INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* BETIM

BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Arthur Carlos de Faria

**Modelagem e Controle de Robô Quadrúpede Utilizando Redes Neurais Preditivas e Adaptativas**

Betim

2022

ARTHUR CARLOS DE FARIA

**Modelagem e Controle de Robô Quadrúpede Utilizando Redes Neurais Preditivas e Adaptativas**

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Leandro Freitas de Abreu

Coorientador: Virgil Del Duca Almeida

Betim

2023

(Página onde será inserida a ficha catalográfica- Solicitar junto à biblioteca)

Arthur Carlos de Faria

**Modelagem e Controle de Robô Quadrúpede Utilizando Redes Neurais Preditivas e Adaptativas**

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Aprovado em: \_\_\_/\_\_\_/\_\_\_\_\_\_ pela banca examinadora:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.(a) Dr. XXXXXXXXX (Orientador) – IFMG Campus Betim

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.(a) Me. XXXXXXXXX (Coorientador) – IFMG Campus Betim

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Dr. XXXXXXXXX– Instituição

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Eng.(a) XXXXXXXXX– Instituição

**Resumo**

Ao utilizar redes Neurais, é possível determinar padrões, correlações, classificar e prever resultados, portanto, pretende-se utilizar essas características como fonte de estudo para, através de uma Rede Neural preditiva, modelar um Sistema Embarcado baseado em Robôs Quadrúpedes para sintetizar um controlador Neural, de forma que este seja capaz de adaptar-se aos dados de entrada, isto é, aprender com o ambiente através dos dados colhidos pelos sensores.

**Palavras-chave:** Robô quadrúpede; Redes Neurais Preditivas; Redes Neurais Adaptativas; Controle de Sistemas.

**ABSTRACT**

By using Neural networks, it is possible to determine patterns, correlations, classify and predict results, therefore, it is intended to use these characteristics as a source of study to, through a predictive Neural Network, model an Embedded System based on Quadruped Robots to synthesize a controller Neural, so that it is able to adapt to the input data, that is, learn from the environment through the data collected by the sensors.

**Keywords:** Quadruped Robot; Predictive Neural Networks; Adaptive Neural Networks; Systems Control.

**LISTA DE ILUSTRAÇÕES**

[1 INTRODUÇÃO 15](#_Toc140672713)

[- 1.1. Justificativa 15](#_Toc140672714)

[- 1.2. Colocação do Problema 15](#_Toc140672715)

[- 1.3. Objetivos 16](#_Toc140672716)

[- 1.4 Organização do Trabalho 16](#_Toc140672717)

[- 2 FUNDAMENTAÇÃO TEÓRICA 17](#_Toc140672718)

[- 2.1 Predição de Estados Baseado em Modelo de Referência 17](#_Toc140672719)

[- 2.1.1 Controle Preditivo 17](#_Toc140672720)

[- 2.1.1 Método do Modelo Interno 18](#_Toc140672721)

[- 2.1.1 Redes Neurais Preditivas em sistemas desconhecidos 19](#_Toc140672722)

[- 2.3 Persistência de Excitação 21](#_Toc140672723)

[- 3 METODOLOGIA 22](#_Toc140672724)

[- 3.1 Objetivos impostos ao robô quadrúpede 22](#_Toc140672725)

[- 3.1 CRONOGRAMA 26](#_Toc140672726)

[- 3.2 RECURSOS 27](#_Toc140672727)

[- 3.2.1 Materiais 27](#_Toc140672728)

[- 3.2.2 Ferramentas computacionais 28](#_Toc140672729)

[- 3.2.3 Equipamentos Utilizados 29](#_Toc140672730)

[- 3.3 MONTAGEM DO SISTEMA 29](#_Toc140672731)

[- 3.3.1 Modelo em 3D 29](#_Toc140672732)

[- 3.4 Estudo e Modelagem do Sistema Com Rede Neural Preditiva 30](#_Toc140672733)

[- 3.4.1 Obtenção dos dados de treinamento 31](#_Toc140672734)

[- 3.4.2 Construção da rede preditiva 33](#_Toc140672735)

[- 3.4.3 Instânciando a rede preditiva 35](#_Toc140672736)

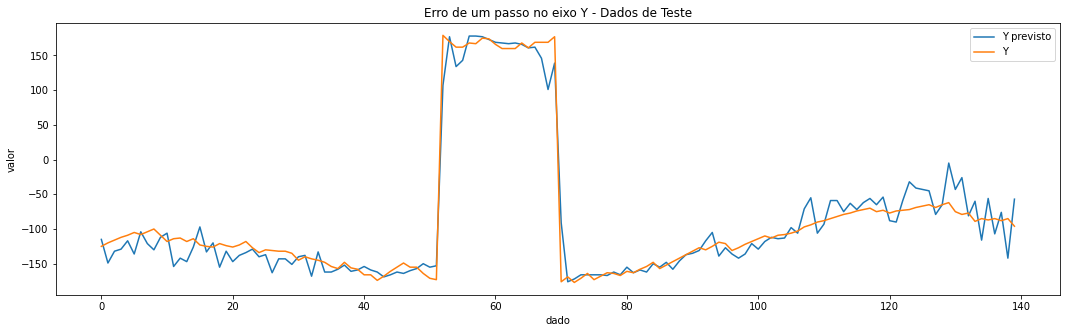
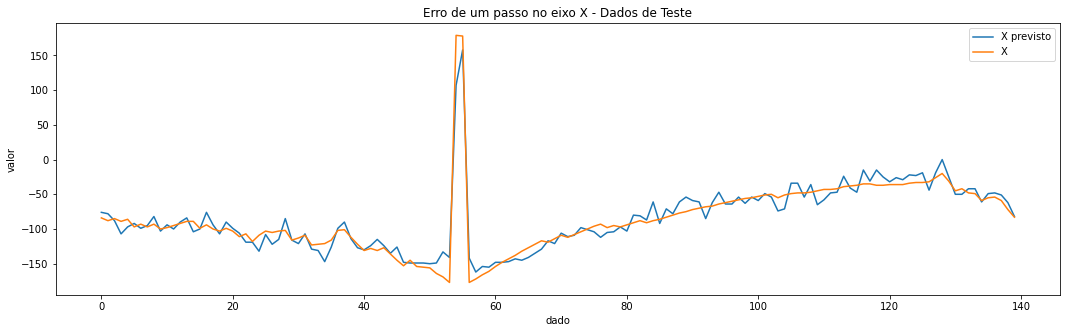
[- 3.4.4 Algoritmo de treinamento da Rede Neural Preditiva 36](#_Toc140672737)

[- 3.4.4.1 Parâmetros de aprendizado e dados de treinamento 36](#_Toc140672738)

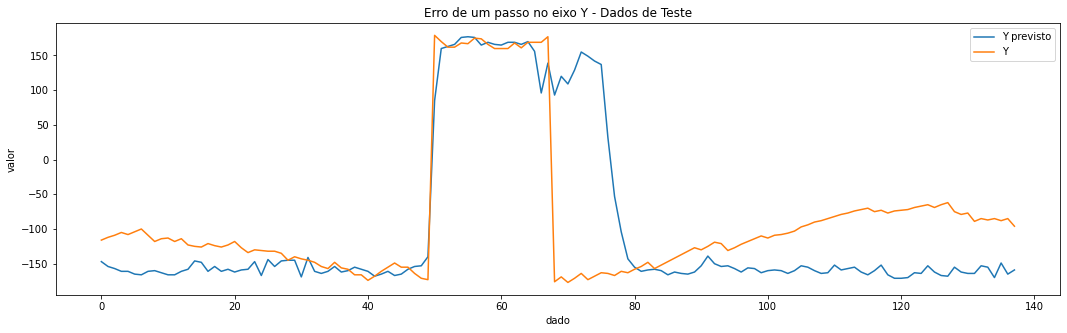
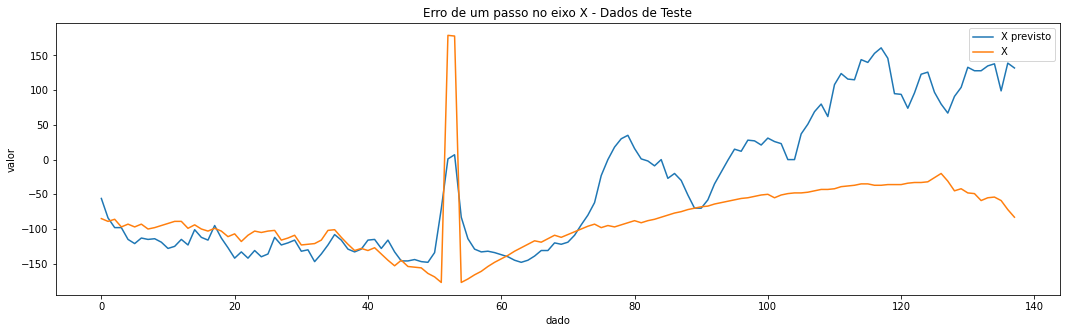
[- 3.4.4.2 Loop de treinamento e condição de parada 36](#_Toc140672739)

[- 3.4.4.3 Resultados do pré-treinamento 38](#_Toc140672740)

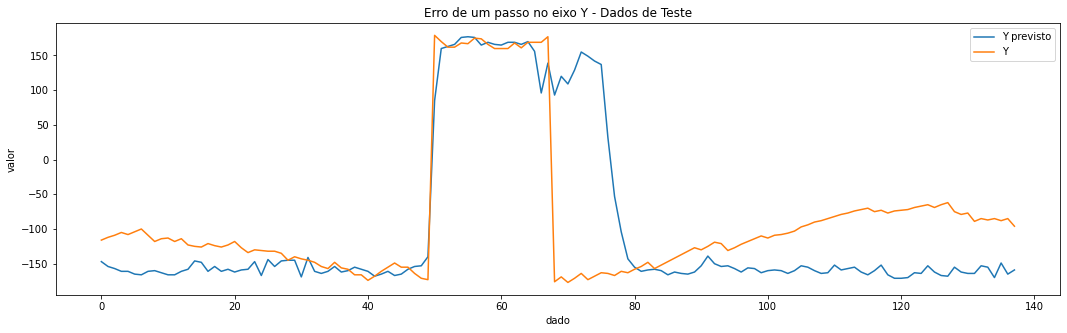
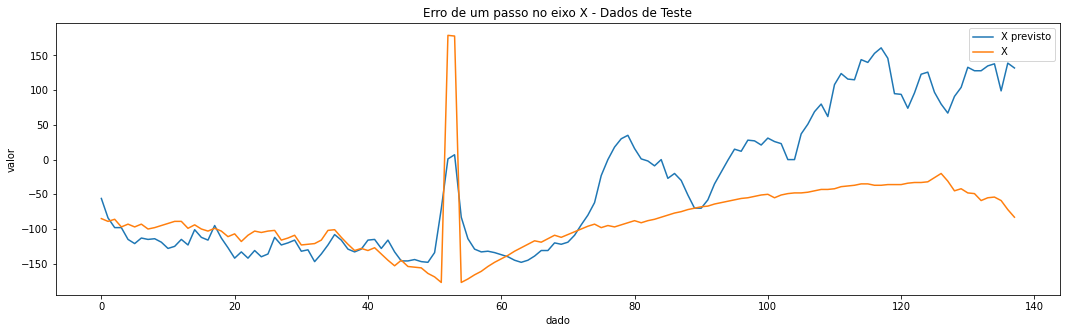
[- 3.4.4.4 Teste de previsão de 1 passo à frente 43](#_Toc140672741)

[ 43](#_Toc140672742)

[- 3.4.4.5 Teste de previsão de 2 passos à frente 43](#_Toc140672743)

[ 43](#_Toc140672744)

[- 3.4.4.6 Teste de previsão de 3 passos à frente 43](#_Toc140672745)

[ 44](#_Toc140672746)

[- 3.4.4.7 Teste de predição Livre 44](#_Toc140672747)

[Gráfico, Gráfico de linhas

Descrição gerada automaticamenteUma imagem contendo Gráfico de linhas

Descrição gerada automaticamente 44](#_Toc140672748)

[Referências 48](#_Toc140672749)

**LISTA DE QUADROS**

[Quadro 1.2 – Referência para estruturação de um trabalho de conclusão de curso (TCC) de um curso de graduação. Essa estrutura não é obrigatória. 15](#_heading=h.1fob9te)

**LISTA DE TABELAS**

Tabela 1 – Vetor Resultante

Tabela 2 – Variáveis do Sistema

Tabela 3 – Cronograma

Tabela 4 – Lista de Materias

**Sumário**

[**INTRODUÇÃO**](#_heading=h.30j0zll) **15**

[1.1. Justificativa](#_heading=h.3wmeswqmr0mn) 15

[1.2. Colocação do Problema](#_heading=h.e5gcc9tf3j81) 15

[1.3. Objetivos](#_heading=h.w6ci6ffdk2xt) 15

[1.4 Organização do Trabalho](#_heading=h.s4nu3zr8g3lg) 16

[**2 FUNDAMENTAÇÃO TEÓRICA**](#_heading=h.eunxazg21o7m) **17**

[2.1 Controle Preditivo Baseado em Modelo Interno](#_heading=h.pcmhxm8ja1hs) 17

[2.1.1 Controle Preditivo](#_heading=h.wdns06z2ay1s) 17

[2.1.1 Método do Modelo Interno](#_heading=h.x146cf58m8xt) 18

[2.1.1 Redes Neurais Preditivas em sistemas desconhecidos](#_heading=h.dhikweckszxm) 19

[2.5 Persistência de Excitação](#_heading=h.4dyr8jvmoj3u) 21

[**3 FUNDAMENTAÇÃO TEÓRICA**](#_heading=h.vx6osy6d42vr) **22**

[3.1 Objetivos impostos ao robô quadrúpede](#_heading=h.tw5vq0kbpt5e) 22

[3.1 CRONOGRAMA](#_heading=h.vj035dr78q1e) 28

[3.2 RECURSOS](#_heading=h.bfxcro2va3kk) 29

[3.2.1 Materiais](#_heading=h.8z1ko2gw2eak) 29

[3.2.2 Ferramentas computacionais](#_heading=h.xxlz55gt0xz7) 30

[3.2.3 Equipamentos Utilizados](#_heading=h.2fif08jrfsb5) 31

[3.3 MONTAGEM DO SISTEMA](#_heading=h.4lgpqy5fue0v) 31

[3.3.1 Modelo em 3D](#_heading=h.o626vgr1za9p) 31

[3.4 Estudo e Modelagem do Sistema](#_heading=h.gwds6otzxh91) 32

[Referências](#_heading=h.44sinio) **34**

# INTRODUÇÃO

A utilização de robôs com controlador embutido, ou como são conhecidos, os sistemas embarcados, é crescente, e atualmente existem diversas tentativas de fazer com que alcancem a inteligência artificial, para que sejam capazes de analisar e aprender com o seu ambiente para atingir um determinado objetivo. Neste trabalho, pretende-se investigar e aplicar técnicas de modelagem de redes neurais a fim de produzir um robô quadrúpede que seja capaz de realizar as seguintes ações:

* Prever as consequências de suas ações, ou seja predizer os estados futuros de seus sensores com base nas decisões tomadas.
* Comandar os motores a fim de atingir um objetivo, permanecer em equilíbrio e evitar obstáculos.

Dessa forma, este trabalho terá como foco principal investigar a utilização de controlador baseado em redes neurais para controle da inclinação do robô quadrúpede e documentar o desempenho do controlador preditivo para tal aplicação.

## 1.1. Justificativa

Reproduzir técnicas de controle baseadas em redes neurais artificiais, para projetar e aplicar um controlador inteligente, tal que seja capaz de prever o comportamento do robô quadrúpede com base no estado atual e nas escolhas tomadas pelo robô para corrigir o seu estado de forma preditiva e adaptativa, tendo como base os método de controle adaptativo por modelo interno, no qual o modelo de referência é construído por uma rede neural preditiva.

As redes neurais são poderosas ferramentas de aprendizado de máquina utilizadas para, através de aprendizado por reforço, reconhecer e aprender a representar ou classificar padrões presentes em conjuntos de dados. *(*Ricardo Fileti Marcon*. 2009).*

## 1.2. Colocação do Problema

Pretende-se projetar um controlador baseado em redes neurais para solucionar o problema descrito e avaliar o desempenho do robô ao longo de seu treinamento utilizando o microcontrolador Arduino Mega.

## 1.3. Objetivos

O trabalho está voltado para a construção de dois modelos de redes neurais, responsáveis pelo controle do robô. Um modelo capaz de prever os próximos N estados, com base no estado inicial dos sensores e um modelo capaz de prever os próximos comandos a serem dados para atingir os objetivos impostos no item 1.3.1.

Com isso, pode-se resumir o objetivo do trabalho em testar este método é analisar o desempenho do controlador em relação aos métodos convencionais.

## 1.4 Organização do Trabalho

O capítulo 2 trata-se da fundamentação teórica, abordando os métodos e teorias acerca de trabalhos anteriores envolvendo Controle Preditivo, Redes Neurais e sistemas embarcados que se aproximam do trabalho a ser desenvolvido com o robô quadrúpede, bem como o seu sistema de controle e o algoritmo a ser utilizado.

No capítulo 3, tem-se o detalhamento, de forma sequencial, acerca de todo o trabalho desenvolvido, partindo da pré-definição dos critérios a serem analisados ao longo dos testes, seguido da modelagem e controle e concluindo com a validação do desempenho do robô quadrúpede ao longo do treinamento e dos testes.

# 2 FUNDAMENTAÇÃO TEÓRICA

Para a compreensão do trabalho desenvolvido e as técnicas utilizadas no robô quadrúpede, tem-se como base os seguintes temas:

* Redes Neurais Artificiais preditivas;
* Redes Neurais Adaptativas;
* Método do Modelo Interno;
* Sistemas Embarcados;
* Condição de Persistência de Excitação;

## 2.1 Predição de Estados Baseado em Modelo de Referência

### 2.1.1 Controle Preditivo

O controle preditivo utiliza técnicas para que seja possível prever o comportamento de um determinado sistema a partir de um modelo matemático, este modelo pode ser tanto uma Rede Neural quanto um modelo matemático da saída em relação às variáveis de estado. As vantagens deste tipo de Controle é poder prever o erro de determinada a ação e tomar a melhor ação de controle a partir do erro previsto, portanto não é preciso que o erro aconteça.

No artigo ***Controle Preditivo Baseado em Modelo para Conversores Formadores de Rede com Operação*** (Rayane Aparecida Guimarães - 2019)***,***é feito o controle por chaveamento de forma preditiva para minimizar o erro em Conversores seguindo os seguinte algoritmo:

*(1) Medir a variável de controle;*

*(2) Aplicar o estado de chaveamento calculado no período de amostragem anterior;*

*(3) Estimar a variável de controle para o instante k + 1;*

*(4) Prever a variável de controle para o instante k + 2 para todos os estados de chaveamento do conversor;*

*(5) Avaliar a função custo;*

*(6) Escolher a ação de controle que minimiza a função custo o qual será aplicado no próximo período de amostragem.*

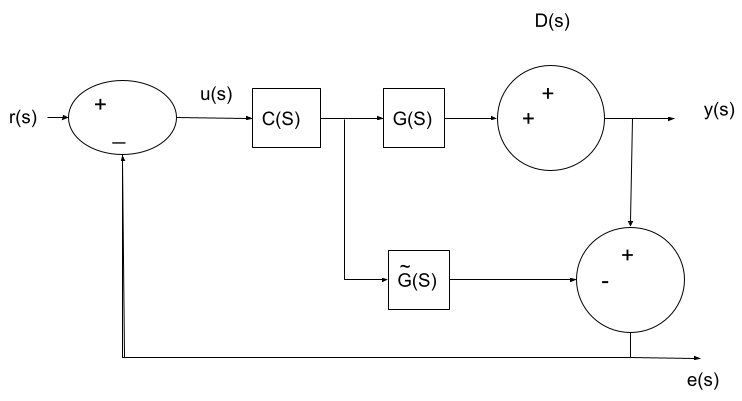
Em termos gerais, o controle preditivo baseado em modelo, MPC (*Model Predictive Control*), pode ser considerado como qualquer algoritmo que usa um modelo do sistema para prever seu comportamento futuro e selecionar a ação de controle mais apropriada com base em um critério de otimização*(KOURO et al. 2009).*

Este algoritmo associado ao método do modelo interno foi utilizado como base para o desenvolvimento da Rede Neural preditiva do robô quadrúpede.

### 2.1.1 Método do Modelo Interno

A Figura 1 mostra o controle feito através do método do Modelo Interno.

Figura 1 – Método do Modelo Interno



Fonte: Autor, baseado no vídeo Internal Model Control - IMC(Sergio Andres Castaño Giraldo, 2021)

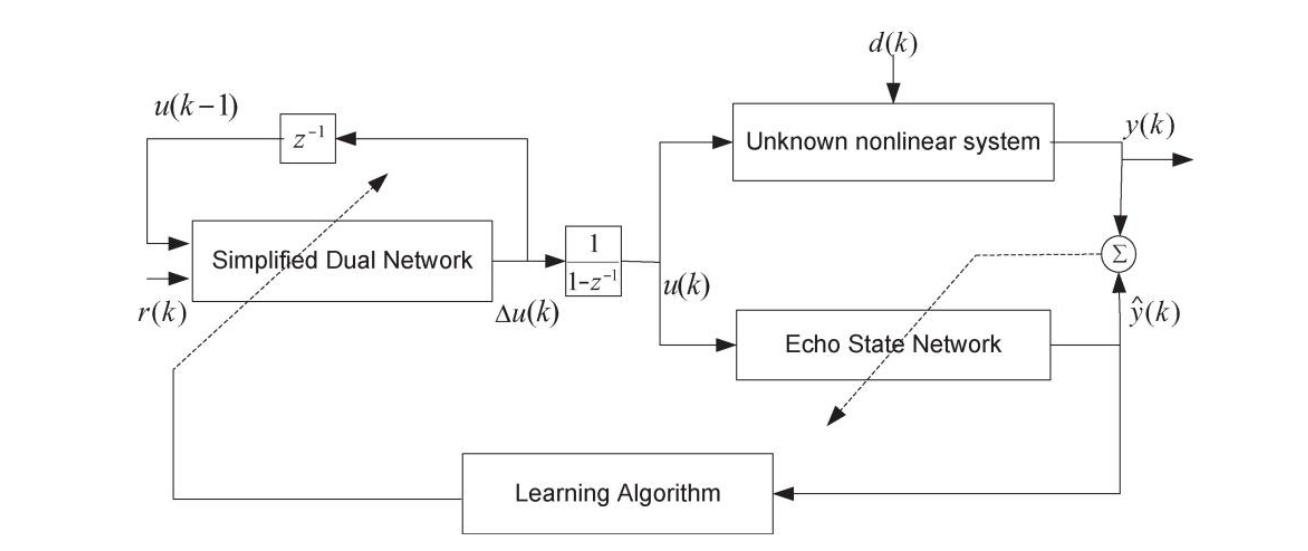
No diagrama, o controle do processo é feito através do erro entre o sinal real do sistema e o seu modelo matemático, a fim de obter um controlador tolerante a erros de modelagem. O modelo matemático é obtido a partir da resposta ao degrau do sistema em malha aberta.

No robô quadrúpede, a rede neural preditiva terá como entrada o estado atual, no instante k, e terá como saída, o estado no instante k+1. Portanto, após o treinamento, esta rede faz o papel de G̃(s), a fim de obter o erro e(s) entre o estado previsto e o estado real, que será utilizado para atualizar os parâmetros do controlador C(s), que se trata de uma rede neural adaptativa.

### 2.1.1 Redes Neurais Preditivas em sistemas desconhecidos

No Artigo “***Model Predictive Control of Unknown Nonlinear Dynamical Systems Based on Recurrent Neural Networks***”, utilizado como uma das bases para o controle, foi utilizado duas redes neurais, a rede de estado de eco (ESN) e a rede dupla simplificada (SDN). A rede ESN é utilizada para identificação do sistema e a rede SDN utilizada para otimização. A abordagem baseada em RNN proposta tem muitas propriedades desejáveis como convergência global e baixa complexidade.

Figura 2 – Modelo de Controle Preditivo Utilizando Redes Neurais.



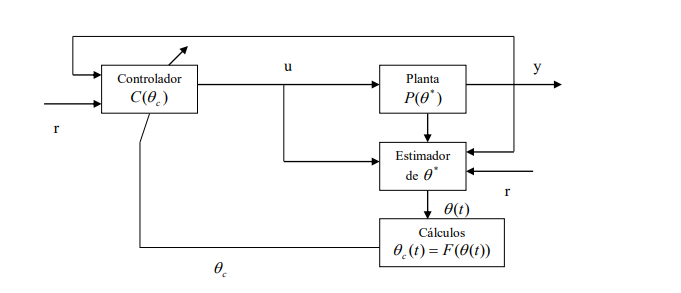
Fonte: Pan, Yunpeng; Wang, Jun.Model. **Predictive Control of Unknown Nonlinear Dynamical Systems Based on Recurrent Neural Networks**. IEEE Transactions on Industrial Electronics. IEEE, volume 59, 10.1109/TIE.2011.2169636, p. (3089 - 3101), Agosto, 2012.

**2.2 Controle Adaptativo Por Modelo de Referência**

Primeiramente, os dados colhidos com o sinal aleatório, após atender a condição de excitação persistente, explicado no item 2.3, são utilizados no algoritmo de treinamento da rede neural preditiva, que após o aprendizado comporta-se de forma que a relação matemática entre os dados de entrada e saída sejam semelhantes a relação existente no modelo do robô quadrúpede, em outras palavras a rede neural será o próprio modelo matemático do robô.

Por fim, a rede neural preditiva obtida no processo será utilizada como fonte de dados para a síntese de um controlador neural adaptativo. As duas redes trabalhando em conjunto formarão o modelo que irá fazer o controle adaptativo do robô quadrúpede após o pré-treinamento, utilizando o método de controle por “Modelo de Referência”, conforme apresenta a figura 3.

Figura 3 – Controle por Modelo de referência.



Fonte: Oliveira, Josenalde Barbosa. **Controle Adaptativo Indireto por Modelo de Referência e Estrutura Variável**. Fevereiro de 2003. Universidade Federal do Rio Grande do Norte. Disponível em https://repositorio.ufrn.br/bitstream/123456789/27034/1/josenaldeBO2003.pdf. Acesso em 25/06/2022.

O método de controle adaptativo por meio da referência de uma estrutura variável de forma indireta(Josenaide Barbosa de Oliveira, 2003) tem como objetivo desenvolver um controle do sistema por meio de um modelo matemático de forma a obter um controlador robusto e menos suscetível a erros de modelagem, isto é útil para o robô quadrúpede, visto que seu sistema poderá sofrer interferências do ambiente que, consequentemente irá produzir incertezas nos parâmetros colhidos para sintetizar o controlador, como o coeficiente de atrito do solo, a inclinação da superfície em que o robô se apresenta, a presença ou não de obstáculos e também os desgastes sofridos pelas suas partes móveis ao passar do tempo.

## 2.3 Persistência de Excitação

Ao obter os dados a serem utilizados no treinamento, deve-se atender a condição de de excitação persistente para que haja convergência do erro de rastreamento (IOANNOU; SUN, 1996; ASTROM; WITTENMARK, 2008; SASTRY; BODSON,

2011).

O método mais comum para viabilizar o cumprimento da condição de excitação persistente é adicionar ao sinal de controle um sinal senoidal de baixa amplitude e rico em frequência(Luiz Torres, 2019), e assim garantir que os dados utilizados no treinamento da rede neural que irão simular o modelo do robô quadrúpede irão representar bem o comportamento do sistema, o que é a base para o desenvolvimento de todo o trabalho.

# 3 METODOLOGIA

## 3.1 Objetivos impostos ao robô quadrúpede

O robô tem os seguintes objetivos:

I: Manter-se em equilíbrio, mesmo que hajam perturbações ;

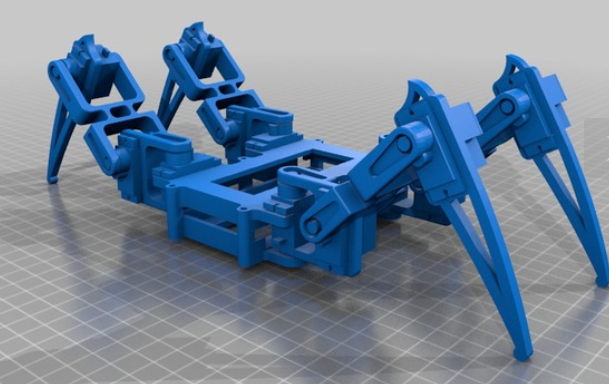
II: Minimizar as vibrações e impactos, a fim de preservar as conexões, ou seja, manter a aceleração nos três eixos o mais próximo de zero possível ;

III:Manter-se virado para o lado que não haja obstáculos, ou seja, sempre que detectar um objeto em uma distância menor do que a distância máxima DM, ele deverá virar o sensor para outra direção.

IV: Prever as consequências de suas ações, ou seja, estimar qual serão os estados resultantes dos comandos dados aos motores para os próximos N passos.

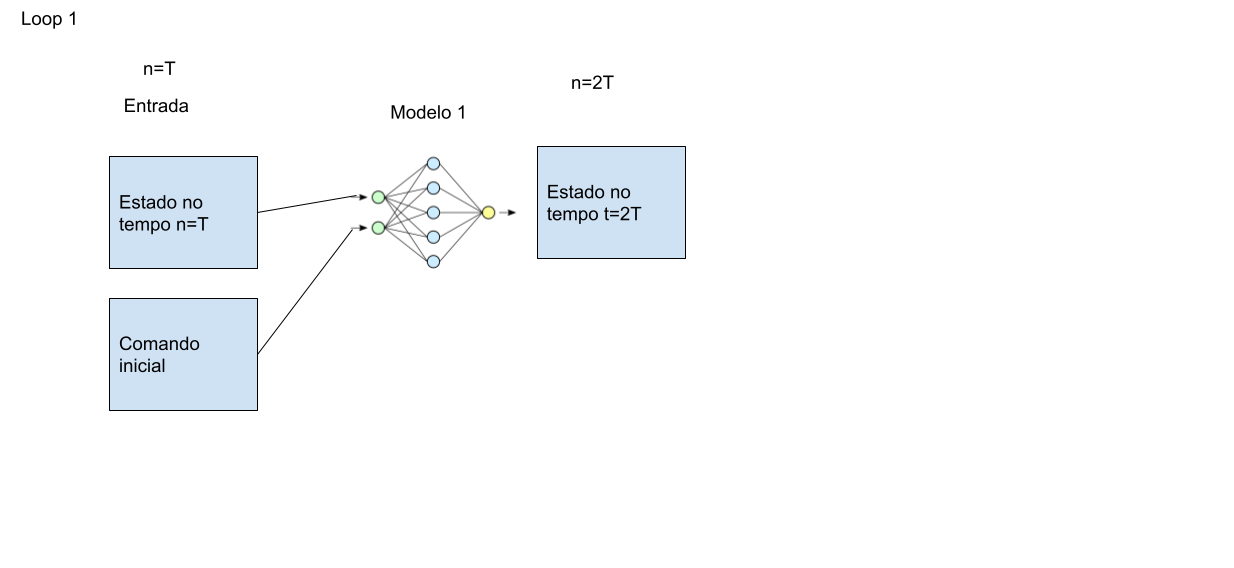
Para testar e colher os resultados da rede neural preditiva em um modelo não linear, foi utilizado um microcontrolador em um sistema embarcado baseado em um robô quadrúpede:

Figura 4: Spider robot, quad robot, quadrupede



Fonte: Monhambe. **Spider robot, quad robot, quadruped**. Thingiverse. Fevereiro de 2018. Disponível em https://www.thingiverse.com/thing:2796820/files. Acesso em 24/06/2022

Figura 5: Diagrama, loop 1



Fonte: Autor.

A aprendizagem do modelo 1 será feita com base no quão certo ele previu o estado, utilizando um erro médio quadrático entre o vetor E[T], estado no tempo T, e o vetor Eh[T], estado previsto para o tempo T. E a aprendizagem do modelo 2 será feita com base no quanto o estado resultante do comando aproximou-se do estado desejado, em que o erro será baseado em uma pontuação dada de acordo com os objetivos que devem ser atingidos. O resultado deste processo será um vetor contendo a trajetória dos estados do sistema, este vetor é dinâmico, e seus valores são constantemente alterados e corrigidos pelos modelos trabalhando em conjunto.

Erro do modelo 1:

Tabela 1 – Vetor Resultante

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vetor resultante | | | | | |
| tempo(n) | n=T | n=2.T | n=3T | ... | n=N.T |
| Estados | E1[T] | E1[2T] | E1[3T] | ... | E1[NT] |
| E2[T] | E2[2T] | E2[3T] | ... | E2[NT] |
| E3[T] | E3[2T] | E3(T) | ... | E3(T) |
| E3[T] | E3[2T] | E3[3T] | ... | E3[NT] |
| Comandos | C1[T] | C1[2T] | C1[3T] | ... | C1[NT] |
| C2[T] | C2[2T] | C2[3T] | ... | C2[NT] |
| C3[T] | C3[T] | C3[3T] | ... | C3[NT] |

Fonte: Autor.

Modelo 1:

O modelo 1 terá como entrada os estados atuais do robô quadrúpede e os comandos dados no tempo atual e, terá como saída o próximo estado do robô. Considerando como entrada o estado e comandos medidos no tempo n=T, a saída do sistema será o estado no tempo n=(2T), onde T é o intervalo de medições feito pelo microcontrolador.

Modelo 2:

O modelo 2 irá determinar os comandos a serem dados para robô no tempo n=T, de acordo com os estados previsto para o tempo n=(2T) e o estado atual, considerando que o primeiro comando dado será um vetor nulo.

Portanto, o modelo de controle preditivo pode ser simplificado da seguinte maneira:

Os estados e comandos do sistema são:

Tabela 2 – Variáveis do Sistema

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variáveis De Estado | | | | |
| Sensores | | | | |
| Nome | Descrição | Range | Tipo | unidade |
| EM1 | Ângulo do Motor 1 | 0-180 | int | graus |
| EM2 | Ângulo do Motor 2 | 0-180 | int | graus |
| EM3 | Ângulo do Motor 3 | 0-180 | int | graus |
| EM4 | Ângulo do Motor 4 | 0-180 | int | graus |
| EM5 | Ângulo do Motor 5 | 0-180 | int | graus |
| EM6 | Ângulo do Motor 6 | 0-180 | int | graus |
| EM7 | Ângulo do Motor 7 | 0-180 | int | graus |
| EM8 | Ângulo do Motor 8 | 0-180 | int | graus |
| EM9 | Ângulo do Motor 9 | 0-180 | int | graus |
| EM10 | Ângulo do Motor 10 | 0-180 | int | graus |
| EM11 | Ângulo do Motor 11 | 0-180 | int | graus |
| EM12 | Ângulo do Motor 12 | 0-180 | int | graus |
| EIX | Inclinação do robô no eixo x | 0-180 | int | graus |
| EIY | Inclinação do robô no eixo y | 0-180 | int | graus |
| EIZ | Inclinação do robô no eixo z | 0-180 | int | graus |
| EAX | Aceleração do robô no eixo x | 0-1 | int | - |
| EAY | Aceleração do robô no eixo y | 0-1 | int | - |
| EAZ | Aceleração do robô no eixo z | 0-1 | int | - |
| EV | Vibração do robô, medido indiretamente através da variação de EIX e EIY | 0-180 | float | graus/período |
| ED | Distância medida pelo sensor ultrassônico | 0-10 | float | metros |
| Comandos | | | | |
| Nome | Descrição | Range | Tipo | unidade |
| CM1 | comanda motor M1 | 0-180 | int | graus |
| CM2 | comanda motor M2 | 0-180 | int | graus |
| CM3 | comanda motor M3 | 0-180 | int | graus |
| CM4 | comanda motor M4 | 0-180 | int | graus |
| CM5 | comanda motor M5 | 0-180 | int | graus |
| CM6 | comanda motor M6 | 0-180 | int | graus |
| CM7 | comanda motor M7 | 0-180 | int | graus |
| CM8 | comanda motor M8 | 0-180 | int | graus |
| CM9 | comanda motor M9 | 0-180 | int | graus |
| CM10 | comanda motor M10 | 0-180 | int | graus |
| CM11 | comanda motor M11 | 0-180 | int | graus |
| CM12 | comanda motor M12 | 0-180 | int | graus |
| Critérios de desempenho do controle utilizados como treinamento do modelo 2(Modelo Preditivo) | | | | |
| Nome | Descrição | | Tipo | unidade |
| D\_TS | Tempo de subida medido em número de períodos | | float | T |
| D\_TA | Tempo de acomodação medido em número de períodos | | float | s |
| D\_E | Erro em regime permanente | | float | % |
| D\_EC | Esforço de controle | | float | % |

Fonte: Autor.

## 3.1 CRONOGRAMA

Tabela 3 – Cronograma

|  |  |  |
| --- | --- | --- |
| **TAREFA** | **INÍCIO** | **TÉRMINO** |
| **TAREFAS PRELIMINARES** | Maio | Junho |
| Aquisição dos Materiais | 1/5/22 | 15/5/22 |
| Montagem do robô | 15/5/22 | 29/5/22 |
| Montagem do circuito | 30/5/22 | 13/6/22 |
| Teste e validação do sistema | 14/6/22 | 28/6/22 |
| Estudo e Modelagem do Sistema | 14/6/22 | 28/6/22 |
| **ESTUDO E MODELAGEM DO SISTEMA** | Junho | Julho |
| Coleta de Dados do Comportamento do Sistema frente a diferentes entradas | 29/6/22 | 9/7/22 |
| Definição de um modelo matemático do Sistema | 10/7/22 | 20/7/22 |
| Estudo do modelo matemático utilizando ferramentas computacionais | 20/7/22 | 30/7/22 |
| **MODELAGEM DAS REDES NEURAIS** | Julho | Setembro |
| Descrição do sistema de controle e o algoritmo a ser utilizado | 30/7/22 | 6/8/22 |
| Definição dos Critérios de Desempenho | 7/8/22 | 14/8/22 |
| Definição dos modelos de Rede Neural | 15/8/22 | 22/8/22 |
| Algoritmo de propagação e retropropagação | 23/8/22 | 30/8/22 |
| Treinamento e Testes das Redes Neurais em Ambiente Simulado | 31/8/22 | 7/9/22 |
| **APLICAÇÃO DAS REDES NEURAIS NO ROBÔ QUADRÚPEDE** | Setembro | Outubro |
| Aplicação das Redes Neurais pré treinadas no robô quadrúpede | 8/9/22 | 15/9/22 |
| Análise dos Critérios de Desempenho Na Rede Pré Treinada | 14/9/22 | 21/9/22 |
| Continuidade do Treinamento no Ambiente Real | 21/9/22 | 28/9/22 |
| Análise de Desempenho e Validação do Sistema | 28/9/22 | 5/10/22 |
| **Otimização Das Redes Neurais** | Outubro | Outubro |
| Técnicas para aperfeiçoamento das Redes Neurais | 6/10/22 | 13/10/22 |
| Interpretação e conclusão dos resultados | 13/10/22 | 20/10/22 |
| Registro de Desempenho | 21/10/22 | 28/10/22 |

Fonte: Autor.

## 3.2 RECURSOS

### 3.2.1 Materiais

Os componentes utilizados para este projeto são divididos em atuadores, sensores e outros, como descritos na tabela a seguir:

Tabela 4 – Lista de Materias

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lista de Materiais | | | | |
| Atuadores | | | | |
| Número | Componente | Modelo | Quantidade | Função |
| 1 | Servo Motor |  | 12 | Controle do ângulo das patas |
| Sensores | | | | |
| Número | Componente | Modelo | Quantidade | Função |
| 2 | Sensor Ultrassônico |  | 1 | Medição da distância linear de objetos no caminho |
| 3 | Módulo Sensor de Inclinação | MPU6050 | 1 | Medir a Inclinação e aceleração em x, y e z do sistema |
| Auxiliares | | | | |
| Número | Componente | Modelo | Quantidade | Função |
| 4 | Arduino Mega | - | 1 | Utilizado para controle do Sistema |
| 5 | Jumpers | - | 28 | Utilizado para conexão dos materiais |
| 6 | Bateria | - | 1 | Para alimentar o sistema e diminuir as variações de tensão da fonte |
| 7 | Fonte 5v-3A | - | 1 | Alimentar o sistema |
| 8 | Filamento de impressão 3D | - | 200g | Para impressão das peças |
| 9 | Espaguete termo retrátil | - | 1M | Para isolar o circuito |
| 10 | Estanho para Solda | - | 50g | Para soldar os componentes |
|  |  |  |  |  |

Fonte: Autor.

Para fazer o controle, foi definido a utilização do Arduino Mega, pela praticidade, custo e quantidade de saídas digitais pwm. A fonte a ser utilizada foi dimensionada pelo consumo de corrente do circuito. Utilizando uma fonte de bancada e um multímetro, foi aplicada uma tensão de 5V no circuito e os motores foram comandados para ângulos aleatórios a fim de definir qual o pico de consumo do sistema. O resultado foi que o circuito teve um consumo de 3A.

### 3.2.2 Ferramentas computacionais

Para auxiliar nos testes e modelagem, tanto do sistema a ser controlado quanto dos modelos de redes neurais, foram utilizados as seguintes ferramentas:

* Jupyter Notebook: Utilizado para desenvolvimento das redes neurais a serem utilizadas no projeto
* Visual Studio Community: Programação do Arduino e simulação das Redes Neurais
* Mathworks MATLAB R2020a: Utilizado para simular o modelo matemático e obter os critérios de desempenho idealizados para o projeto
* Tinkercad: Utilizado para testar o código em c++ a ser utilizado no arduino mega

### 3.2.3 Equipamentos Utilizados

Para montagem e testes, foram utilizados os seguintes equipamentos:

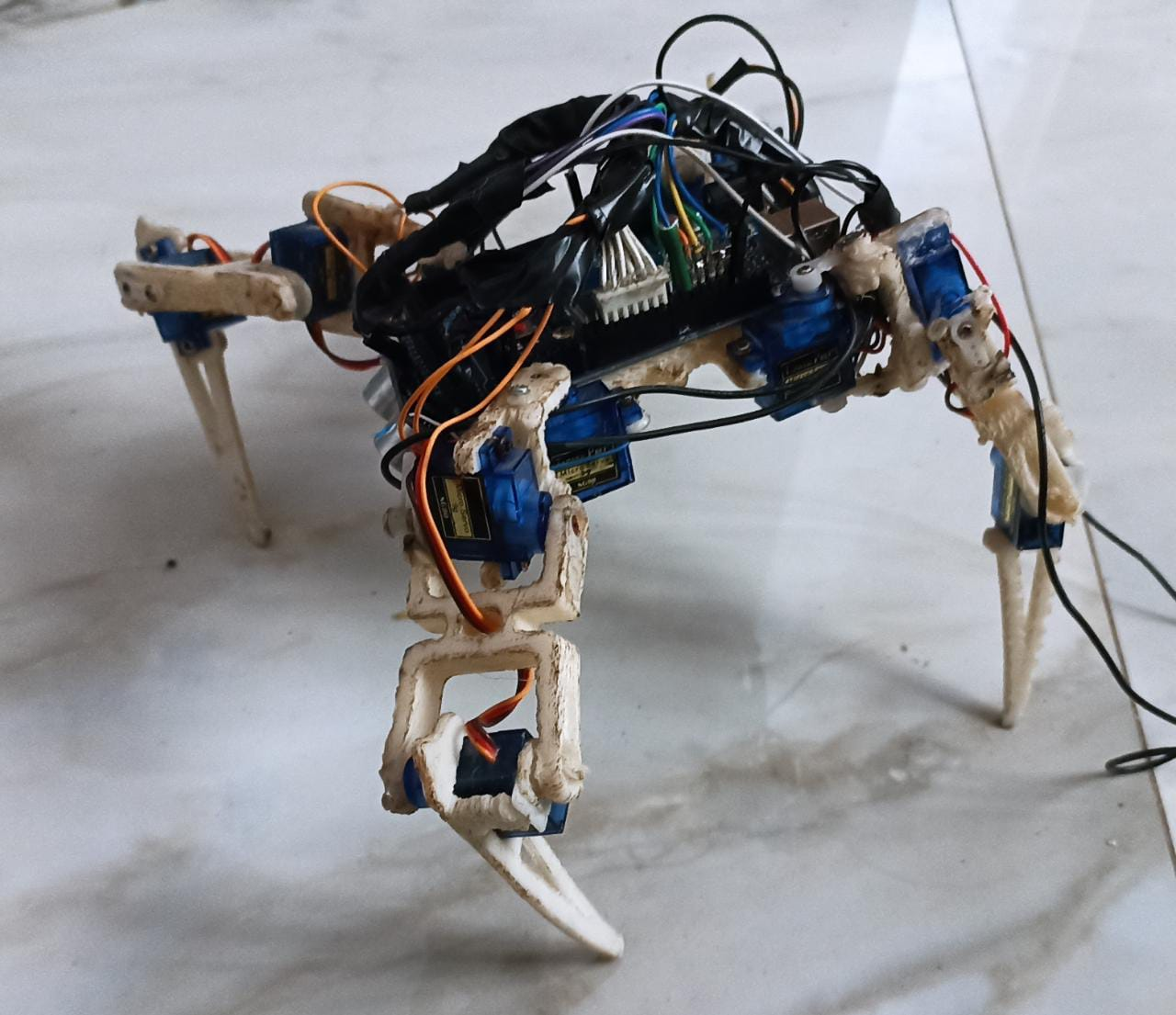
* Multímetros;
* Osciloscópio;
* Ferro de solda;
* Computador;
* Fonte de bancada;

## 3.3 MONTAGEM DO SISTEMA

### 3.3.1 Modelo em 3D

O modelo em 3D foi obtido no site thingiverse, criado pelo usuário Monhambe em 16 de Fevereiro de 2018. Foi utilizada uma impressora 3D para obter as peças do sistema a ser controlado, na figura 2, que foi montado utilizando parafusos para fixar os servos motores. Por fim, os componentes foram conectados ao controlador com o auxílio de jumpers e de solda com estanho

Figura 9: Robô Quadrúpede Montado



Fonte: Autor.

Após a Montagem, foram feitos testes de continuidade e a alimentação do sistema, para testar os componentes e validar a montagem do sistema.

## 3.4 Estudo e Modelagem do Sistema Com Rede Neural Preditiva

Para modelar um sistema é preciso descrever matematicamente o seu comportamento de forma a prever os estados de suas variáveis em função de perturbações e da própria dinâmica existente no sistema, isto é, cria um bloco capaz de relacionar de forma aproximada uma entrada e uma saída.

Para o robô quadrúpede, pode-se considerar como entrada, os estados dos sensores e comandos a serem dados aos atuadores. Os estados são os ângulos atuais dos motores, a inclinação nos três eixos(x,y e z), a aceleração nos três eixos, e a distância medida pelo sensor ultrassônico. Os comandos dados são os ângulos inseridos na função Servo.write(), da biblioteca “Servo.h”, que determina para qual ângulo o motor deverá se mover.

Como saída, pode-se considerar os estados resultantes, ou seja, a inclinação, a aceleração e a distância da próxima leitura do sistema.

A técnica de modelagem utilizada é descrita no fluxograma a seguir:.

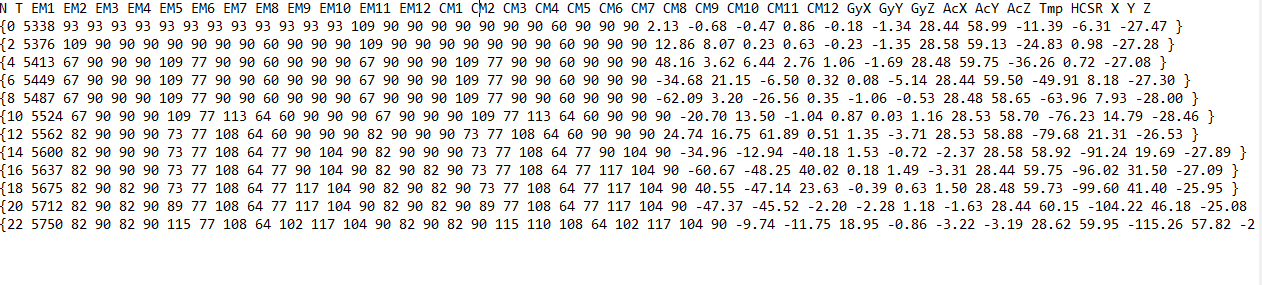
FIGURA 10: Fluxograma do Sistema Desejado



Com esta técnica, pretende-se obter um modelo capaz de representar matematicamente o comportamento do sistema, prevendo os estados futuros com base nas variáveis de entrada do sistema com o menor erro possível. O modelo preditivo resultante foi a base para a construção da rede neural que controlará o sistema, com base no erro previsto pelo modelo obtido neste método.

## 3.4.1 Obtenção dos dados de treinamento

Foi desenvolvido um algoritmo no robô quadrúpede para coletar dados, o qual gera comandos aleatórios nos atuadores (motores de passo) a fim de assegurar o cumprimento da condição de excitação persistente mencionada na seção 2.3. A cada intervalo de tempo, inicialmente definido como 20ms, o valor é transmitido via serial para que possa ser registrado pelo computador. Os valores são separados por meio da string “\t”, que é interpretada visualmente como um espaço no editor de texto.



Os dados coletados a partir do algoritmo no robô quadrúpede foram organizados de forma automática em um conjunto de treinamento para uma rede neural preditiva. Cada amostra do conjunto de treinamento seria composta por uma sequência de valores gerados pelos atuadores em um determinado intervalo de tempo, juntamente com o valor correspondente da saída desejada (inclinação do robô nos eixos x e y). Esses dados são usados para treinar a rede neural a fim de prever a saída desejada com base nos valores dos atuadores.

Nesse caso, os dados coletados pelo algoritmo no robô quadrúpede são utilizados para treinar uma rede neural que servirá como modelo matemático do robô. A rede neural é capaz de simular o comportamento do robô em resposta a diferentes entradas dos atuadores.

Para isso, os dados coletados são pré-processados e organizados em um formato adequado para treinamento da rede neural. Foram utilizadas técnicas como normalização e padronização dos dados para melhorar o desempenho do modelo. Com os dados organizados, a rede neural faz uma busca no arquivo “dados.txt” e utiliza-os para aprender o comportamento do robô no ambiente atual em resposta aos comandos aleatórios utilizados.

Após o treinamento, a rede neural pode ser utilizada para fazer predições do comportamento do robô com base em diferentes entradas dos atuadores. Essas predições foram comparadas com os dados reais do robô para avaliar a qualidade do modelo. O modelo pôde então ser refinado e aprimorado por meio de ajustes na arquitetura da rede neural e na seleção de hiperparâmetros até que seja capaz de simular com precisão o comportamento do robô quadrúpede.

* + **3.4.2 Construção da rede preditiva**

A rede preditiva foi criada através da definição de uma classe denominada como “redeNeuralPreditiva” que herda as características da classe “Module” importada da biblioteca torch em python. A rede Neural possui os atributos “tamanhoEntrada” e “tamanhoSaída”, que são utilizados para determinar respectivamente o número de neurônios da camada de entrada da rede e o número de neurônios da camada de saída.

Em relação às camadas da rede neural, foram utilizadas principalmente camadas densas (fully connected) com ativações ReLU (Rectified Linear Unit), que são comumente utilizadas em redes neurais para aprendizado profundo. A função ReLU define a saída da camada como o máximo entre zero e a soma ponderada das entradas, o que introduz não-linearidade e ajuda a rede neural a aprender relações mais complexas entre os dados.

Além da utilização da função de ativação ReLU, foram incorporadas camadas com ativações RELU6 e sigmoide na rede neural. A função RELU6 é uma variação da função ReLU que limita o valor máximo de saída em 6, o que pode ser útil para prevenir problemas de explosão do gradiente durante o treinamento. Já a função sigmoide é conhecida por produzir saídas entre 0 e 1, sendo especialmente apropriada para o problema em questão, já que os valores utilizados foram normalizados em uma escala entre 0 e 1.

A função de ativação RELU6 foi utilizada na camada de entrada e nas camadas internas da rede, enquanto a função sigmoide foi utilizada na saída da rede neural.

Vou colocar aqui uma imagem do gráfico e da função de ativação ReLU6 e sigmoide.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

* + **3.4.3 Instanciando a rede preditiva**

Texto

Descrição gerada automaticamente

* + **3.4.4 Algoritmo de treinamento da Rede Neural Preditiva**

Para o treinamento da rede neural, foi utilizada a função de perda MSE (Mean Squared Error), que calcula a média do quadrado da diferença entre as previsões da rede neural e os dados reais do robô quadrúpede. A função MSE é comumente usada em problemas de regressão, nos quais o objetivo é prever um valor numérico contínuo, como a posição ou a velocidade do robô, por este motivo foi determinada como adequada para o projeto em questão.

O otimizador utilizado durante o treinamento foi o do gradiente estocástico (SGD) que adapta as taxas de aprendizagem dos pesos de cada camada da rede neural de acordo com as estatísticas de primeira e segunda ordem das derivadas parciais da função de perda. Ele atualiza os pesos da rede em cada iteração do treinamento, com base no gradiente da função de perda em relação aos pesos e em uma taxa de aprendizado fixa.

* + - 3.4.4.1 Parâmetros de aprendizado e dados de treinamento

Texto

Descrição gerada automaticamente com confiança média

* + - 3.4.4.2 Loop de treinamento e condição de parada

Inicialmente, utilizou-se uma taxa de aprendizado alta para estimular a ampla exploração do espaço de parâmetros da rede neural, evitando que a rede neural fique presa em mínimos locais não otimizados durante o treinamento e permitindo uma busca mais abrangente no espaço de soluções. No entanto, sempre que o algoritmo detecta que a rede neural não está mais evoluindo, ou seja, quando a melhora no desempenho é insignificante em relação à taxa de aprendizado atual, o treinamento é interrompido para realizar ajustes neste hiperparâmetro e permitir que a rede neural retome o aprendizado de forma mais eficiente. Esse processo de ajuste da taxa de aprendizado é repetido em um loop até que ocorra a convergência do erro médio quadrático.

Uma imagem contendo Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

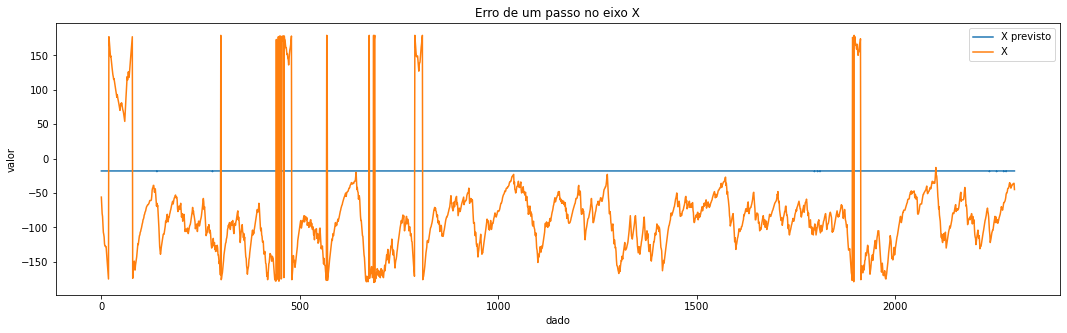
Descrição gerada automaticamente

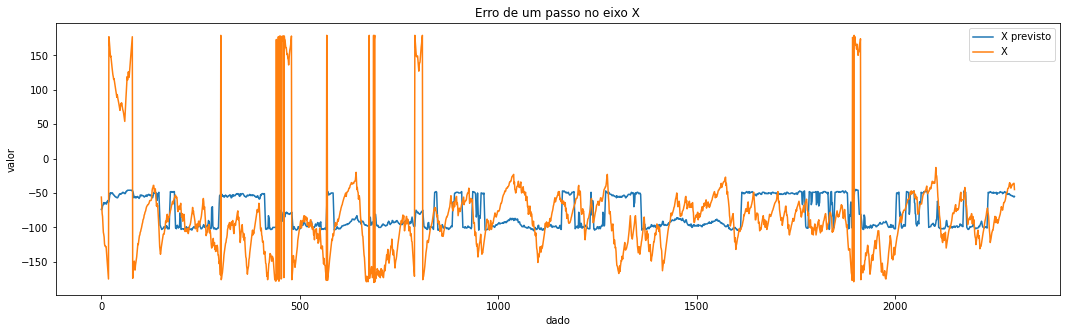
Texto

Descrição gerada automaticamente

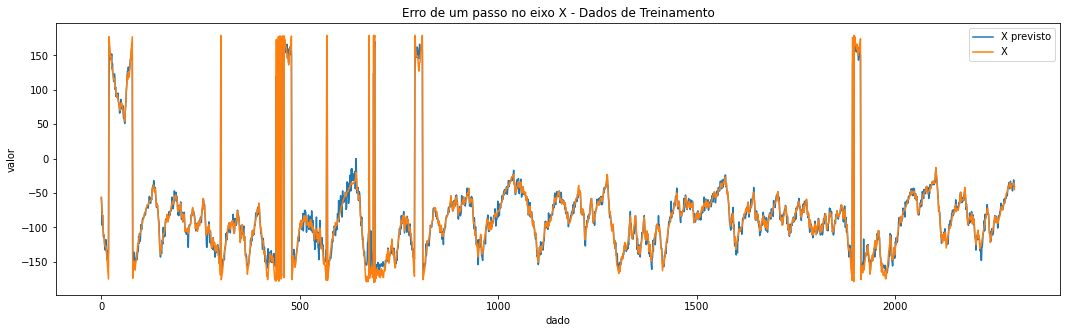
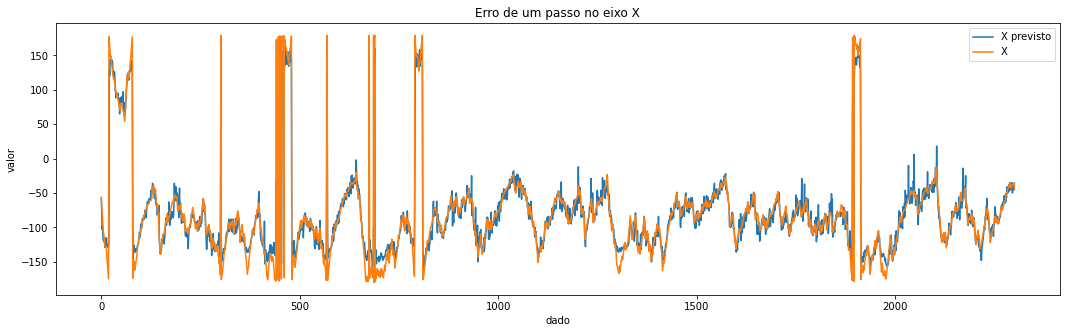
* + - 3.4.4.3 Resultados do pré-treinamento

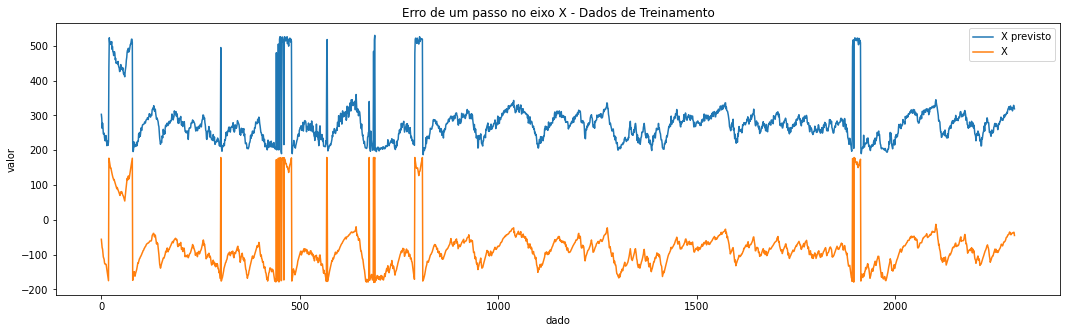
Evolução no eixo X:



Gráfico, Histograma

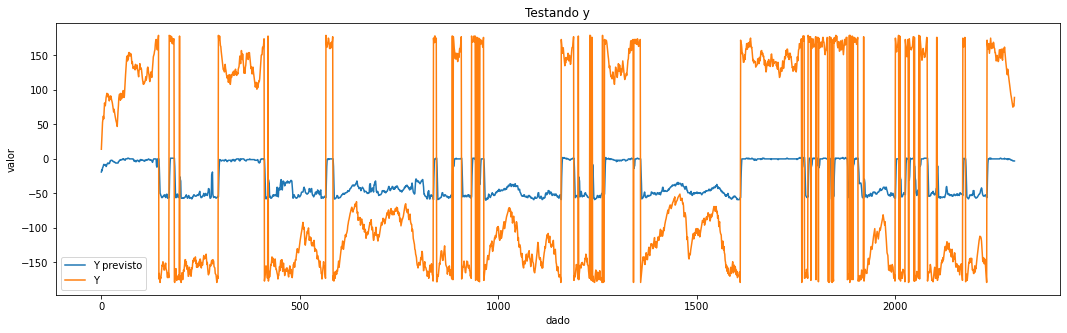
Descrição gerada automaticamente Gráfico

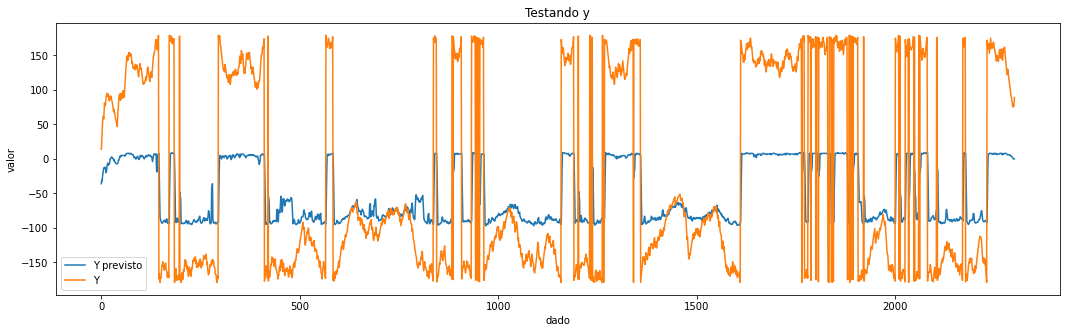
Descrição gerada automaticamente

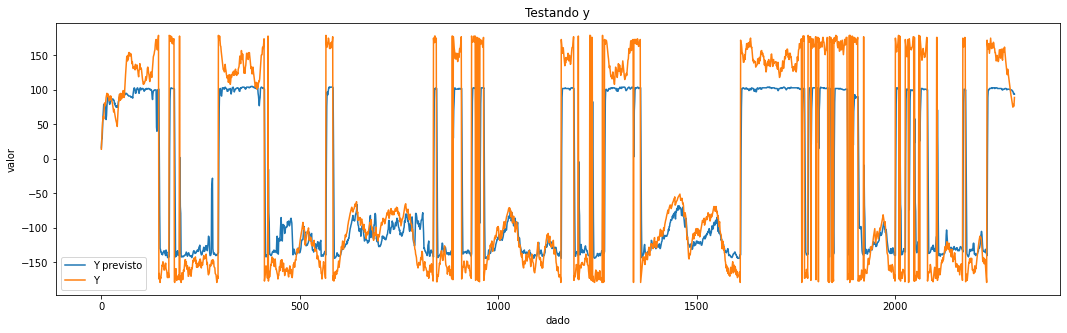


Evolução no eixo y:

Gráfico, Histograma

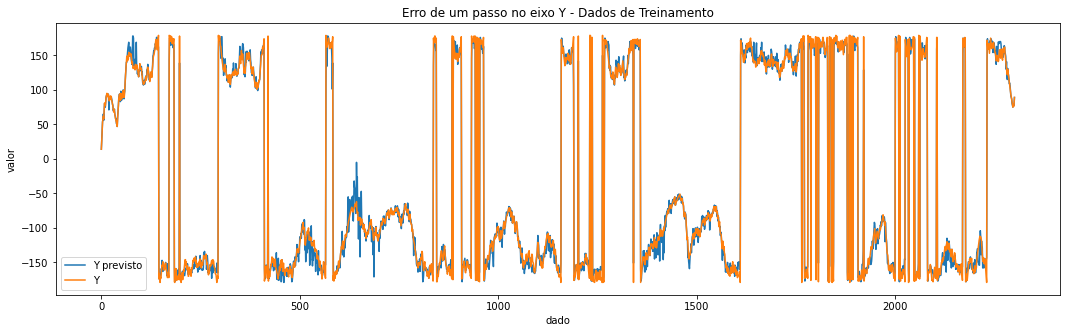
Descrição gerada automaticamente

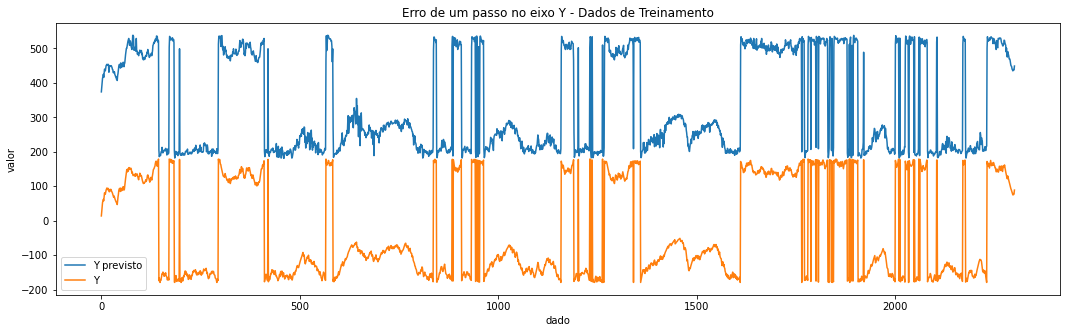




Gráfico, Histograma

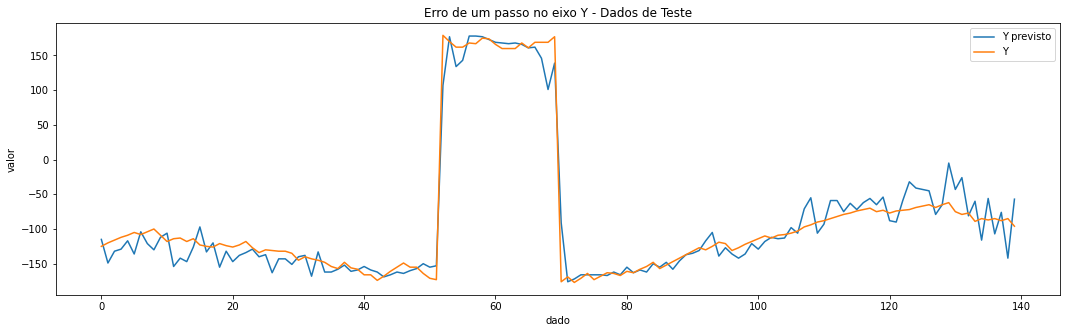
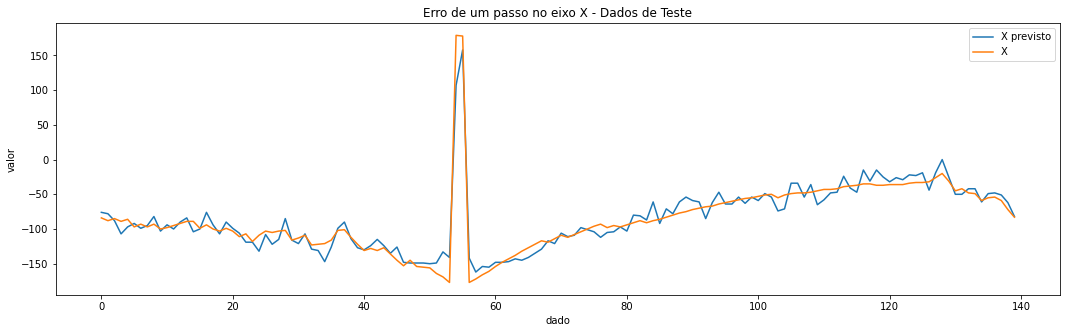
Descrição gerada automaticamente



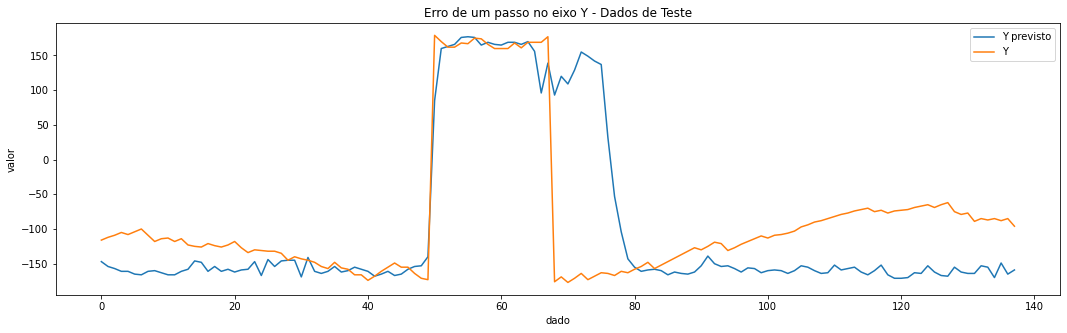
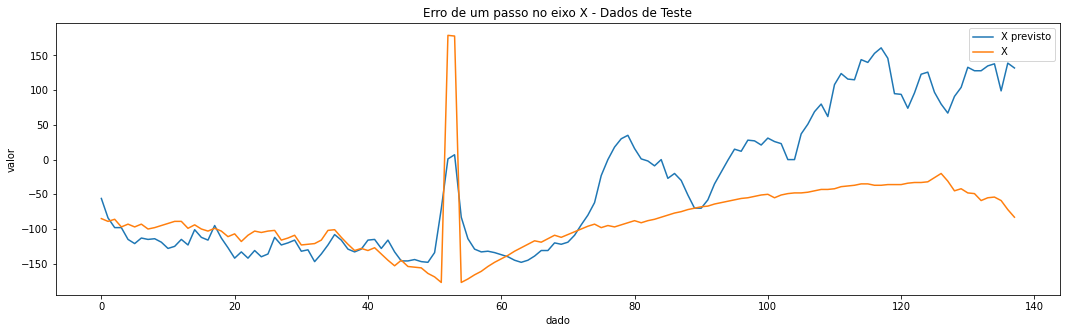


Vou colocar aqui os gráficos mostrando a evolução do treinamento

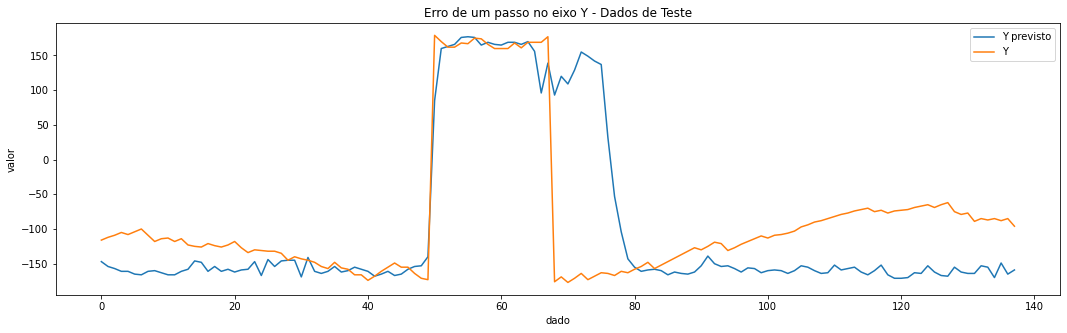
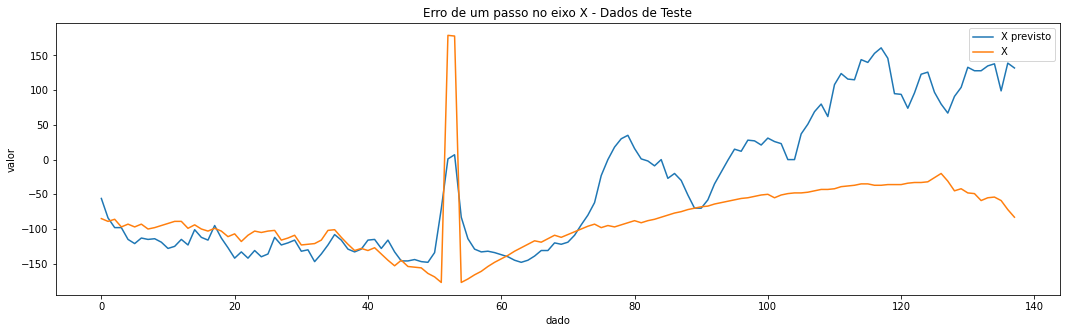
* + - 3.4.4.4 Teste de previsão de 1 passo à frente



* + - 3.4.4.5 Teste de previsão de 2 passos à frente



* + - 3.4.4.6 Teste de previsão de 3 passos à frente

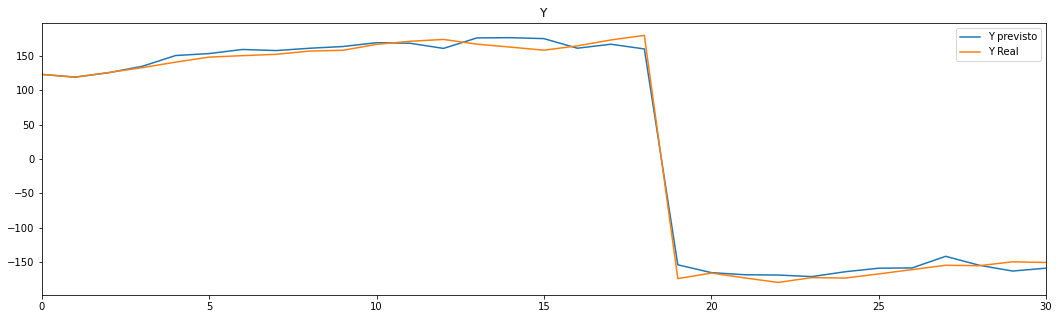
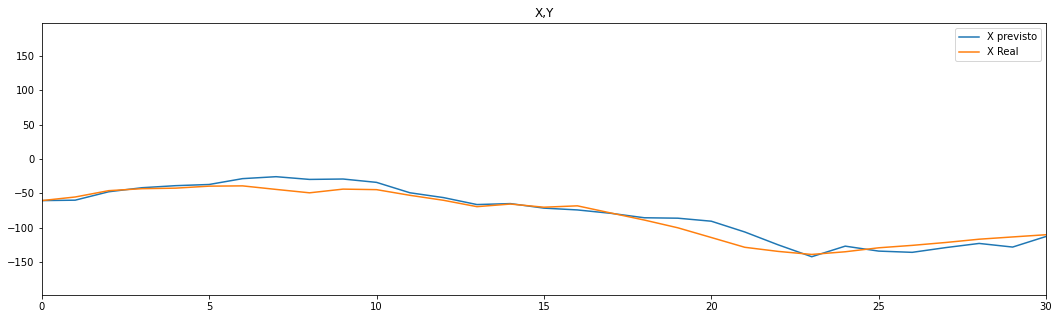
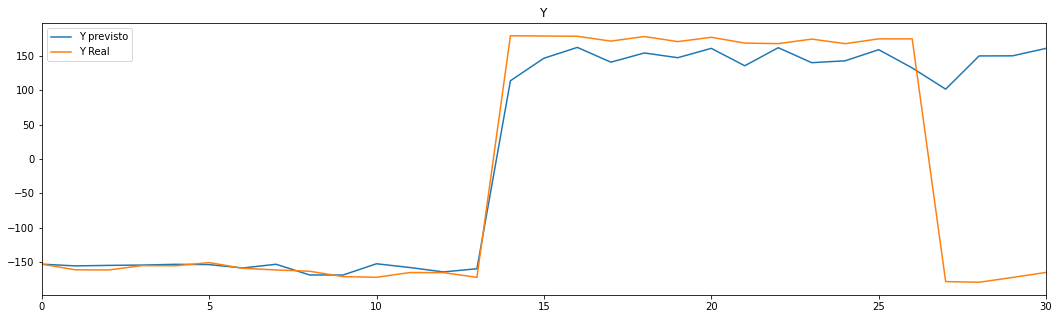
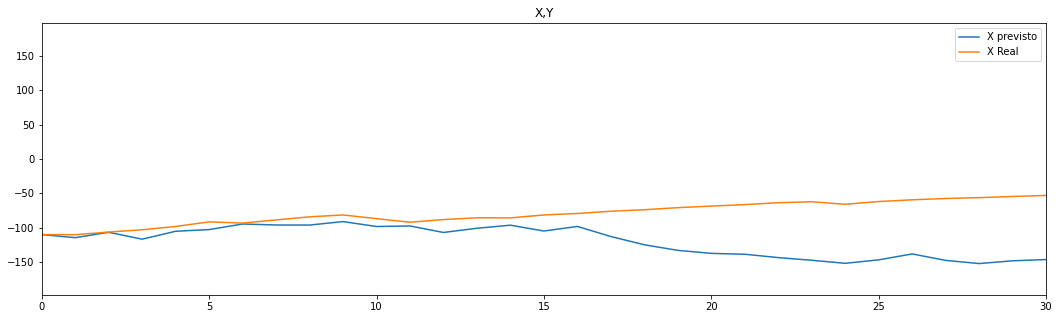
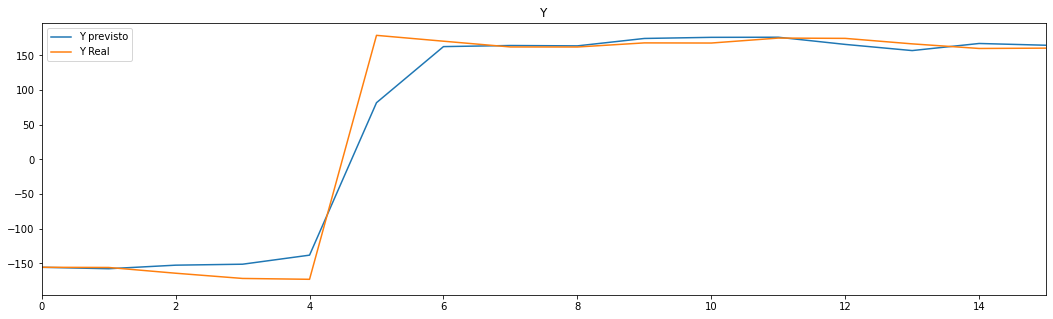
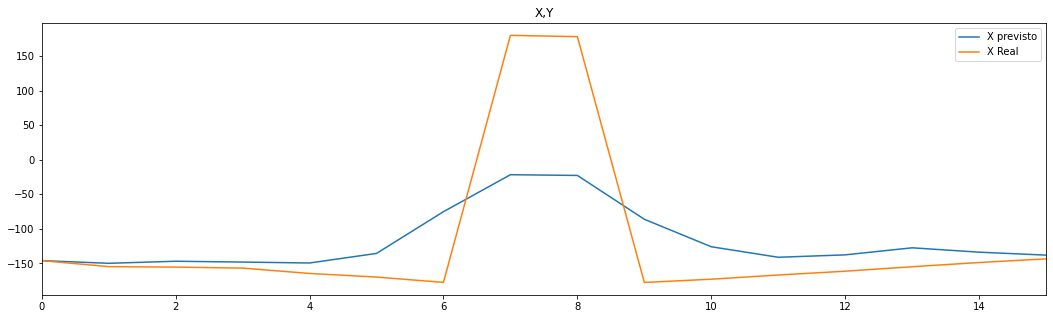
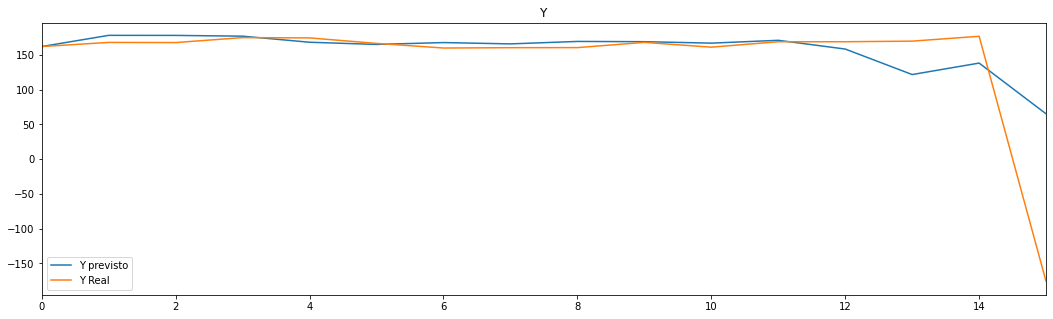
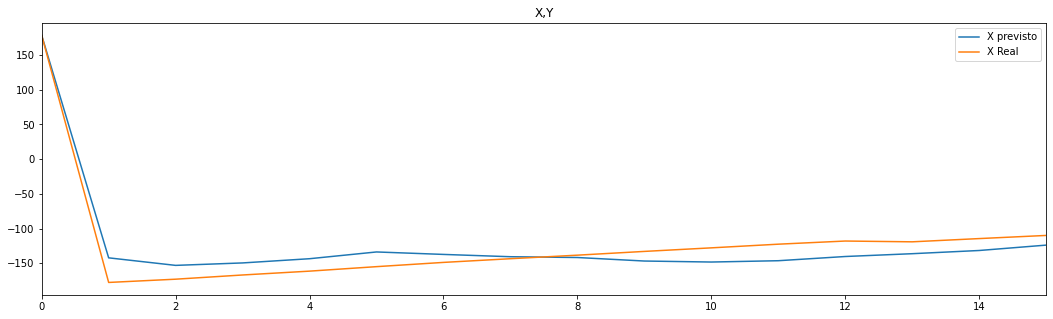
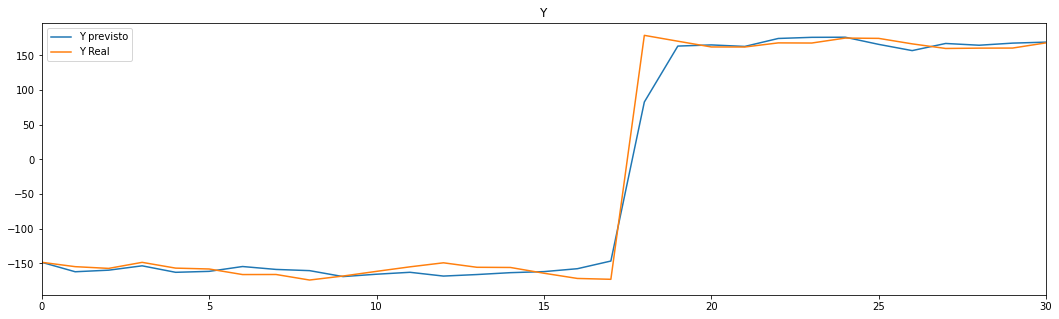
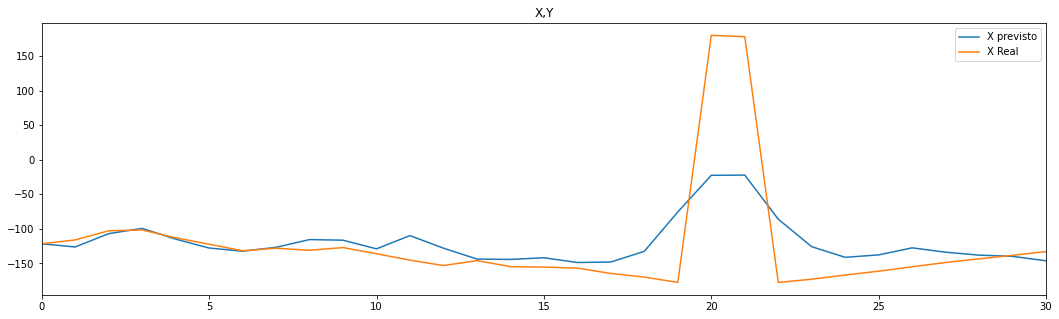
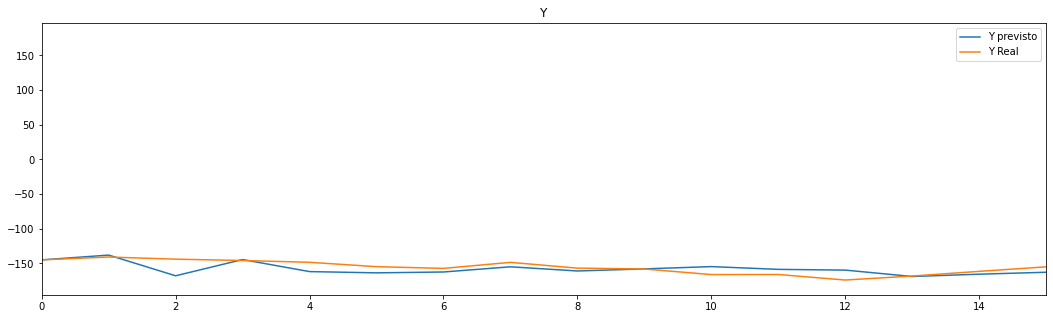
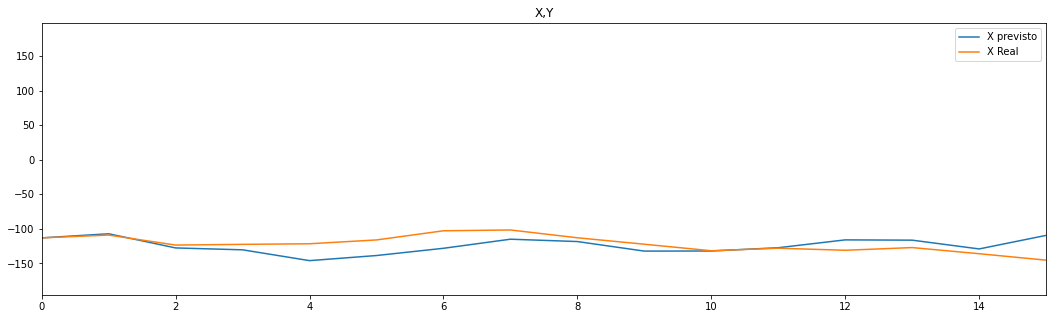
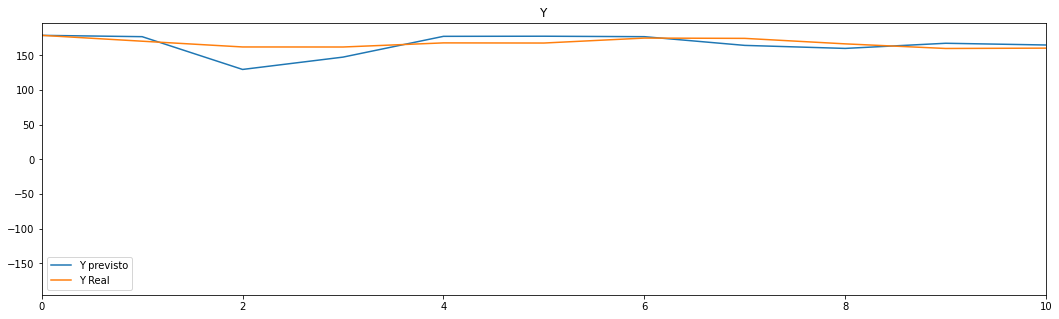
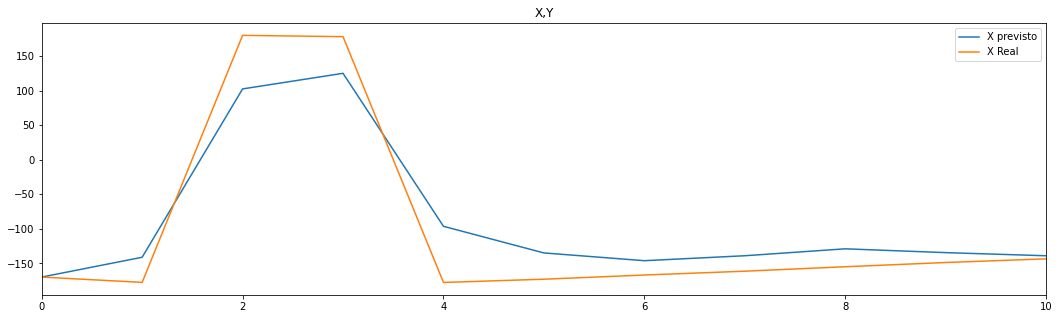
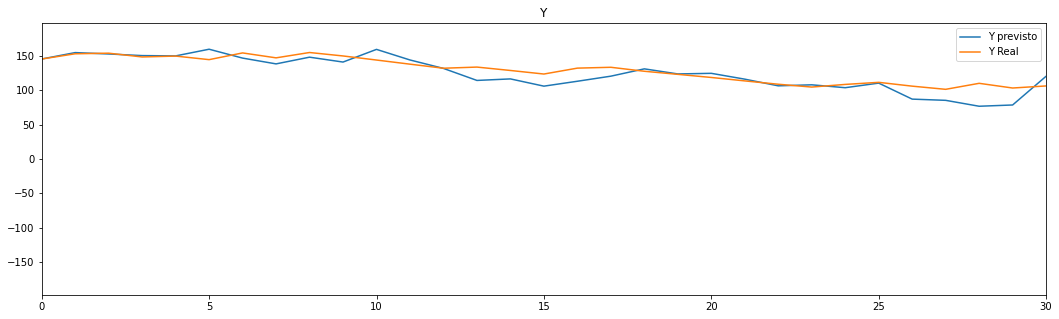
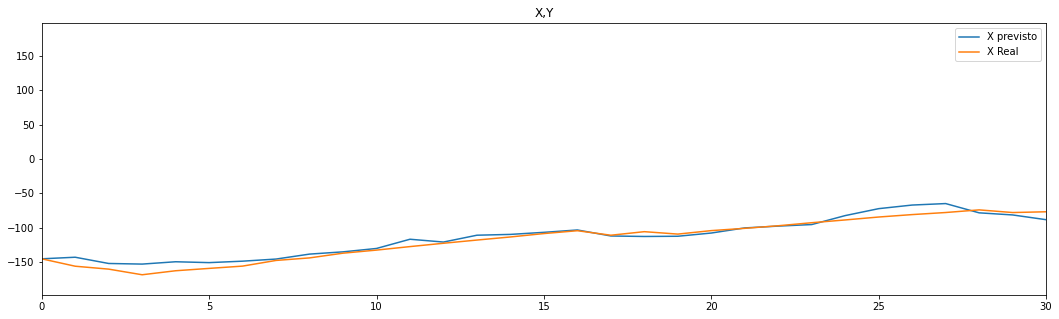


* + - 3.4.4.7 Teste de predição Livre

Gráfico, Gráfico de linhas

Descrição gerada automaticamenteUma imagem contendo Gráfico de linhas

Descrição gerada automaticamente



## 3.5 Rede Neural de Controle para Robô Quadrúpede: Ação Proativa

A combinação de uma rede neural preditiva e uma rede neural controladora permite a criação de um sistema inteligente capaz de tomar decisões eficientes e adaptativas em tempo real, a fim de evitar potenciais problemas e otimizar o desempenho do robô. Nesta seção, é apresentada a rede neural de controle implementada, destacando sua arquitetura, treinamento e integração com a rede neural preditiva para a obtenção de resultados promissores.

## 3.4.1 Princípio de Funcionamento do Controlador

A rede neural de controle implementada no contexto deste trabalho desempenha um papel fundamental na tomada de decisões proativas e na geração de ações corretivas antecipadas para o robô quadrúpede. Essa rede neural recebe como entrada os dados previstos pela rede neural preditiva, que fornece uma estimativa do comportamento futuro do robô com base em informações passadas e atuais. Em seguida, a rede neural de controle processa esses dados previstos e produz uma ação de controle apropriada para evitar problemas ou melhorar o desempenho do robô.

## 3.4.2 Estrutura da Rede Neural do Controlador

## 3.4.3 Algoritmo de treinamento

O treinamento do robô quadrúpede foi realizado por meio de um sistema de recompensa, no qual a rede neural de controle aprende a gerar ações que levem o robô a se aproximar do objetivo desejado, que consiste em manter-se em equilíbrio com ângulos X e Y igual a zero. Esse sistema de recompensa desempenha um papel crucial no processo de treinamento, permitindo que o robô aprenda a tomar decisões proativas para manter seu equilíbrio.

Inicialmente, foi definida uma função de recompensa, na qual foram atribuídos valores de recompensa ou penalidade com base nas ações tomadas pelo robô e na proximidade do estado atual em relação ao estado de equilíbrio desejado. Durante o treinamento, o robô quadrúpede executou ações com base nas saídas da rede neural de controle, e os ângulos X e Y foram medidos e comparados com o estado de equilíbrio desejado.

A função de recompensa foi então aplicada, atribuindo valores de recompensa positivos quando o robô se aproximava do equilíbrio e penalidades quando ele se afastava. Esses valores de recompensa ou penalidade foram utilizados para atualizar os pesos sinápticos da rede neural de controle. Isso foi feito por meio de algoritmos de aprendizado reforçado, como o algoritmo de gradiente descendente, no qual os gradientes da função de recompensa foram utilizados para ajustar os pesos da rede.

Após cada ação, a pontuação total acumulada, resultante da aplicação da função de recompensa, foi utilizada como sinal para atualizar os pesos sinápticos da rede neural de controle. Esse processo iterativo de coleta de dados, cálculo da recompensa, atualização dos pesos e iterações de treinamento permitiu que a rede neural de controle aprendesse a associar corretamente as ações tomadas com as recompensas e penalidades recebidas.

Ao longo do treinamento, a rede neural de controle refinou suas habilidades, aprendendo a gerar ações mais eficientes e adaptativas para manter o robô quadrúpede equilibrado. O processo de treinamento foi repetido várias vezes, permitindo que o robô aprimorasse seu desempenho e maximizasse a pontuação obtida por meio do sistema de recompensa.

É importante ressaltar que a definição da função de recompensa e a seleção adequada dos parâmetros associados foram realizadas com cuidado, a fim de obter resultados satisfatórios. Além disso, técnicas adicionais, como desconto temporal, foram consideradas para otimizar o treinamento do robô quadrúpede.

A fórmula para aplicar o desconto temporal e calcular a recompensa utilizada foi:

Onde:

V(t) é o valor total descontado no tempo t,

R(t+n) é a recompensa obtida k passos à frente no tempo,

γ é o fator de desconto (0 ≤ γ ≤ 1)

## É importante observar que a importância dos valores de recompensa diminui à medida que nos afastamos do tempo atual (t=0). Isso ocorre devido à aplicação do fator de desconto temporal, que atribui menos peso às recompensas futuras/passadas em relação às recompensas imediatas. Consequentemente, as recompensas distantes no tempo têm menos influência nas decisões atuais, levando a um maior foco nas ações e resultados imediatos, mas ainda levando em consideração a influência das tomadas de decisões que afetam o futuro.

A função de recompensa R(t) retorna um valor entre 0 e 1, representando o quão próximo o robô está do objetivo naquele instante de tempo. Ela é calculada como 1 menos o erro médio quadrático normalizado em uma escala de 0 a 1.

A fórmula utilizada para calcular a recompensa é:

Onde:

X\_t e Y\_t são os ângulos medidos/previstos pelo robô no instante de tempo t,

X\_objetivo e Y\_objetivo são os ângulos de referência desejados para o robô,

180^2 é o valor máximo possível para o erro quadrático, considerando que a diferença máxima entre os ângulos é de 180.

## 3.4.3.1 Algoritmo de treinamento

# Referências

Oliveira, Josenalde Barbosa. Controle Adaptativo Indireto por Modelo de Referência e Estrutura Variável. Fevereiro de 2003. Universidade Federal do Rio Grande do Norte. Disponível em https://repositorio.ufrn.br/bitstream/123456789/27034/1/josenaldeBO2003.pdf. Acesso em 25/06/2022.

Monhambe. Spider robot, quad robot, quadruped. Thingiverse. Fevereiro de 2018. Disponível em https://www.thingiverse.com/thing:2796820/files. Acesso em 24/06/2022.

Torres, Luiz. Controle adaptativo de sistemas dinâmicos: uma proposta para o relaxamento da condição de excitação persistente. UNIPÊ. Disponível em https://docs.google.com/document/d/1leEvG7BlZA\_AzOKKqlpbLkzYGu30YNW5/edit. Acesso em 23/06/2022.

Pan, Yunpeng; Wang, Jun.Model. Predictive Control of Unknown Nonlinear Dynamical Systems Based on Recurrent Neural Networks. IEEE Transactions on Industrial Electronics. IEEE, volume 59, 10.1109/TIE.2011.2169636, p. (3089 - 3101), Agosto, 2012.

Aurélio, Marco. ALGORITMOS GENÉTICOS: PRINCÍPIOS E APLICAÇÕES. ICA: Laboratório de Inteligência Computacional Aplicada. Disponível em http://www.inf.ufsc.br/~mauro.roisenberg/ine5377/Cursos-ICA/MQ-intro\_apost.pdf. Acesso em 23/06/2022.

SANTOS, Ana Paula; MILAGRES, Nádia; CAMPOS,Aline, MARGOTI,Luciana, AMARAL, Gleison; BARROSO, Marcos. **APLICAÇÃO DE REPRESENTAÇÕES EM BLOCOS INTERCONECTADOS EM IDENTIFICAÇÃO CAIXA-CINZA DE SISTEMAS DINÂMICOS NÃO LINEARES**. Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal de São João del-Rei. Disponível em https://www.researchgate.net/profile/Luciana-Margoti/publication/268438807\_APLICACAO\_DE\_REPRESENTACOES\_EM\_BLOCOS\_INTERCONECTADOS\_EM\_IDENTIFICACAO\_CAIXA-CINZA\_DE\_SISTEMAS\_DINAMICOS\_NAO\_LINEARES/links/5656f42b08ae4988a7b51542/APLICACAO-DE-REPRESENTACOES-EM-BLOCOS-INTERCONECTADOS-EM-IDENTIFICACAO-CAIXA-CINZA-DE-SISTEMAS-DINAMICOS-NAO-LINEARES.pdf. Acesso em 23/06/2022.