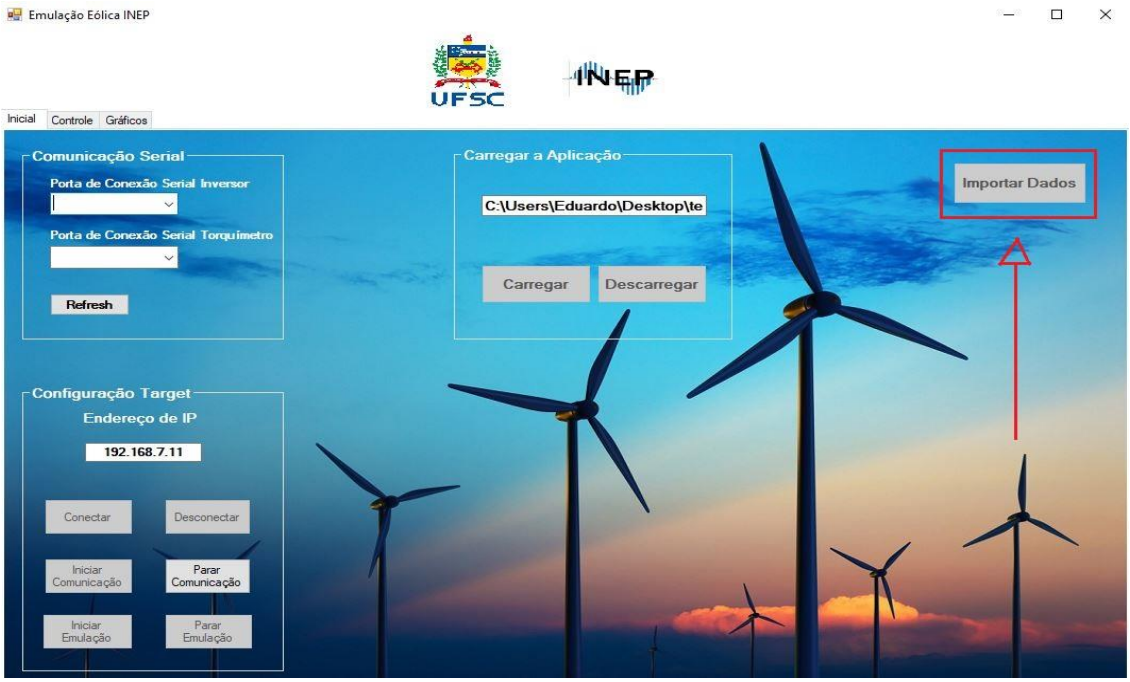
Fonte: autor.

APÊNDICE B – OBTENÇÃO DOS DADOS DE EMULAÇÃO

Após finalizada a operação de emulação da bancada eólica com o *Simulink Real Time* e a aplicação .NET, é possível obter as leituras de velocidade de vento, potência mecânica, torque de referência, torque real e velocidade mecânica lidos pelo transdutor *Interface* referentes a todo o período de execução da emulação. Esta ferramenta foi implementada através de *File Scopes* inseridos no modelo do *Simulink Real Time* que roda na *target machine*, que armazenam os dados em arquivos .DAT na memória da *target machine* e estão disponíveis após a finalização da emulação.

Para transferir os arquivos que contém os dados da emulação da *target machine* para a *host machine*, após a finalização da emulação o botão *Importar Dados* da aba *Inicial* na aplicação .NET é liberado ao usuário para permitir a transferência dos arquivos.

Figura 71. Tela inicial da aplicação com o botão *Importar Dados.*

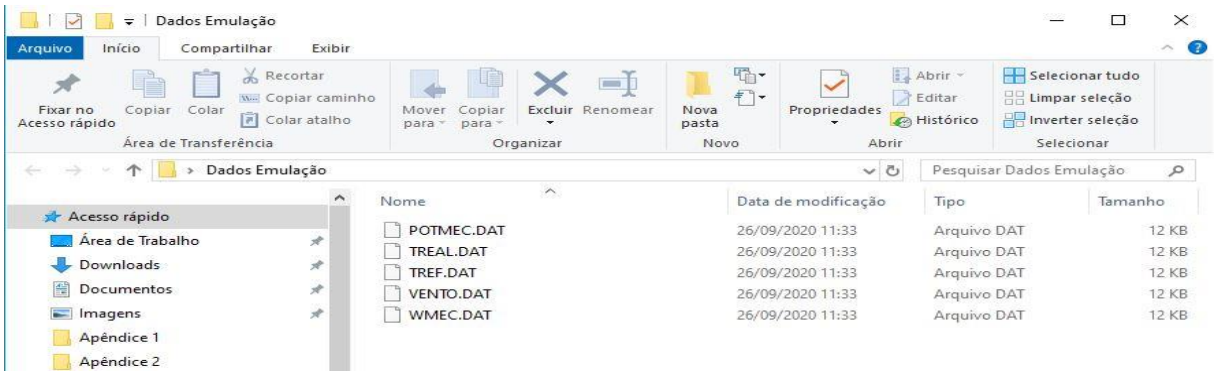


Fonte: Autor.

Ao apertar o botão *Importar Dados,* o usuário seleciona a pasta de destino para qual os arquivos serão transferidos. São então transferidos da *target machine* para a *host machine* cinco arquivos .DAT: POTMEC.DAT, contendo os dados de potência mecânica; TREAL.DAT, contendo os dados de torque mecânico lidos pelo transdutor *Interface*; TREF.DAT, contendo os dados de torque de referência calculados pelo

modelo matemático da turbina eólica; VENTO.DAT, contendo as velocidades de vento utilizadas na emulação e WMEC.DAT, contendo as velocidades mecânicas lidas no eixo do motor acionador pelo transdutor Interface.

Figura 72. Arquivos de dados gerados e transferidos para a *host machine.*



Fonte: Autor.

Os arquivos .DAT trata-se de arquivos binários que só podem ser lidos pelas mesmas estruturas de dados que os geraram. Logo, para realizar a aquisição correta dos dados contidos nestes arquivos, é necessário a utilização do *software Matlab.*

Para a leitura destes dados no *software Matlab*, utiliza-se o seguinte *script* para conversão dos dados.

Figura 73. *Script* para leitura dos dados de emulação contidos nos arquivos .DAT.

Editor - C:\Users\Eduardo\Desktop\carregarDadosEmulacao.m

carregarDadosEmulacao.m

```
h = fopen('TReal.dat');
slrtfile_data = fread(h);
fclose(h);
matlab_data_TReal = SimulinkRealTime.utils.getFileScopeData(slrtfile_data);

h = fopen('TRef.dat');
slrtfile_data = fread(h);
fclose(h);
matlab_data_TRef = SimulinkRealTime.utils.getFileScopeData(slrtfile_data);

h = fopen('Vento.dat');
slrtfile_data = fread(h);
fclose(h);
matlab_data_Vento = SimulinkRealTime.utils.getFileScopeData(slrtfile_data);

h = fopen('WMec.dat');
slrtfile_data = fread(h);
fclose(h);
matlab_data_WMec = SimulinkRealTime.utils.getFileScopeData(slrtfile_data);

h = fopen('PotMec.dat');
slrtfile_data = fread(h);
fclose(h);
matlab_data_PotMec = SimulinkRealTime.utils.getFileScopeData(slrtfile_data);

erro = matlab_data_TRef.data(:,1) - matlab_data_TReal.data(:,1);
```

Fonte: Autor.