ANDRÉ DE SOUZA REIS
FERNANDO KAMNITZER BRACCO

PLATAFORMA DIDÁTICA DE MÉTODOS DE CONTROLE COM O PÊNDULO INVERTIDO ROTACIONAL

São Paulo
2021

ANDRÉ DE SOUZA REIS
FERNANDO KAMNITZER BRACCO

PLATAFORMA DIDÁTICA DE MÉTODOS DE CONTROLE COM O PÊNDULO INVERTIDO ROTACIONAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduação em engenharia mecatrônica.

Área de Concentração:

Área de concentração

Orientadora:

Prof. Dra. Larissa Driemeier

Co-orientador:

Prof. Dr. Rafael Traldi Moura

São Paulo
2021

AGRADECIMENTOS

Gostariamos de agradecer à Prof. Dra. Larissa Driemeier e ao Prof. Dr. Rafael Traldi Moura pela sólida orientação para a realização deste trabalho. Também somos gratos aos familiares, amigos e professores que tiveram parte fundamental na realização deste projeto através de críticas e sugestões.

"Inteligência é a capacidade de se adaptar à mudança."

-- Stephen Hawking

RESUMO

O pêndulo invertido, um dos problemas mais difundidos na engenharia, é frequentemente selecionado para atuar como plataforma didática de ensino na área de controle, principalmente em função de seu caráter não linear e de equilíbrio instável. Além disso, a construção de um protótipo não demanda custos elevados e nem requer alta complexidade. Nesse contexto, o projeto consiste em desenvolver uma plataforma didática de testes a fim de explorar técnicas de aprendizagem por reforço e de optimização para problemas de controle não lineares de equilíbrio instável, e compará-las com técnicas clássicas. Para isso, foi modelado e construído um pêndulo invertido rotacional passível de ser reproduzido em laboratório.

Palavras-Chave – Pêndulo invertido rotacional, controle clássico, aprendizagem por reforço, *Q-Learning*, *swing up*.

ABSTRACT

The inverted pendulum, one of the most widespread problems in engineering, is often selected to serve as a didactic platform in the control field, mainly due to its non-linear character and unstable balance. Furthermore, the construction of a prototype does not demand high costs nor require high complexity. In this context, the project consists of developing a didactic platform aimed at exploring reinforcement learning and optimization for non-linear control problems of unstable equilibrium, and comparing them with classical techniques. For this purpose, it was built a rotary inverted pendulum structure capable of being reproduced in a laboratory.

Keywords – Rotary inverted pendulum, classical control, reinforcement learning, *Q-Learning*, *swing up*.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

O pêndulo invertido consiste em um dos problemas fundamentais da engenharia [8]. Amplamente estudado e utilizado como exemplo no contexto didático, sua importância advém primordialmente da variedade de aplicações teóricas e práticas nas quais o modelo pode ser introduzido. Enquanto seu valor teórico surge do caráter não linear e instável do sistema, sua relevância prática deriva de uma série de aplicações reais. Os autores ainda ressaltam a vantagem do custo reduzido e simplicidade de construção do protótipo em um laboratório.

Além de traduzir de forma simplificada uma dinâmica complexa, o processo de modelagem do pêndulo invertido abrange uma ampla gama de tópicos abordados na teoria de controle clássico [9], auxiliando de maneira eficiente na elaboração de uma plataforma didática.

Um exemplo de aplicação do pêndulo invertido no contexto didático pode observado na Figura 1, em que é utilizada a plataforma FRDM-K64F da NXP para implementação de um controle digital LQR em um laboratório de controle aplicado [1]. Além disso, outra abordagem interessante consiste no controle por meio de inteligência artificial, devido à capacidade do protótipo de proporcionar e replicar testes de maneira automática, requisito essencial na aprendizagem por reforço.

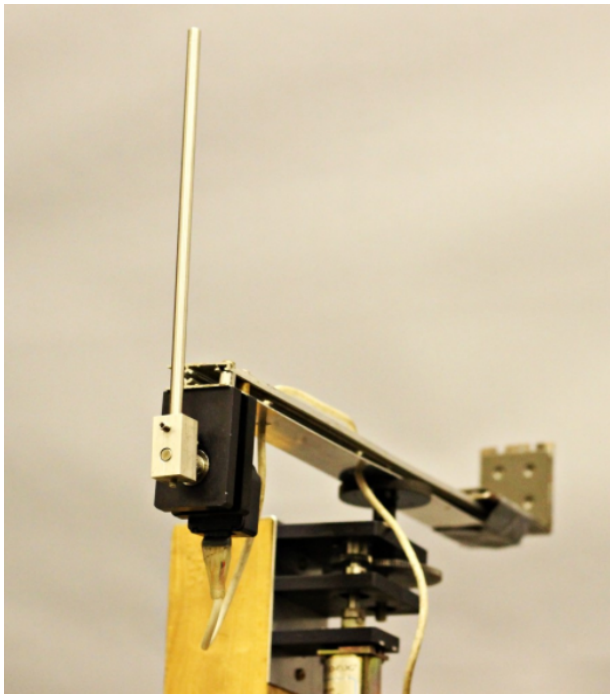
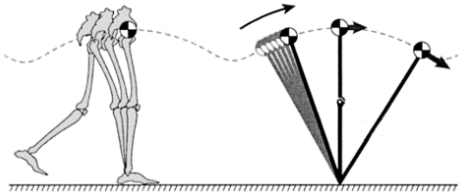


Figura 1: Pêndulo invertido rotacional ou Pêndulo de Furuta, extraído de [1].

Nesse contexto, torna-se possível realizar análises e comparações tanto da otimização da estrutura mecânica quanto do desempenho do algoritmo de controle. Em função disso, o modelo é frequentemente selecionado para testar novos métodos de controle, tais como controle ótimo, controle *fuzzy*, controle através de redes neurais, controle preditivo e métodos híbridos como combinações dos anteriores [8].

Quanto às diversas aplicações do pêndulo invertido no mundo real, nota-se exemplos nos âmbitos econômico e social, nos quais os mais recorrentes se apresentam em áreas como transporte e mobilidade, envolvendo dispositivos como o *Segway* [8–10] e foguetes (lançamento e pouso) [11], sistemas de auxílio à locomoção humana (suporte para andar) [8, 10], transporte de objetos através de *drones* [4] e até construções de larga escala [12]. As Figuras 2a, 2b e 3 ilustram exemplos de aplicações.



(a) Modelo de pêndulo invertido para locomoção humana, extraído de [13].

(b) *Inverted Pendulum Moving Robot*, extraído de [14].

Figura 2: Exemplos de aplicações com o estudo da dinâmica do pêndulo invertido

Figura 3: Lançamento de foguete, extraído de [2].

Por fim, evidenciada a relevância do modelo no contexto da engenharia [8], o pêndulo invertido rotacional foi selecionado como objeto de estudo deste projeto, proporcionando um ambiente simultaneamente simples e completo para se realizar as análises descritas a seguir.

1.2 Estado da Arte

1.2.1 Construção

Conforme enfatizado, o pêndulo invertido e o pêndulo invertido rotacional são problemas clássicos de controle e largamente estudados. Por conta disso, existem no mercado plataformas de estudo que incluem a mecânica, o hardware e o software necessários para estudar a dinâmica e o controle de tais problemas. Para focar os esforços no estudo especificamente do controle da estrutura e não na sua prototipagem e fabricação é comum encontrar publicações que utilizam tais plataformas.

Os trabalhos [15, 16] utilizam a plataforma ilustrada na Figura 4a, que possui como estrutura mecânica basicamente dois perfis de metal. A atuação do pêndulo é feita por meio de um motor DC escovado e o sensoriamento do ângulo do pêndulo, e do eixo do motor são feitos através de encoders.

Já a QUBE-Servo RIP platform é uma plataforma comercial mais recente e utilizada em [17,18]. O sistema possui características similares à citada anteriormente, como utilizar um motor DC escovado como atuador, e encoders para obter as posições do pêndulo e eixo do motor.

(a) Plataforma utilizada por [15, 16], extraído de [19].

(b) Plataforma utilizada por [17, 18], extraído de [17].

Figura 4: Exemplos de estruturas comerciais para o estudo da dinâmica do pêndulo invertido rotacional

1.2.2 Modelagem

A dinâmica do pêndulo invertido rotacional é altamente não-linear, porém, uma vez que o objetivo é estabilizar o pêndulo na posição de equilíbrio instável, é comum utilizar

um modelo linearizado. Em contra partida, quando o objetivo é levar o pêndulo da posição de equilíbrio estável para a de equilíbrio instável, já não é possível trabalhar com um modelo linearizado.

Os trabalhos [3,15–17] tem como objetivo controlar o pêndulo na posição de equilíbrio instável, portanto, utilizam a estratégia de linearização e trabalham com o espaço de estados. Para encontrar as equações de movimento linearizadas, os trabalhos [3, 15, 17] utilizam as Equações de Euler-Lagrange. Já em [16] os autores optam por encontrar as equações não lineares que descrevem o movimento da estrutura e em seguida linearizá-las.

Os modelos dos trabalhos citados anteriormente incluem a dinâmica da estrutura do pêndulo e do motor DC, possuindo como variável de controle o valor de tensão a qual o motor DC é submetido, e como estados, os valores de posição e velocidade angular medidos pelos encoders.

1.2.3 Técnicas clássicas de controle

Como citado anteriormente, o controle do pêndulo invertido pode ter como objetivo manter o pêndulo próximo da posição de equilíbrio instável, ou levá-lo da posição de equilíbrio estável para a de equilíbrio instável. Para cada situação existe um conjunto de técnicas que podem ser utilizadas.

Para manter o pêndulo na posição de equilíbrio instável é comum utilizar algoritmos como: controle Proporcional-Integral-Derivativo (PID), Regulador Quadrático Linear (RQL) e Alocação de Polos (AP).

Uma das técnicas utilizadas em [16] é o controle PID, com uma configuração de dois controladores em série. Esta configuração é necessária pois deseja-se controlar não apenas a posição do pêndulo, mas também a posição ou a velocidade do braço. Em [16], o segundo controlador foi utilizado para a posição do braço.

Já o RQL, utilizado em [3,15–17], se caracteriza por encontrar uma regra de controle ótima para transferir o sistema de um estado inicial até um estado final minimizando determinados parâmetros. Os parâmetros geralmente utilizados são: o erro entre estado atual e estado desejado; e a energia gasta no processo.

Uma vantagem do RQL é o fato de se poder controlar mais de um estado ao mesmo tempo, que como já mencionado, é necessário para o controle do pêndulo invertido rotacional. Porém, uma desvantagem deste método é a necessidade de se encontrar matrizes que ponderam a prioridade de controle entre estados, e a forma mais comum de encontrar esta matriz é através de tentativa e erro, como é feito em [3, 15–17].

Na Figura 5 pode-se observar a resposta obtida pelo pêndulo invertido rotacional de [3] utilizando o controlador RQL desenvolvido, no qual houve uma perturbação em formato de degrau entre os instantes 1 e 3 segundos.

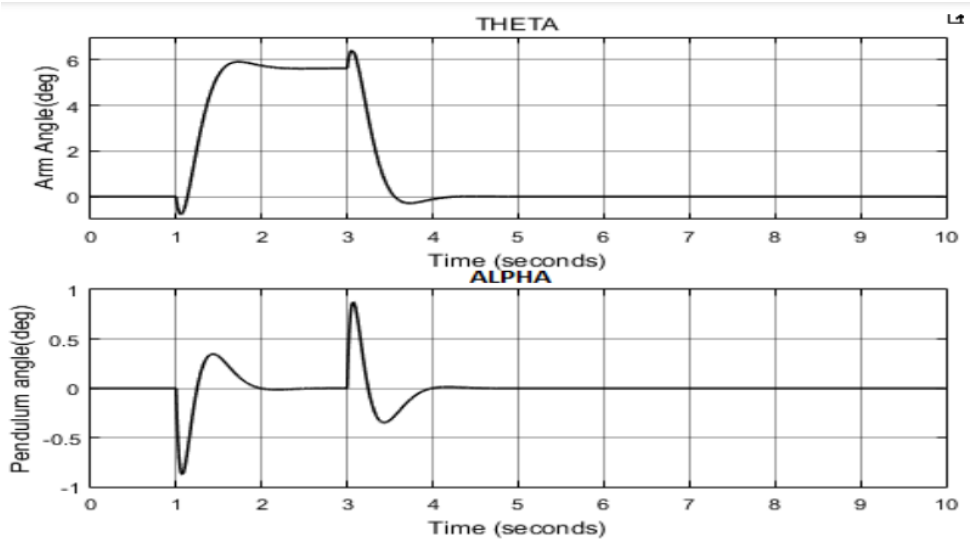


Figura 5: Resposta da planta desenvolvida por [3] utilizando um controlador RQL, extraído de [4].

Para realizar o controle com o objetivo de levar o pêndulo da posição de equilíbrio instável para a de equilíbrio estável as técnicas citadas anteriormente não são eficazes. Uma abordagem comum é basear o controle na energia total do pêndulo, ou seja, na soma das energias cinética e potencial. Toma-se como referência a energia total do pêndulo na posição vertical superior e o controle é feito com base na energia total instantânea do sistema em relação à referência, tentando sempre diminuir tal diferença.

[5] desenvolve um modelo de pêndulo invertido a fim de testar uma derivação da abordagem de controle citada anteriormente. Conforme os resultados apresentados, reduz-se o tempo necessário para o pêndulo ir da posição vertical inferior para a superior ao considerar possíveis atritos e resistências do sistema. Dessa forma, é introduzido um ganho no esforço de controle quando o pêndulo se encontra distante da posição de equilíbrio superior.

Observa-se na Figura 6 os resultados obtidos da simulação realizada por [5], cuja posição inicial do pêndulo foi a vertical inferior e a posição final, a vertical superior. LP é a curva que representa a utilização do algoritmo mais comum, e E, a utilização do algoritmo proposto.

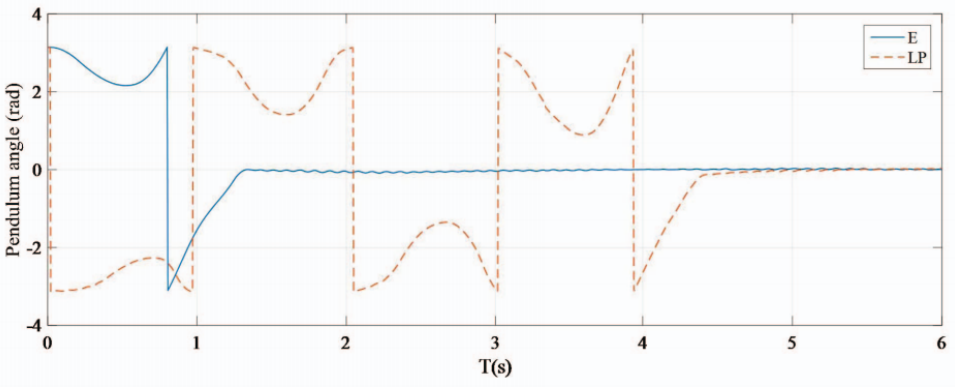


Figura 6: Simulação feita por [5] comparando dois algoritmos de controle, extraída de [5].

1.2.4 Técnicas de aprendizado por reforço

Apesar de não constituir uma tecnologia recente, a inteligência artificial tem ganhado mais relevância e ambiente para aplicação com o aumento da capacidade de processamento computacional, requisito importante no processo de aprendizagem de máquinas. Assim, tem apresentado uma crescente influência em manufatura e processos industriais, sendo uma de suas aplicações o projeto de controladores através de aprendizagem por reforço [20].

Conforme ressaltado anteriormente, uma das características que amplia a complexidade no desenvolvimento e implementação dos controladores para o pêndulo invertido rotacional consiste na determinação dos parâmetros de controle a serem utilizados, principalmente no RQL, no qual as possibilidades de combinações são elevadas.

Por meio da aprendizagem por reforço, tal processo de determinação dos parâmetros é eliminado, o que implica redução do esforço humano comumente envolvido no projeto [20], principalmente ao considerar um sistema de natureza instável e não linear. Dessa maneira, em [20] é desenvolvido um controlador para um pêndulo invertido rotacional através da combinação de uma rede neural artificial com aprendizagem por reforço, além de estabelecer uma comparação com um controlador convencional quanto ao algoritmo de *swing-up* e de estabilização.

A aprendizagem profunda, por sua vez, atualmente permite escalar a aprendizagem por reforço para problemas previamente intratáveis [21]. Em [6] se propõe a combinação de ambas com o objetivo de melhorar o desempenho no controle, ou seja, utiliza o pêndulo invertido rotacional como um ambiente para implementar o processo de aprendizagem por reforço profunda. O algoritmo de controle desenvolvido na aplicação é o *Deep Q-Network*, que utiliza uma rede neural convolucional para inferir as ações do agente da aprendizagem e implementa o processo de aprendizagem por reforço do controle do pêndulo.

Após a execução do processo de aprendizagem, em [6] se propõe a utilização de um controlador PID a fim de implementar aprendizagem por imitação e assim agilizar o procedimento, comparando-se com o algoritmo sem imitação. Ambos os procedimentos obtiveram sucesso em controlar o pêndulo na posição instável, sendo que os resultados obtidos em ambas as tarefas podem ser observados nas Figuras 7 e 8, sendo que a segunda ilustra o comportamento do sistema controlado pelo algoritmo com imitação. A partir das figuras, se observa que o procedimento de aprendizagem sem imitação é finalizado com 599 episódios, enquanto o procedimento com imitação termina após 340 episódios. Assim, o erro é reduzido mais rapidamente e como conclusão se infere que a estratégia da aprendizagem por imitação reduz o tempo de aprendizagem de forma eficiente [6].

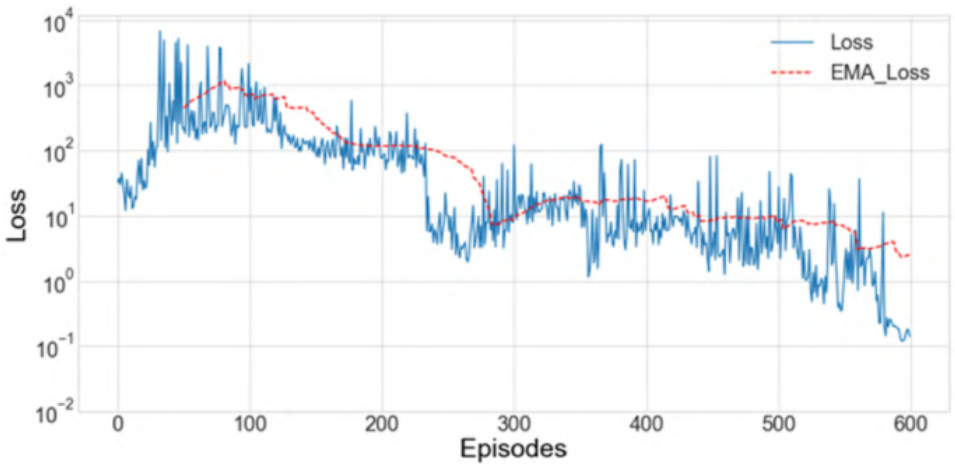


Figura 7: Aprendizagem da *Deep Q-Network* sem imitação, extraído de [6].

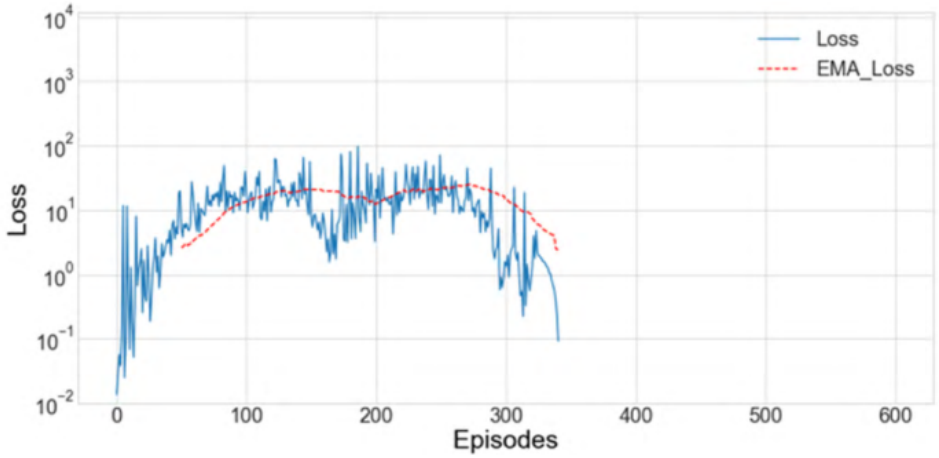


Figura 8: Aprendizagem da *Deep Q-Network* com imitação, extraído de [6].

2 OBJETIVOS

O objetivo do projeto é construir uma plataforma didática de testes que possua a estrutura de pêndulo invertido rotacional e utilizá-la para proporcionar e consolidar o ensino de técnicas de controle.

Inicialmente, com o objetivo de proporcionar o aprendizado em controle clássico, busca-se elaborar um algoritmo para manter o pêndulo na posição de equilíbrio instável. Pretendendo-se implementar inicialmente o controle proporcional integrativo derivativo (PID).

Na sequência, o projeto também deve abranger o estudo da dinâmica de levar o pêndulo da posição de equilíbrio estável para a de equilíbrio instável. Para isso pretende-se introduzir algoritmos comumente empregados no contexto, que são baseados na avaliação da energia do sistema. São conhecidos como algoritmos de 'Swing-up'.

Posteriormente, pretende-se propor o mesmo controle através da abordagem da inteligência artificial, com o desenvolvimento de algoritmos de aprendizagem por reforço. Neste caso, os algoritmos podem ser aplicados tanto para manter o pêndulo equilibrado na posição instável, quanto para levá-lo até esta posição.

Por fim, pretende-se analisar e comparar os resultados de todas as abordagens através de aspectos como: energia total gasta e robustez. Assim, o projeto deve elucidar considerações quanto às vantagens e desvantagens do uso de técnicas de aprendizado por reforço para este problema em relação às técnicas de controle clássico.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Modelo do pêndulo invertido

As equações (3.1) e (3.2) que descrevem o comportamento não linear de todo o sistema, foram obtidas em [8] utilizando equações cinemáticas, dinâmicas, o motor DC modelado na equação (3.3) e desprezando as forças de atrito. A nomenclatura utilizada está ilustrada na Figura 9 e descrita na Tabela 1.

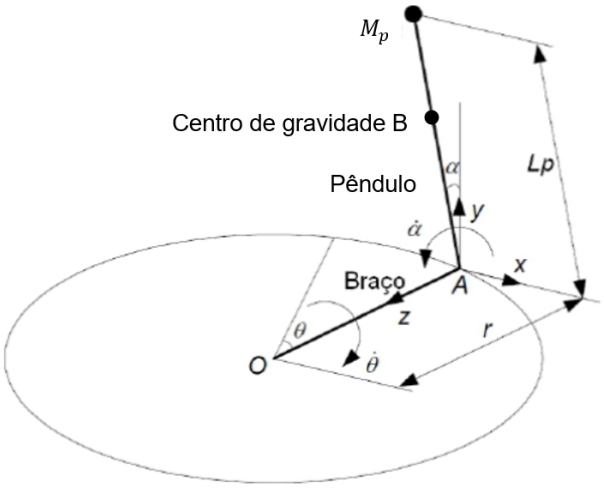


Figura 9: Parâmetros utilizados no modelo da estrutura.

$$\ddot{\alpha} = \frac{1}{ac - b^2 \cos^2 \alpha}(ad \sin \alpha - b^2 \sin \alpha \cos \alpha \dot{\alpha}^2 - be \cos \alpha \dot{\theta} + bf \cos \alpha V_m) \tag{3.1}$$

$$\ddot{\theta} = \frac{1}{ac - b^2 \cos^2 \alpha}(bc \sin \alpha \dot{\alpha} + bd \sin \alpha \cos \alpha - ce\dot{\theta} + cfV_m) \tag{3.2}$$

Em que: $a = J_{eq} + M_p r^2$, $b = M_p L_p r$, $c = J_p + M_p L_p^2$, $d = M_p g L_p$, $e = \frac{K_t K_m}{R_m}$, $f = \frac{K_t}{R_m}$.

$$\tau_{output} = \frac{K_t(V_m - K_m \dot{\theta})}{R_m} \tag{3.3}$$

Variável	Descrição
g	Aceleração da gravidade
J_{eq}	Momento de inércia da montagem braço/pêndulo em relação ao eixo do motor
J_p	Momento de inércia do pêndulo em relação ao seu eixo de rotação
K_t	Constante corrente-torque do motor
K_m	Constante de força contra eletromotriz do motor
L_p	Distância entre centro de massa do pêndulo e seu eixo de rotação
M_p	Massa do pêndulo
r	Comprimento do braço da estrutura
R_m	Resistência da armadura do motor
V_m	Tensão a qual o motor é submetido
α	Ângulo que varia em torno do eixo z, definido entre o pêndulo e a posição de equilíbrio instável
θ	Ângulo do braço da estrutura que varia em torno do eixo y
τ_{output}	Torque fornecido pelo motor

Tabela 1: Variáveis utilizadas no modelo da estrutura.

3.2 Método de controle: LQR

Dado o sistema dinâmico linear representado por $\dot{x} = Ax + Bu$, em que x é o vetor de estados, u a entrada, e as matrizes A e B representam o modelo dinâmico em particular, o Regulador Quadrático Linear, do inglês *Linear Quadratic Regulator* tem como objetivo encontrar o vetor u que minimize a função de custo quadrático dado pela expressão (3.4).

$$J_{QRL} = \int_0^\infty (x^T Q x + u^T R u) dt \tag{3.4}$$

Portanto, através da matriz Q, se impõe quais estados terão prioridade em serem levados para a origem, ou anulados, de forma que quanto maior o valor do elemento da diagonal de Q, maior será a prioridade para o estado correspondente. De forma análoga, quanto maior o valor do elemento da diagonal de R, maior será a minimização do esforço de controle correspondente.

Considerando a formulação de Euler Lagrange para o seguinte problema de minimização:

$$\min_u \int_0^T L(x,u)dt + \Phi(x(T)), \text{ sujeito a } \dot{x} = f(x,u) \tag{3.5}$$

A solução u deve satisfazer as seguintes restrições:

$$\begin{cases} H_u = 0 \\ \dot{\lambda}^T = -H_x \\ \lambda(T) = \phi_x(x(T)) \end{cases} \tag{3.6}$$

em que $H = L + \lambda^T f$.

Para utilizar tais fórmulas na minimização de (3.4), define-se $H = \frac{1}{2}\left(x^T Q x + u^T R u\right) + \lambda^T(Ax + Bu)$. Assim, a primeira restrição indicará que:

$$u = -R^{-1}B^T\lambda \tag{3.7}$$

Já a segunda restrição e a definição de $\lambda(t)$ da seguinte forma: $\lambda(t) = P(t)x(t)$, indicará:

$$\dot{P} + PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{3.8}$$

Considerando $T \to \infty$ e $\dot{P} = 0$, obtém-se a Equação de Riccati, uma equação não linear de primeira ordem:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{3.9}$$

Assim, encontrando a solução de (3.9) de forma regressiva no tempo, é possível determinar a lei de controle, que será dada por:

$$u = -R^{-1}B^T P x = K_{LQR}\, x \tag{3.10}$$

Nota-se que apesar de ser necessário utilizar as matrizes A e B para encontrar a lei de controle, é possível utilizar o Regulador Quadrático Linear em sistemas não linearizados, uma vez que para encontrar u, basta conhecer x.

3.3 Método de controle: Swing-up

O problema do pêndulo invertido pode ser subdividido em dois: controlá-lo na posição de equilíbrio instável, e levá-lo da posição vertical inferior para a posição vertical superior,

movimento chamado de *swing up*. Para executar a segunda tarefa, diversas soluções já foram propostas, porém uma das mais utilizadas é o controle não linear baseado em energia.

Tomando como base os parâmetros mostrados na Figura 9 e na Tabela 1, a equação de movimento do pêndulo pode ser descrita da seguinte forma:

$$J_p\ddot{\alpha} - M_p\ g\ L_p \sin(\theta) + M_p\tau_{output}\ L_p\ r\cos(\alpha) = 0 \tag{3.11}$$

Já a equação de Energia (cinética e potencial gravitacional), pode ser escrita da seguinte forma:

$$E = \frac{1}{2}J_p\ \dot{\alpha}^2 + M_p\ g\ L_p(\cos(\alpha) - 1) \tag{3.12}$$

Derivando a equação (3.12) e substituindo $\ddot{\alpha}$ da equação (3.11), obtém-se:

$$\frac{dE}{dt} = J_p\ \dot{\alpha}\ \ddot{\alpha} - M_p\ g\ L_p\ \dot{\alpha}\sin(\alpha) = -M_p\ \tau_{output}\ L_p\ r\ \dot{\alpha}\cos(\alpha) \tag{3.13}$$

Como citado em [5], existem formas de se encontrar leis de controle baseadas na função de Lyapunov, porém pode-se apenas garantir que $\frac{dE}{dt}$ seja positivo. Ou seja:

$$u = -ksign(\dot{\alpha}\cos\alpha) \tag{3.14}$$

Onde k é um parâmetro ajustável e:

$$sign(x) = \begin{cases} 1, \text{se } x >= 0 \\ -1, \text{se } x < 0 \end{cases} \tag{3.15}$$

Como também citado por [5], por conta de possíveis resistências e atritos do sistema, pode ser que mais energia deva ser transferida ao pêndulo para se chegar na posição de equilíbrio instável. Para isso, a seguinte otimização é sugerida:

$$u = -k\left[1 + \left(\frac{\alpha}{\pi}\right)^2\right] sign(\dot{\alpha}\cos(\alpha)) \tag{3.16}$$

3.4 Aprendizado por reforço

3.4.1 Introdução

Aprendizado por reforço pode ser interpretado como um treinamento de modelos de Aprendizado de Máquina para a execução de uma determinada tarefa. A principal característica de tal treinamento é que não é necessário classificar a ação tomada pelo modelo, ou agente, como correta ou incorreta, mas apenas retornar ao modelo uma recompensa positiva ou negativa, de acordo com o objetivo final da tarefa. Para isso, estrutura-se o problema como a seguir.

O agente observa os estados S_t e a recompensa recebida R_t de um determinado ambiente e através da sua política, baseada em um grupo de parâmetros θ, executa uma ação A_t. Tal ação alterará os estados do ambiente para S_{t+1}, que de acordo com o objetivo do agente gerará a recompensa R_{t+1}. Tal estrutura e parâmetros podem ser observadas na Figura 10.

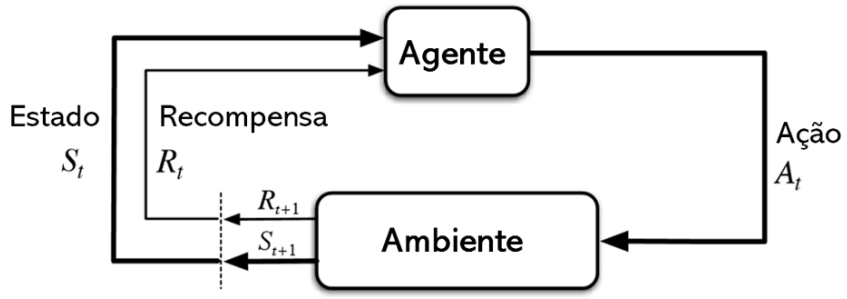


Figura 10: Modelo de aprendizado de máquina e respectivos parâmetros. Adaptado de [7]

Esta sequência de acontecimentos é chamada de "episódio" e após um determinado número de episódios, as recompensas são acumuladas e a política de ações do agente é avaliada. Caso o objetivo do agente não seja atingido, os parâmetros θ da política de ações são alterados e todo o processo se repete de forma iterativa.

3.4.2 Algoritmo: *Q-Learning*

O *Q-Learning* constitui um dos algoritmos mais comuns para a implementação de aprendizado por reforço, cujo objetivo se baseia em determinar qual a melhor ação a ser tomada, dado um estado específico. Dessa forma, ao interagir com o ambiente, o agente busca aprender uma política que maximize a recompensa total esperada.

Inicialmente, uma vez definidos o ambiente, o agente, as possíveis ações a serem tomadas e a função de recompensa, é necessário construir uma tabela de valores, conhecida como *Q-Table*. Tal tabela relaciona os estados com as ações, atribuindo um valor referente às recompensas adquiridas para as ações em determinado estado. Como inicialmente o ambiente é desconhecido, pode-se inicializar a tabela com valores arbitrários.

Na sequência, o algoritmo pressupõe que o agente tome uma ação, que inicialmente é aleatória, dado o desconhecimento. Tal ação implicará uma recompensa conforme definida no ambiente, que permitirá a consequente atualização da tabela, caracterizando a relação da ação com aquele estado. A atualização, por sua vez, é implementada com base na equação de *Bellman*, que relaciona o valor de um estado com a máxima recompensa esperada e o valor do estado anterior, aliado a um fator de desconto, que objetiva reduzir seu valor conforme o número de ações tomadas até o instante atual [22].

A equação de *Bellman* na forma determinística está apresentada na equação 3.17, em que λ constitui o fator de desconto e s' o próximo estado.

$$V(s) = max_a(R(s,a) + \lambda V(s')) \tag{3.17}$$

em que $R(S,a)$ é a recompensa obtida por estar no estado s e tomar a ação a.

O valor de um estado consiste no maior entre todos os valores Q possíveis obtidos na tabela. Assim, torna-se possível reescrever a equação analisando o valor Q de uma posição na tabela em função dos mesmos parâmetros, mas considerando o par ação-estado, conforme a equação 3.18.

$$Q(s,a) = R(s,a) + \lambda max_{a'} Q(s',a') \tag{3.18}$$

Por fim, considerando a diferença temporal entre o novo valor obtido e o referente ao instante anterior, inclui-se uma taxa de aprendizado α que multiplica a diferença e soma ao valor prévio, obtendo-se a equação 3.19, que expõe a atualização de fato implementada na tabela.

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) + \lambda max_{a'} Q(s',a') - Q_{t-1}(s,a)) \tag{3.19}$$

Uma vez atualizado o valor, o processo se repete de forma iterativa ao longo do processo de aprendizado, atualizando os valores da tabela até que determinado objetivo seja

atingido ou a aprendizagem seja interrompida. Conforme descrito, inicialmente as ações realizadas são aleatórias, entretanto, conforme se avança no processo de aprendizagem, as ações passam a ser tomadas considerando os valores já conhecidos da tabela.

Ao fim do processo, a partir da *Q-Table* obtida com as atualizações, é possível selecionar a melhor ação a ser tomada pelo agente em determinado estado no ambiente.

3.4.3 Aprimoramentos do modelo

3.4.3.1 *Deep Reinforcement Learning (Deep Q-Learning)*

O algoritmo *Q-Learning* atua de forma satisfatória em contextos que abordam ambientes simples, com um número reduzido de estados e ações possíveis. Uma vez que o ambiente se torna mais complexo (com o crescimento da quantidade de ações e estados), o número de combinações se eleva significativamente, tornando inviável a aplicação do algoritmo. Nesse contexto, se introduz o *Deep Q-Learning*, um algoritmo desenvolvido a fim de fornecer um aproximador para os valores Q, por meio da implementação de redes neurais [22].

Dessa forma, o *Deep Q-Learning* pressupõe a substituição da *Q-Table* por uma rede neural, cujas entradas são os estados e como saída obtém-se os valores Q (referentes à tabela) relacionados às ações. A partir dos valores extraídos como saídas, é possível realizar uma comparação com os valores previamente estimados e atualizar os pesos ao longo das camadas da rede por meio de um algoritmo como retropropagação ou gradiente descendente.

Com o objetivo de construir uma rede neural, torna-se necessário o estabelecimento de uma função de custo, novamente baseada na equação de Bellman, mas neste caso buscando minimizar o quadrado da diferença entre os lados da equação 3.18. A função de custo resultante é representada pela equação 3.20, em que a aproximação de Q é indicada como $Q(s, a; \theta)$, no qual θ representa os pesos treináveis da rede.

$$custo = [Q(s, a; \theta) - (R(s, a) + \lambda max_{a'} Q(s', a'; \theta))]^2 \tag{3.20}$$

Uma vez estabelecida a função de custo, a fim de realizar o treinamento é necessário considerar ainda o conceito de *experience replay*, que consiste na compreensão equivocada do ambiente pelo agente em função de uma possível sequência de estados interdependentes

e similares. A solução envolve uma não atualização imediata dos pesos da rede a cada episódio. Ao longo do treinamento os episódios são salvos e após determinado limite, é efetuada uma amostragem aleatória pelo agente, que seleciona apenas esses episódios para aprendizagem. Tal procedimento evita o possível enviesamento proveniente da sequência de estados, permitindo um treinamento eficiente da rede.

Por fim, enquanto o processo de seleção da ação a ser tomada no *Q-Learning* envolve apenas a escolha do maior valor Q, no *Deep Q-Learning* a seleção requer um tratamento dos valores Q por uma função, como a *softmax* ou a ϵ-*greedy* [22]. Tal tratamento objetiva alterar a forma de seleção das ações ao curso do treinamento, passando de ações aleatórias (enquanto o conhecimento do ambiente é limitado) para ações considerando o conhecimento já adquirido sobre o ambiente e os pesos atualizados da rede neural.

3.4.3.2 *Imitation Learning*

Os algoritmos de aprendizagem por reforço previamente descritos possuem bom desempenho em situações que envolvem uma função de recompensa bem definida com elevada frequência de ocorrência, através da qual é possível se aproximar à melhor política. Em alguns casos, no entanto, as recompensas podem ser bem esparsas, o que dificulta o processo de aprendizagem. Nesses contextos é possível desenvolver manualmente novas funções de recompensas cujas frequências de ocorrência são superiores.

O desenvolvimento manual das funções também é relevante em cenários nos quais não há uma função direta de recompensa, tais como aprendizagem de veículos autônomos [23]. Todavia, o processo de determinação da nova função de forma a satisfazer o comportamento desejado pode ser complexo. Nesse contexto, pode-se introduzir como solução um algoritmo de *Imitation Learning*. Ao invés de promover o aprendizado por meio de recompensas esparsas ou especificando manualmente uma função de recompensa, o algoritmo propõe o aprendizado com base em um conjunto de demonstrações fornecidas por um *expert* (normalmente humano), buscando aprender a melhor política seguindo suas ações. Nesse sentido, a aprendizagem pode ocorrer de forma mais eficiente.

Um algoritmo de *Imitation Learning* pode ser implementado através de diferentes métodos, tais como *Behavioural Cloning*, *Direct Policy Learning* e *Inverse Reinforcement Learning*, entre outros.

A primeira abordagem constitui a forma mais simples de *Imitation Learning*, além de

ser eficiente. O processo de aprendizagem da política do *expert* pelo algoritmo *Behavioural Cloning* se baseia em aprendizagem supervisionada, com a divisão das demonstrações em pares estado-ação e a consequente definição de uma função de custo, que se objetiva minimizar. No entanto, uma de suas suposições principais consiste na distribuição idêntica e independente dos pares estado-ação, o que pode não se aplicar em diversos modelos. Além disso, a suposição implica que erros em diferentes estados se somam, possivelmente levando o agente a estados nos quais o *expert* nunca esteve, induzindo um comportamento indefinido.

Já o algoritmo *Direct Policy Learning* não apresenta a mesma limitação, ao pressupor um acesso em tempo real ao *expert*, de forma que avalie as ações tomadas pelo agente durante a aprendizagem. Dessa forma torna-se possível corrigir os erros antes que estados desconhecidos sejam atingidos. Todavia, tal procedimento pode não ser plausível, uma vez que a interatividade com o *expert* em diversas situações não é disponível.

O algoritmo *Inverse Reinforcement Learning*, por sua vez, propõe uma aprendizagem da função de recompensa do ambiente com base nas demonstrações e com isso determinar a política ótima. Dessa forma, ao longo do processo de aprendizado busca-se estimar os parâmetros referentes à função que causariam o comportamento do agente. Uma vez estimados, a partir da função torna-se possível estabelecer uma política de forma a maximizá-la. Na sequência, deve-se compará-la com a política do *expert* e atualizar a função de recompensa. Este processo pode ser repetido de forma iterativa até que a política seja satisfatória, com ações do agente induzindo respostas semelhantes as do *expert*.

4 METODOLOGIA

O procedimento a ser adotado no desenvolvimento do projeto se baseia na ordem de requisitos definidos na seção 2. A construção do pêndulo invertido rotacional demanda inicialmente o projeto simultâneo da mecânica e da eletrônica inseridas, exigindo a definição e caracterização dos componentes envolvidos. Tal determinação, por sua vez, exige estabelecimento dos requisitos a serem satisfeitos, o que implica a estimação dos esforços no protótipo. Dessa forma, com o intuito de promover a estimação, a etapa inicial consiste na modelagem do sistema e definição de seus requisitos e parâmetros relevantes.

Uma vez bem delimitados os requisitos, é possível realizar simulações da dinâmica do sistema com inserção dos esforços de controle, elucidando as especificações necessárias aos componentes mecânicos e eletrônicos para que satisfaçam os requisitos. Ao fim dessa etapa deve ser plausível a caracterização da lista de componentes assim como o desenvolvimento e as implementações dos projetos mecânico e eletrônico.

Na sequência, com o protótipo físico já finalizado, torna-se viável a execução dos projetos computacional e de controle, envolvendo a definição do microcontrolador a ser utilizado bem como o desenvolvimento dos algoritmos de controle a serem implementados. Tal processo permite enfim a realização dos testes, além das análises de resultados obtidos com suas respectivas comparações propostas, como tempo de estabilização, energia total gasta e robustez. Com isso obtém-se embasamento para as especificações da plataforma didática como objetivo do projeto.

4.1 Parâmetros e requisitos

Conforme ressaltado anteriormente, a definição dos parâmetros e requisitos envolvidos consiste um uma etapa inicial e essencial ao projeto. Suas especificações podem ser estabelecidas com base em projetos prévios que envolvem a construção de pêndulos

Parâmetro	Valor	Justificativa
Tempo de execução (*Swing Up*)	< 10 [s]	Tempo suficiente para execução do algoritmo conforme simulações e projetos prévios [24].
Esforço de controle	< 1 [Nm]	Reduz a necessidade de motor com custo elevado.
Preço	< 1000 [R$]	Orçamento suficiente para a construção de diversos protótipos, tornando plausível a replicação em laboratório.
Dimensões	<400x200x200 [mm]	Deve ser adequada para manipulação em uma bancada de laboratório.

Tabela 2: Tabela de requisitos do projeto.

invertidos rotacionais [24] e conforme requisitos essenciais no controle do pêndulo, como em sua capacidade de estabilização vertical.

Dessa forma, os parâmetros estudados e requisitos puderam ser classificados conforme a Tabela 2.

Já a validação dos valores das três linhas iniciais apresentadas na Tabela 2 podem ser verificadas por meio das simulações implementadas e detalhadas na seção 7.1.

5 PROJETO MECÂNICO

Uma vez encontradas as equações dinâmicas do sistema, torna-se necessário obter os parâmetros M_p, J_p, L_p, r e J_{eq}. Para isso, uma primeira proposta da estrutura do sistema foi modelada no software de CAD 3D SOLIDWORKS, no qual os materiais, dimensões e posicionamento das peças da estrutura foram especificados, sendo assim possível obter os parâmetros apresentados na Tabela 3.

Variável	Valor	Descrição
r	0.16 $[m]$	Comprimento do braço da estrutura
L_p	0.075 $[m]$	Distância entre centro de massa do pêndulo e seu eixo de rotação
J_{eq}	0.00012839 $[kg.m^2]$	Momento de inércia da montagem braço/pêndulo em relação ao eixo do motor
J_p	0.00005027 $[kg.m^2]$	Momento de inércia do pêndulo em relação ao seu eixo de rotação
M_p	0.022 $[kg]$	Massa do pêndulo

Tabela 3: Parâmetros extraídos do modelo da estrutura.

Na Figura 11 pode-se observar a proposta desenvolvida da estrutura, que possui 181.3 mm de altura e uma distância entre o eixo do motor e o eixo do pêndulo, na posição vertical, de 112 mm. As peças destacadas em vermelho foram produzidas com impressora 3D. O pêndulo e o eixo do encoder, que possuem a mesma cor, têm diâmetros de 8 mm e foram produzidos em aço, as peças em marrom foram fabricadas em madeira, e o braço (cinza) foi produzido em alumínio.

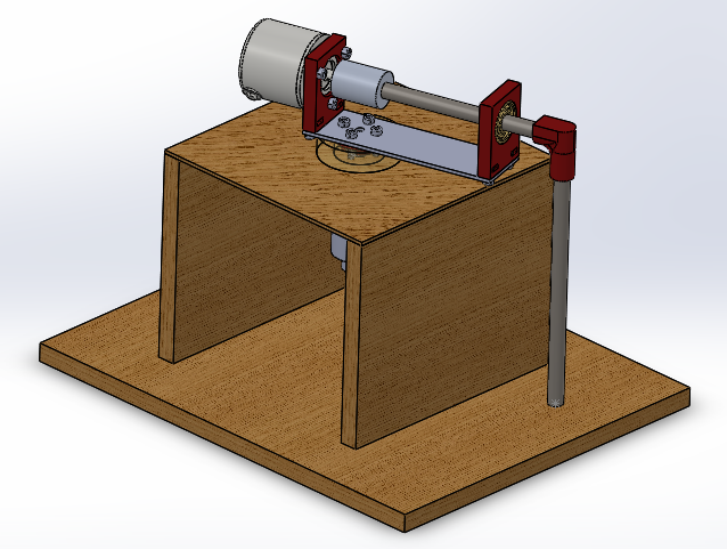


Figura 11: Primeira proposta da estrutura modelada no software SOLIDWORKS

Optou-se por utilizar um rolamento axial para diminuir as cargas sobre o eixo do motor e um acoplamento comercial para unir o eixo do encoder ao eixo horizontal do pêndulo. As características tanto do rolamento, quanto do acoplamento se encontram nas tabelas 4 e 5, respectivamente.

Modelo	51105
Altura	11 mm
Diâmetro interno	25 mm
Diâmetro externo	42 mm

Tabela 4: Especificações do rolamento axial utilizado

Modelo	AC-1925AL
Diâmetro eixo 1	6 mm
Diâmetro eixo 2	8 mm
Altura	25 mm
Diâmetro Externo	19mm
Velocidade Máxima	6.000RPM
Material	Alumínio
Torque Máximo	10 Kg/cm

Tabela 5: Especificações do acoplamento

5.1 Fabricação

Após a impressão das peças em 3D, o corte das chapas de madeira, e dos eixos de aço, notou-se que a fabricação do braço, inicialmente proposta em alumínio, poderia ser

feita com chapa de MDF de 3mm, sem que a perda de rigidez afetasse o desempenho da estrutura.

Com todas as peças fabricadas, a primeira versão da estrutura, mostrada na Figura 12 foi montada e testada. Seu desempenho mecânico foi satisfatório, com o motor sendo capaz de mover o braço com facilidade, e o pêndulo de se mover livremente. Além disso, nenhum dos elementos apresentou folgas ou instabilidades significativas.

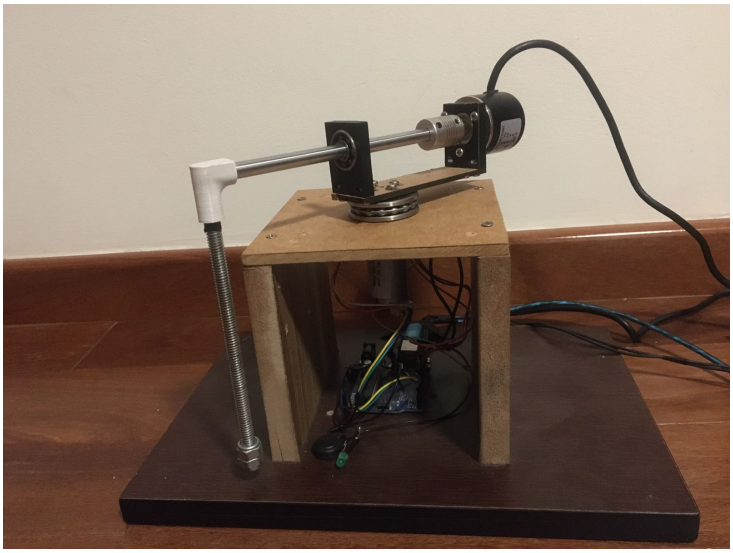


Figura 12: Protótipo da estrutura

O único ponto a se ressaltar é a fixação do cabo do encoder. Uma vez que este apresenta natureza mais rígida do que o esperado, caso na montagem da estrutura ele não seja posicionado adequadamente, ele exercerá um torque resistivo que afeta significativamente o movimento do braço. A Figura 13 exemplifica uma forma de posicionar o cabo para que este problema não ocorra.

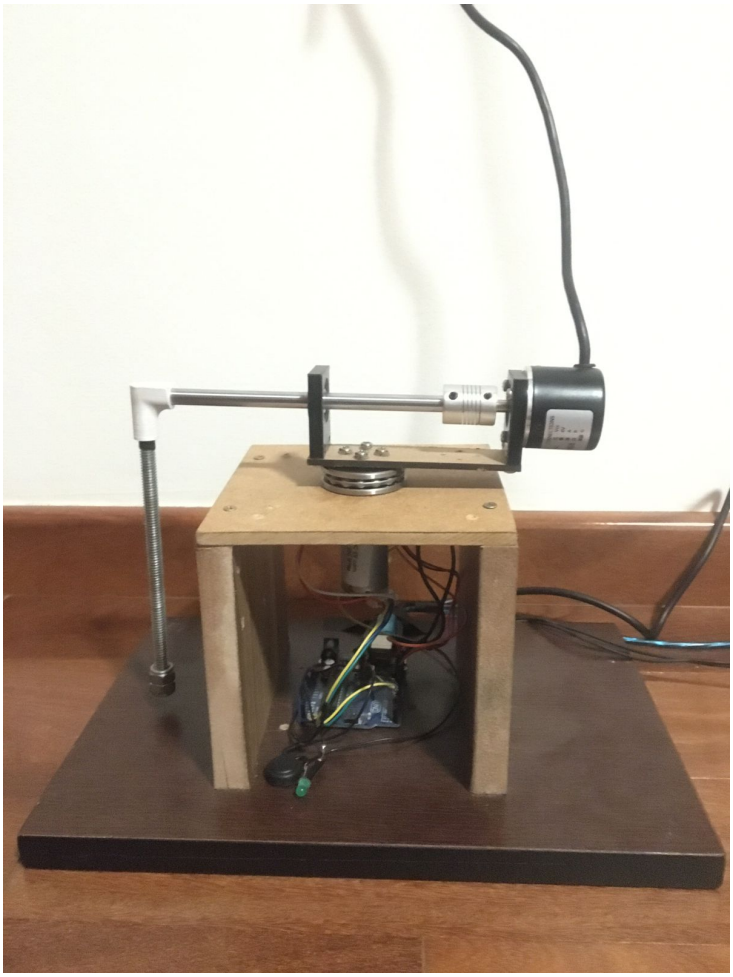


Figura 13: Posição adequada do cabo do encoder

Os desenhos de fabricação de todos os componentes da estrutura se encontram no anexo A.

6 PROJETO ELETRÔNICO

6.1 Escolha de componentes

Com as equações dinâmicas do sistema e com os parâmetros mecânicos definidos, obteve-se o esforço necessário para o controle do pêndulo invertido e consequentemente a relação rotação-torque exigida, sendo assim possível definir o motor a ser utilizado.

As especificações do motor escolhido podem ser observadas na Tabela 6.

Tensão nominal DC	6 V
Velocidade nominal	280 rpm
Corrente nominal	0.13A
Torque de Stall	10 kg.cm
Corrente de Stall	3.2 A
Redução	1:34
Resolução do encoder	341.2

Tabela 6: Especificações do motor selecionado

Na sequência foram realizados testes de leitura dos sinais dos encoders por um *ArduinoUnoR*3, que demonstrou-se eficaz. Dessa forma optou-se pela sua introdução no projeto.

Algumas de suas especificações são ilustradas pela tabela 7.

Microcontrolador	ATmega328
Memória Flash	32Kb
Velocidade de Clock	16MHz
Comunicação	I2C, SPI, UART

Tabela 7: Especificações do *ArduinoUnoR*3

Uma vez que o $ArduinoUnoR3$ não apresenta capacidade computacional suficiente para a implementação de algoritmos de inteligência artificial, definiu-se que o mesmo atuaria recebendo, interpretando e enviando sinais. Dessa forma, a tarefa referente ao processamento foi delegada a um computador, cuja comunicação se dá via serial. O processador utilizado então passou a consistir em um $Intel(R)Core(TM)i7-8750HCPU@2.20GHz$.

Por fim, escolheu-se um módulo de ponte H baseado no CI L298N para servir como driver do motor, e um encoder para ser acoplado ao pêndulo cujas principais características podem ser observadas na Tabela 8.

Modelo	LPD3806-600BM-G5-24C
Resolução	600 PPR
Alimentação	5-24V DC
Rotação máxima	5000 rpm

Tabela 8: Especificações do encoder acoplado ao pêndulo

6.2 Diagrama elétrico

Seguindo as especificações dos componentes bem como os requisitos de projeto, foi possível projetar e construir o diagrama elétrico do sistema, que é ilustrado pela Figura 14.

A partir do diagrama é possível observar que o *Arduino Uno R3* realiza a alimentação elétrica de circuito do encoder individual e do encoder do motor, cujas tensões são de $5V$ e $3.3V$, respectivamente. Além disso, recebe os sinais de ambos os canais dos encoders, que são transmitidos através dos pinos $D2$ a $D5$.

Já a Ponte H recebe a alimentação elétrica de potência do sistema, que é de $6V$, e a transmite ao motor ($M1+$ e $M1-$) por meio dois pinos 2 e 3 ($OUT1$ e $OUT2$). O controle do sentido de rotação do motor é efetuado através de sinais enviados pelos pinos $D7$ e $D8$ do *Arduino Uno R3*, e a velocidade de rotação é controlada pelo nível de tensão atribuído pelo pino $D11$, que atua com PWM.

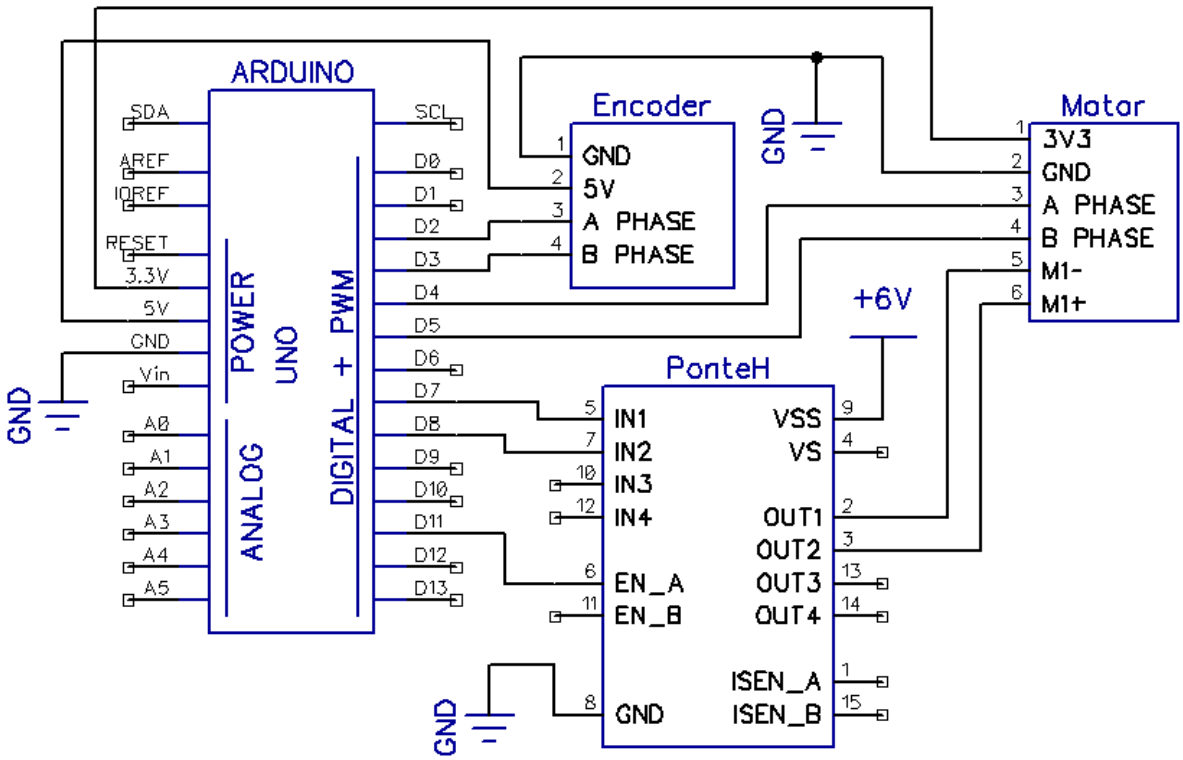


Figura 14: Diagrama elétrico.

7 PROJETO DE CONTROLE

7.1 Simulações

Conforme descrição da metodologia na seção 4, uma vez realizada a modelagem do pêndulo, foi necessário implementar simulações de seu controle. Assim, foi possível estabelecer e validar os requisitos para a consequente definição e caracterização da lista de componentes. Para tal, foi utilizado o auxílio do ambiente de programação gráfica Simulink, integrado ao MATLAB.

No software, com o objetivo de promover as simulações foi inicialmente inserida a dinâmica do pêndulo, conforme modelagem descrita na seção 3.1. As equações não lineares de movimento foram seccionadas em dois blocos principais cujas funções se referem às duas variáveis que representam os graus de liberdade do sistema, α e θ. O diagrama de blocos desenvolvido para descrever a dinâmica pode ser observado na Figura 15.

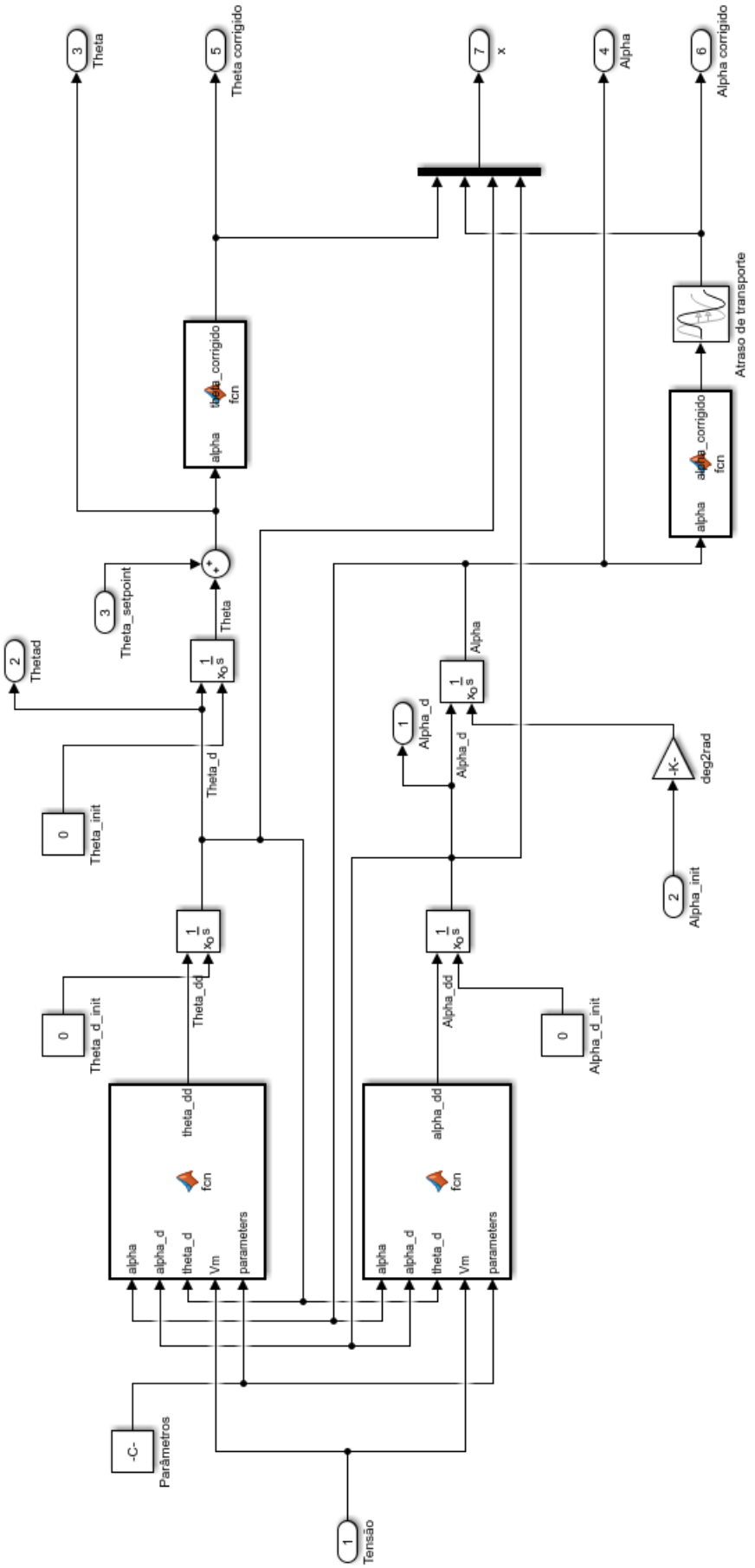


Figura 15: Diagrama de blocos com equações de movimento (Simulink).

É importante ressaltar que o diagrama inicial se baseava no esforço de controle representado pelo torque, uma vez que um dos objetivos era promover a identificação do esforço necessário para a escolha do motor. Uma vez identificado e selecionado o motor foi possível inserir a sua dinâmica no modelo e traduzir o valor de torque para uma tensão, que representa de fato o esforço de controle utilizado.

Dessa forma, as entradas do diagrama consistem nos parâmetros do modelo do pêndulo, das especificações do motor e de seu esforço de controle, representado pelo valor da tensão atribuída. Os parâmetros referentes ao pêndulo foram determinados de acordo com os dados extraídos do modelo da estrutura mecânica, conforme detalhado na seção 5 e expostos na Tabela 3.

As demais funções presentes na Figura 15 se referem a uma correção numérica das variáveis. Já os blocos restantes consistem nas integrações das variáveis e, por fim, foi inserido um atraso no transporte do sinal de α, com o objetivo de representar o atraso de leitura do encoder.

Após a inserção do modelo, foram desenvolvidos e implementados blocos referentes aos algoritmos descritos previamente e essenciais ao controle do pêndulo, o algoritmo de *swing-up* e o de balanceamento na estabilização vertical.

Com respeito ao balanceamento, foi criado um bloco que implementa o controle Regulador Quadrático Linear, cuja matriz Q foi otimizada de forma iterativa com o objetivo de reduzir o tempo de estabilização do pêndulo assim como o esforço de controle. Sua constituição final pode ser observada na equação 7.1. Suas linhas se referem aos estados θ, α, $\dot{\theta}$ e $\dot{\alpha}$, respectivamente em ordem superior a inferior. Para a matriz R foi atribuído o valor 1.

$$Q = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \tag{7.1}$$

Quanto ao controle de *swing up*, foi desenvolvida uma função que implementa a lógica de controle não linear baseada em energia e detalhada na seção 3.3. Assim como o diagrama que representa a dinâmica, a função depende dos parâmetros da estrutura e do

motor. Já o ganho k, analogamente à matriz Q, foi determinado de forma iterativa, e o valor que atingiu o melhor desempenho simulado foi de 20.

A fim de determinar em que instante cada controle é utilizado, foi inserido ainda um bloco *switch* que seleciona o módulo de controle (balanceamento ou *swing up*). A mudança ocorre conforme deflexão do ângulo do pêndulo, optando pelo balanceamento apenas quando inferior a 10°. O diagrama de blocos completo pode ser observado na Figura 16.

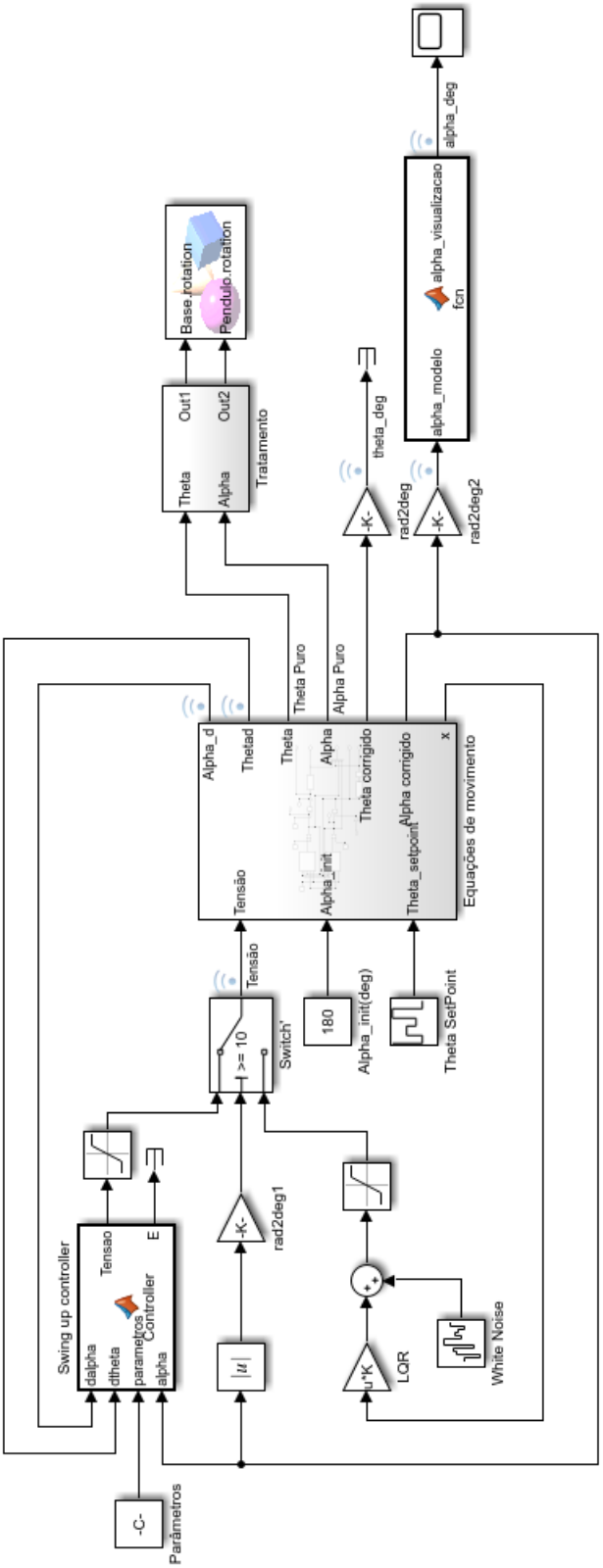


Figura 16: Diagrama de blocos completo (Simulink).

Pode-se notar ainda no diagrama a inclusão de blocos de saturação, de conversão de unidades (radianos para graus) e tratamentos para visualização, com auxílio de um

modelo mecânico simplificado desenvolvido no software SOLIDWORKS. As saturações foram inseridas com o objetivo de regular e evitar picos dos esforços de controle. Por fim, foi incluído um bloco para adicionar um sinal de ruído branco ao modelo, de forma que represente oscilações e imperfeições inerentes ao sistema real.

A seguir são apresentadas nas Figuras 17 a 22 os resultados das simulações realizadas nos quais o valor de α inicial consistiu em 180°, 90° e 10°, respectivamente, sendo 180° ou -180° a posição do pêndulo de equilíbrio estável e 0° a posição de equilíbrio instável. As figuras exibem o sinal de α e a tensão aplicadas para cada simulação efetuada.

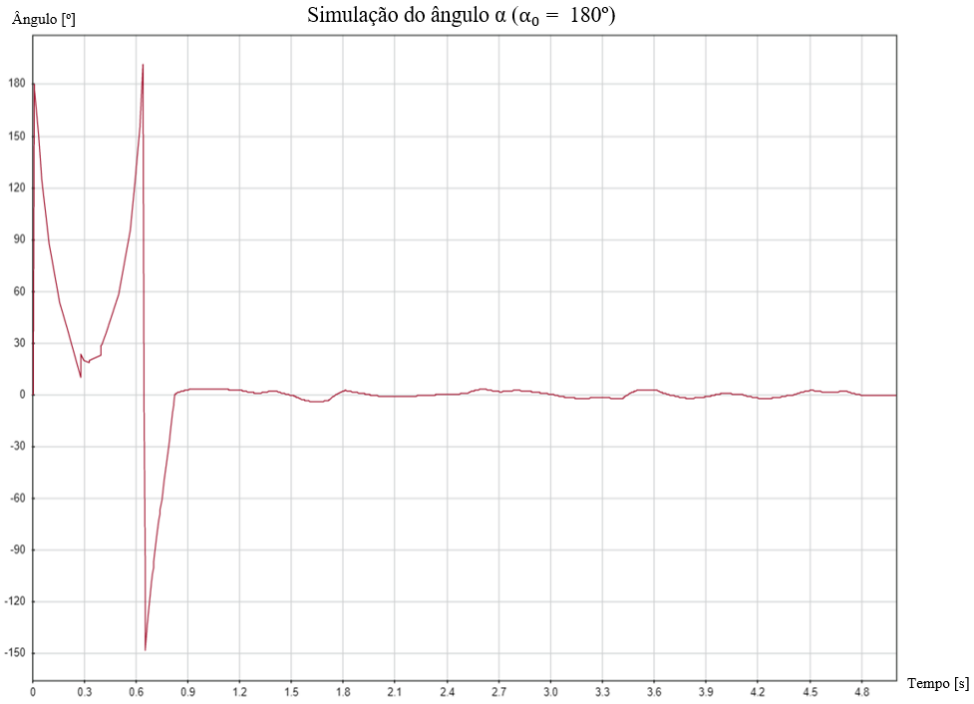


Figura 17: Ângulo do pêndulo na simulação do controle com valor de α inicial de 180°.

Todas as simulações apresentadas decorrem em um período total de 5 segundos. Os picos observados nos gráficos referentes ao sinal de α ocorrem em função da passagem do pêndulo pela posição de equilíbrio estável, no qual o ângulo se altera instantaneamente de 180° para -180°, ou o contrário. Já as irregularidades observadas se devem à presença de ruído branco.

Observa-se que os esforços de controle foram limitados com saturação de $6V$, uma vez que consiste na tensão máxima de atuação do motor selecionado.

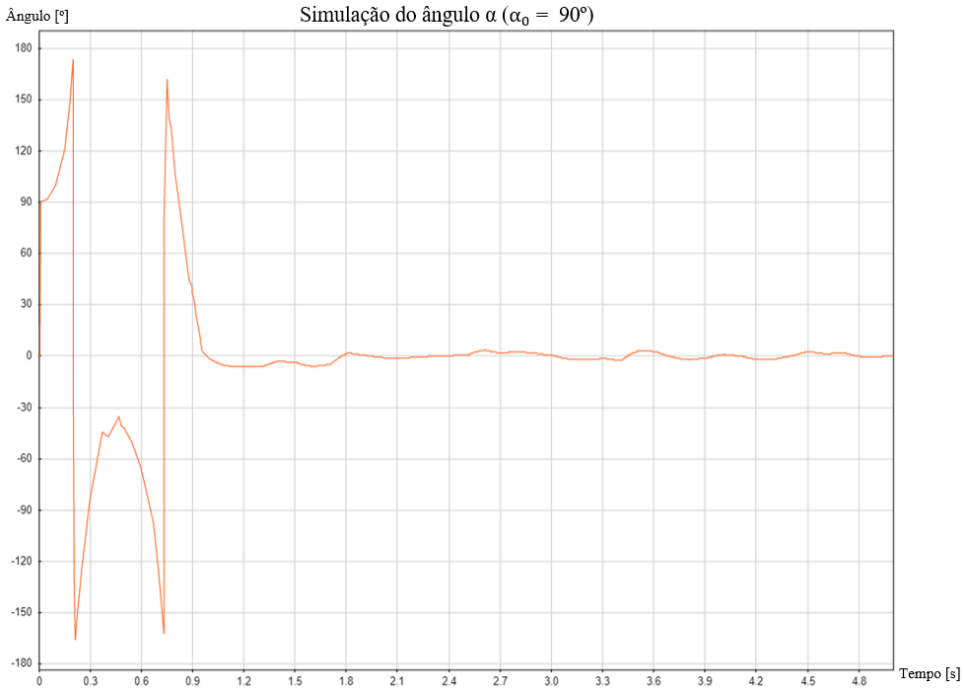


Figura 18: Ângulo do pêndulo na simulação do controle com valor de α inicial de 90°.

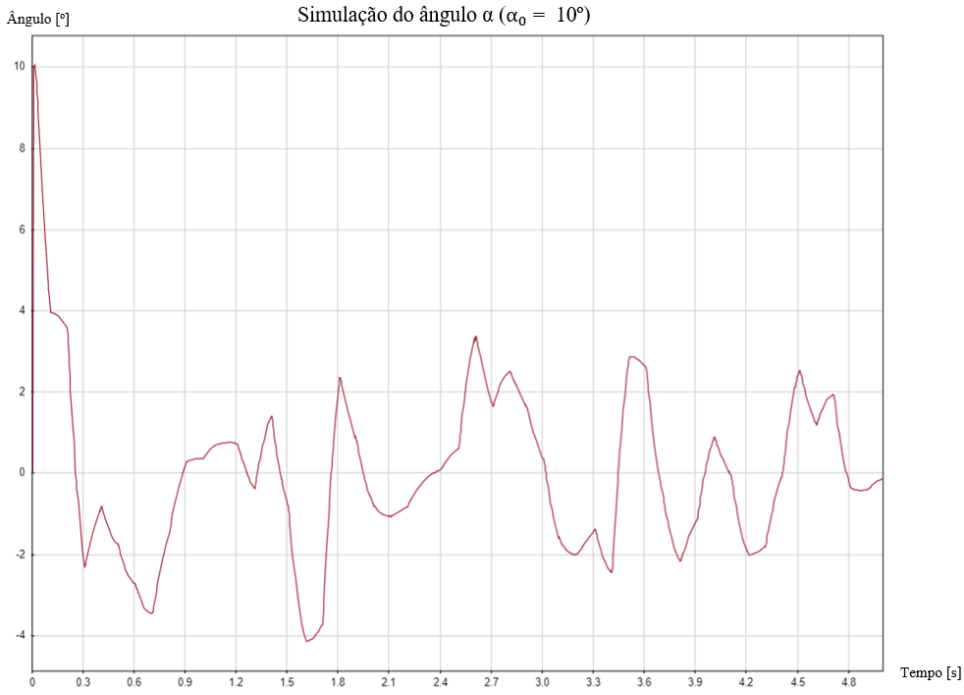


Figura 19: Ângulo do pêndulo na simulação do controle com valor de α inicial de 10°.

Conforme ilustrado pelas Figuras 17 e 18, a fim de atingir a região de equilíbrio instável (entre 10° e -10°), foi necessária apenas uma oscilação do braço para α inicial de 180° e 90°. Ambos controles de *swing up* exigiram cerca de 1 segundo para que o bloco de *switch* efetuasse a troca para o LQR, já acionado desde o início na simulação com α

inicial de 10°. Pela Figura 19 nota-se um rápido equilíbrio com a subsequente presença do ruído branco.

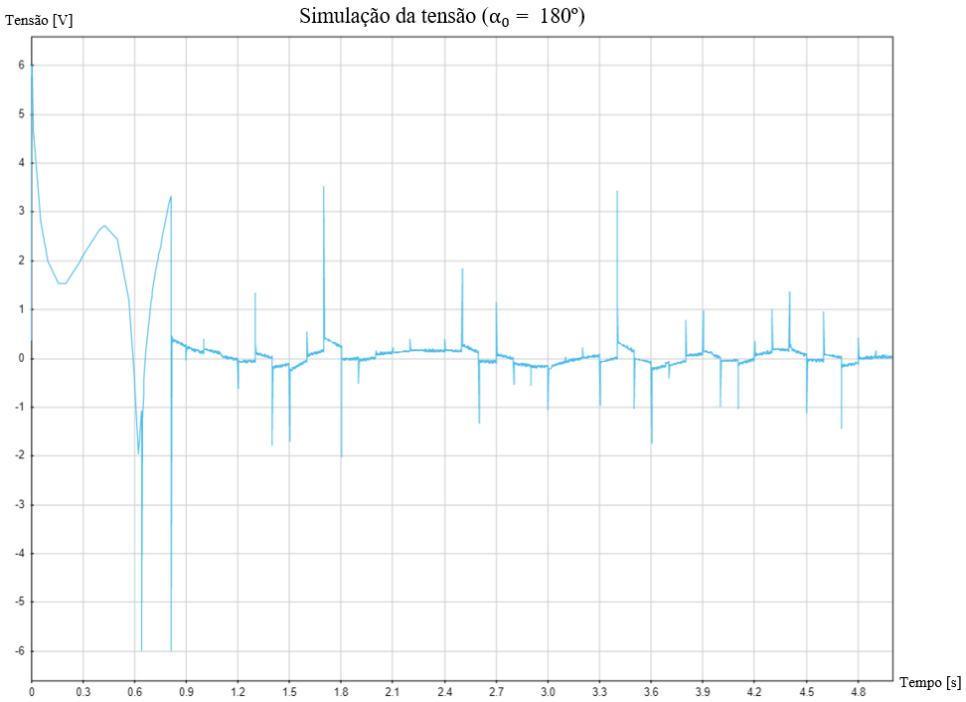


Figura 20: Tensão do motor na simulação do controle com valor de α inicial de 180°.

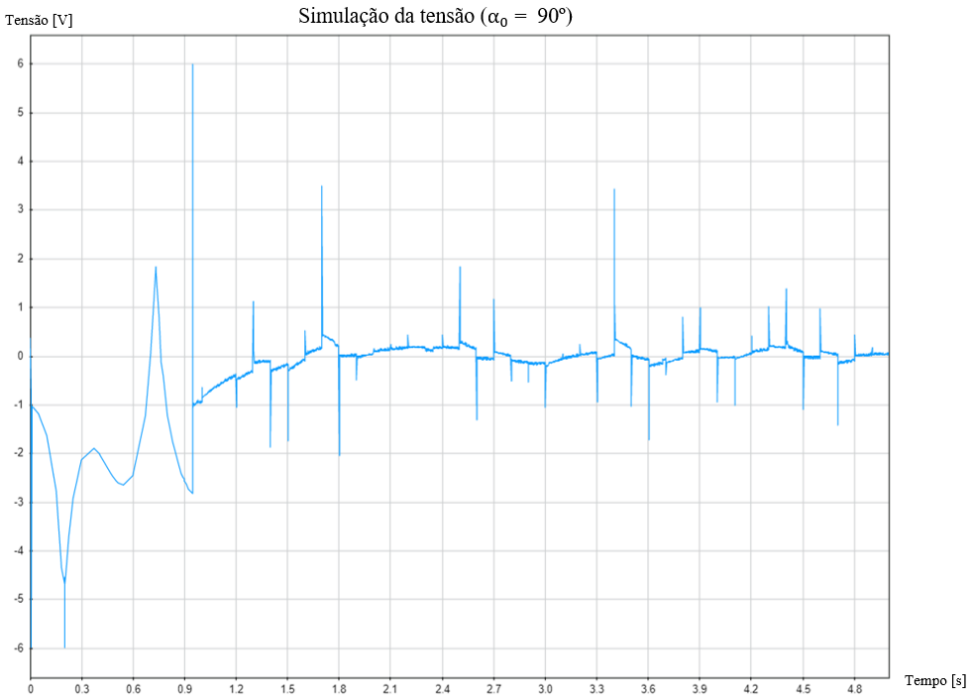


Figura 21: Tensão do motor na simulação do controle do pêndulo com valor de α inicial de 90°.

Os sinais de tensão das Figuras 20 e 21 apresentam picos nos instantes em que α

= 180°, o que é coerente com a função de controle *swing up* implementada. Além disso atingem novamente o limite de $6V$ quando realizada a mudança de controle do pêndulo.

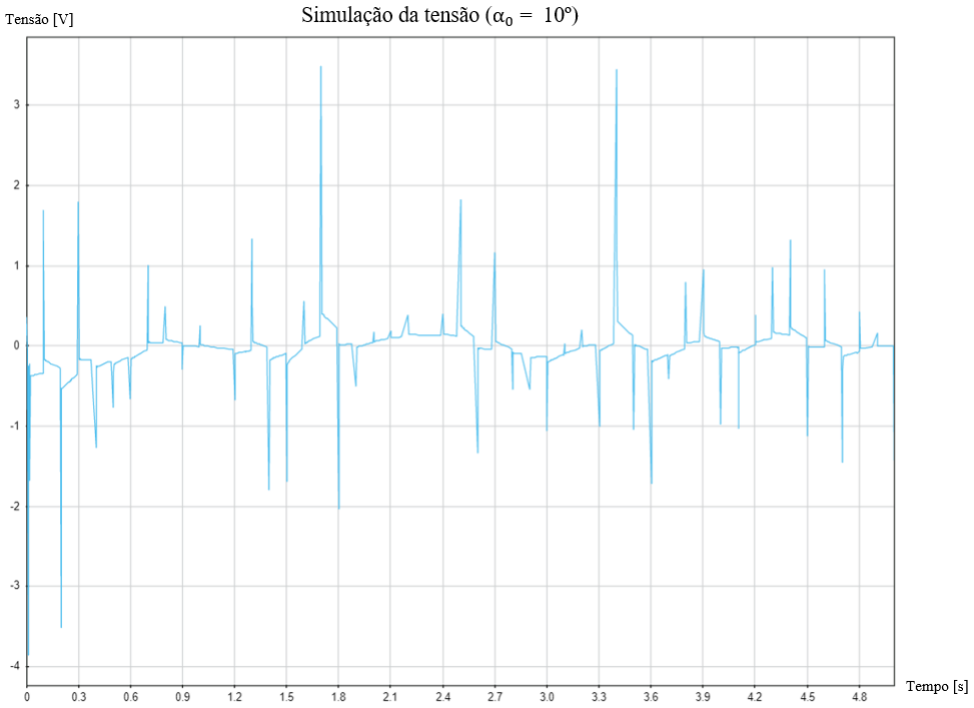


Figura 22: Tensão do motor na simulação do controle do pêndulo com valor de α inicial de 10°.

Nota-se pela figura 22, no qual utilizou-se apenas o controle de balanceamento (LQR), que as tensões aplicadas foram inferiores, não atingindo em qualquer instante o limite de $6V$ estabelecido pelo motor.

8 PROJETO COMPUTACIONAL

8.1 Interface: Computador - Arduino

Como já mencionado anteriormente, utilizou-se um Arduino Uno R3 para ler e armazenar as posições e velocidades dos dois encoders, e controlar a tensão imposta nos terminais do motor DC através de uma ponte H baseada no chip L298N. Além disso, foi utilizado um computador para processar as leituras dos encoders e de acordo com o algoritmo utilizado, obter a tensão desejada para ser imposta nos terminais do motor.

Dessa forma tornou-se necessário estabelecer uma comunicação entre o computador e o Arduino, a qual se deu por meio do uso de comunicação serial através de um cabo USB e da arquitetura mestre escravo.

O Arduino, exercendo o papel de escravo, ao mesmo tempo que conta os pulsos de ambos os encoders através de rotinas de interrupção, aguarda que uma nova mensagem via serial seja recebida. Esta mensagem é composta exclusivamente por um valor de -255 a 255, cujo módulo será proporcional à tensão imposta ao motor DC, e o sinal determinará a polaridade, ou a direção que o motor irá assumir.

Após receber tal mensagem e impor a respectiva tensão aos terminais do motor através da ponte H, o Arduino retorna outra mensagem serial para o computador sendo composta por quatro números: dois com a contagem de pulsos atuais de cada encoder, e dois com a variação temporal de tais contagens. Com isso, torna-se possível determinar os ângulos e velocidades atuais de cada encoder.

O *script* desenvolvido e implementado para se comunicar com o Arduino e processar as informações necessárias foi escrito em Python e dividido em classes. A classe **Encoder** é responsável pela comunicação e seu código, assim como os demais utilizados neste projeto, pode ser observado no **repositório do projeto**.

8.2 Implementação Controle Clássico

O método de controle clássico selecionado e implementado foi o Proporcional Derivativo (PD), porém, como se deseja controlar tanto a posição do pêndulo quanto a do braço, optou-se por utilizar dois controladores em paralelo. Dessa forma, cada controlador recebe a posição e a velocidade da respectiva estrutura e calcula o esforço de controle necessário individualmente. Em seguida, ambos os esforços calculados são somados para se obter a tensão que se deseja impor ao motor DC.

A classe utilizada para implementação dos dois controladores PD é a classe **PID**. E o *script* responsável pela implementação dos dois controladores em paralelo e por toda a dinâmica do teste é o **CC-PID-Pendulum.py**.

8.3 Implementação Swing-Up

A implementação do algoritmo de *Swing-Up* se deu a partir da equação 3.16. Para isso, inicialmente definiu-se um laço que fosse interrompido caso o módulo da posição angular do pêndulo seja inferior a 10°, assim, o pêndulo se encontraria próximo ao equilíbrio. Neste laço executa-se o cálculo da equação, em que realiza-se o produto entre um ganho e o sinal da multiplicação do cosseno da posição angular pela velocidade angular do pêndulo.

Vale notar que o ganho é associado ao valor de PWM a ser enviado ao motor, assim, conforme a posição e direção de rotação, a entrada de controle atribuída ao motor oscila apenas entre dois valores, o ganho positivo e o ganho negativo. O *script* responsável pela implementação do algoritmo é o **SwingUp.py**.

8.4 Implementação Q-Learning

A sequência de eventos do início ao fim de um episódio implementada no *script* responsável pela execução do algoritmo *Q-Learning* pode ser observada no fluxograma da Figura 23.

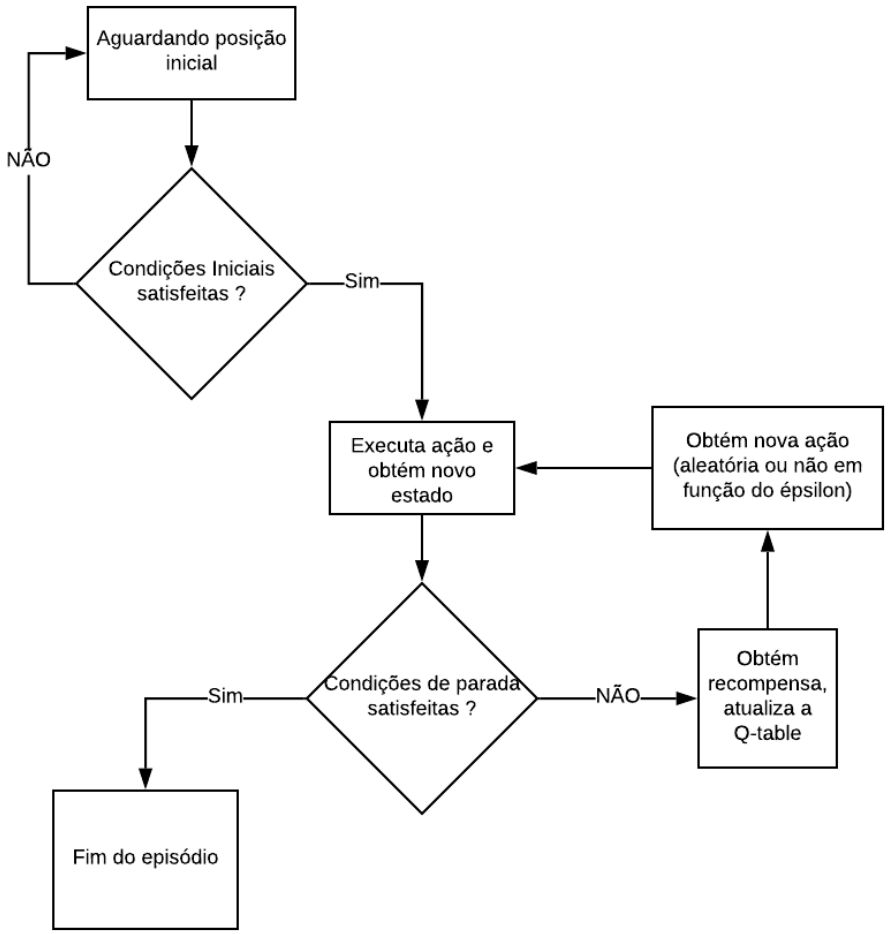


Figura 23: Fluxograma da lógica de um episódio do algoritmo *Q-Learning*

Como se pode observar, no inicio de cada episódio o algoritmo aguarda até que determinada condição seja satisfeita, sendo ela: o módulo do ângulo do pêndulo ser inferior a 15 graus. Portanto, para se iniciar um episódio é necessário levar manualmente o pêndulo para esta posição.

Uma vez iniciado, o episódio apenas se encerrará caso as seguintes condições sejam satisfeitas: o módulo da posição do pêndulo for superior a 15 graus ou o módulo da posição do braço for superior a 180 graus.

Durante a execução do episódio, pode-se observar a realização de um ciclo, que se inicia com a execução de uma ação e a consequente obtenção de um novo estado. Em seguida verifica-se se a condição de parada foi satisfeita, em caso negativo, o cálculo da recompensa e a atualização da *Q-Table* são executados e por fim é realizada a obtenção de uma nova ação.

A decisão de se obter uma ação aleatória ou uma ação baseada na *Q-Table* segue uma probabilidade definida pela variável epsilon, de forma que a cada episódio, a variável é decrementada linearmente, diminuindo assim a probabilidade de se tomar uma ação aleatória.

A classe responsável por implementar os métodos utilizados pelo algoritmo *Q-Learning* é a **PartialQLearning** e o *script* responsável por toda a dinâmica de aprendizagem através deste método é o **PartialQL-Pendulum.py**.

8.4.1 Decisões de projeto

Conforme estabelecido pelo algoritmo *Q-Learning*, em sua implementação é necessário definir suas principais atribuições, tais como a função de recompensa a ser utilizada, a forma de decaimento da variável epsilon ao longo dos episódios e outros hiperparâmetros. A seguir, explicita-se estas decisões e as justificativas de cada uma.

Apesar de haver formas alternativas de se realizar o decremento da variável epsilon ao longo dos episódios, como levando em consideração os resultados parciais do aprendizado, por exemplo, o mais comum é decrementá-la de forma linear. Neste projeto optou-se pelo decremento linear, de forma que epsilon no começo do treinamento é igual a 1, e ao se executar o episódio que representa 80% do treinamento, epsilon é igual a zero. Assim os 20% finais dos episódios são realizados tomando ações exclusivamente não aleatórias.

Para a escolha da função recompensa, ou seja, a função que avalia a ação tomada, testou-se algumas alternativas, dentre elas: funções que penalizavam de forma linear o afastamento do pêndulo de seu ponto de equilíbrio, e incentivavam a aproximação, levando em consideração posição e velocidade angular; funções binárias que consideravam apenas a posição do pêndulo, recompensando positivamente quando o pêndulo se localizava dentro de uma determinada distância do seu ponto de equilíbrio; e funções incrementais, que basicamente recompensavam cada ação tomada de acordo com o tempo decorrido desde o começo do episódio.

Por conta de um número grande de ações serem tomadas em um curto intervalo de tempo e a inércia do sistema ser relativamente grande, a avaliação de uma ação em função dos estados correntes do sistema se torna uma tarefa complexa.

Funções que avaliam o conjunto de todas as ações realizadas desde o início do episódio se tornam mais simples e justas, apesar de diminuírem a velocidade de aprendizado, uma vez que se trata de uma recompensa média.

Para contornar tal problema, optou-se por uma função recompensa que leva em consideração o estado atual da planta, e o estado após um número definido de iterações. Ou seja, a recompensa é dada à iteração atual, de acordo com o estado atual e um estado futuro do pêndulo.

O algoritmo *Q-Learning*, como se sabe, precisa de estados e ações discretas, de forma que quanto maior o número de ações e de estados, maior será a *Q-Table* e maior o tempo necessário de aprendizagem para atingir os objetivos desejados.

Diante disso, seria inviável haver um estado para cada número de pulsos em uma volta do encoder, assim como uma ação para cada possível valor de PWM. Por isso, optou-se por aumentar a discretização com auxílio de um controlador PD, a fim de diminuir ao máximo o tamanho da *Q-Table*, sem que se perdesse a capacidade de estabilização do pêndulo.

Para esta tarefa, utilizou-se um controlador PD capaz de estabilizar o pêndulo através de estados e ações não discretizados, de forma que aumentou-se a discretização tanto dos estados quanto das ações até um grau em que ainda fosse possível estabilizar o pêndulo através do controlador. Obtendo assim a discretização utilizada no projeto.

8.4.2 Problemas e soluções adotadas

8.4.2.1 Frequência de ações e aleatoriedade

Como já mencionado anteriormente, a plataforma desenvolvida neste trabalho executa muitas ações em um pequeno intervalo de tempo, ao mesmo tempo que a inércia do sistema é relativamente grande. Em outras palavras, a influência de cada ação instantânea no movimento do pêndulo é muito pequena, o que dificulta sua avaliação.

Esta característica somada à tomada de ações aleatórias, dificulta ainda mais a avaliação de cada ação, especialmente na parte inicial do treinamento, quando epsilon ainda é alto. Dessa forma, para um mesmo estado, diferentes ações podem ser tomadas em um curto intervalo de tempo.

A solução adotada para este problema foi escolher aleatoriamente, no início de cada episódio, uma ação para cada estado. Assim, para um determinado estado, quando houver a necessidade de se executar uma ação aleatória, sempre se executará a ação previamente associada.

Esta solução resolve parcialmente o problema, uma vez que, quando epsilon for próximo de 0.5, para um mesmo estado, haverá 50% de chance de se executar uma ação não aleatória (a ação com maior *Q-Value*), e 50% de se executar uma ação aleatória (associada previamente ao estado), de forma que em um curto intervalo de tempo ambas podem ser executadas alternadamente, dificultando a avaliação das ações.

Diante disso, optou-se por além da solução adotada anteriormente, no início de cada episódio determinar quais estados implicarão ações aleatórias (escolhidas previamente) e quais implicarão ações não aleatórias (com maior *Q-Value*). Dessa forma, a probabilidade de um episódio ser escolhido para executar uma ação aleatória ou não aleatória é determinada por epsilon.

8.4.2.2 *Q-Table* simétrica

Outro problema enfrentado foi o tempo de aprendizado que se mostrou demasiadamente grande. Uma medida tomada para diminuí-lo foi considerar que a *Q-Table* final seria simétrica, e esta hipótese é coerente uma vez que, desconsiderando imperfeições mecânicas, o problema do pêndulo invertido é simétrico.

Portanto, para se determinar o estado do pêndulo, considerou-se as variáveis associadas ao braço (como anteriormente), o módulo do ângulo do pêndulo em relação à vertical, e sua velocidade angular, de forma que esta é considerada positiva quando o pêndulo se aproxima da posição de equilíbrio instável e negativa quando se afasta.

Desta forma, um estado desta nova abordagem será equivalente a dois estados da abordagem anterior, o que reduz o tamanho da *Q-Table* pela metade, implicando redução também do tempo de aprendizagem.

8.4.2.3 Descarte da posição do braço

Ao longo do desenvolvimento, notou-se também que não era relevante determinar a posição na qual o braço se encontra no momento de equilíbrio do pêndulo. Dessa forma,

desconsiderou-se esses estados, reduzindo em uma dimensão a *Q-Table* e agilizando o processo de aprendizagem.

8.4.2.4 Adição de botão e LED

Durante os testes e treinamentos, constatou-se que alguns pulsos do encoder do pêndulo estavam sendo perdidos, apesar do Arduino realizar esta tarefa através de rotinas de interrupção e não executar outras tarefas complexas no seu laço principal.

Por conta disso, dois componentes foram adicionados ao circuito: um LED e um botão. Assim, com uma alteração no *script* do Arduino, uma vez que o pêndulo se encontre na posição de equilíbrio estável, o LED acende. Dessa forma, passou a ser possível identificar a perda de pulsos e recalibrar a posição do pêndulo através do botão adicionado.

8.5 Implementação controlador híbrido

Como alternativa ao modelo baseado exclusivamente no algoritmo *Q-Learning*, desenvolveu-se também um modelo híbrido, que utiliza tanto inteligência artificial, quanto controle clássico para se obter o esforço de controle desejado.

Para o desenvolvimento do controlador híbrido, utilizou-se um modelo de inteligência artificial similar ao descrito na seção anterior, porém que recebe como entrada apenas a posição do pêndulo.

Em paralelo, executa-se um controlador similar ao descrito na seção 8.2, porém que não recebe como entrada a posição do pêndulo, ou que possui a constante relacionada a este estado nula.

Por fim, ambos os esforços de controle calculados são somados para se obter a tensão que será imposta ao motor DC.

9 RESULTADOS

9.1 Swing-Up

Os resultados obtidos para a realização da dinâmica de *Swing-up* através do método descrito na seção 8.3, podem ser observados na Figura 24.

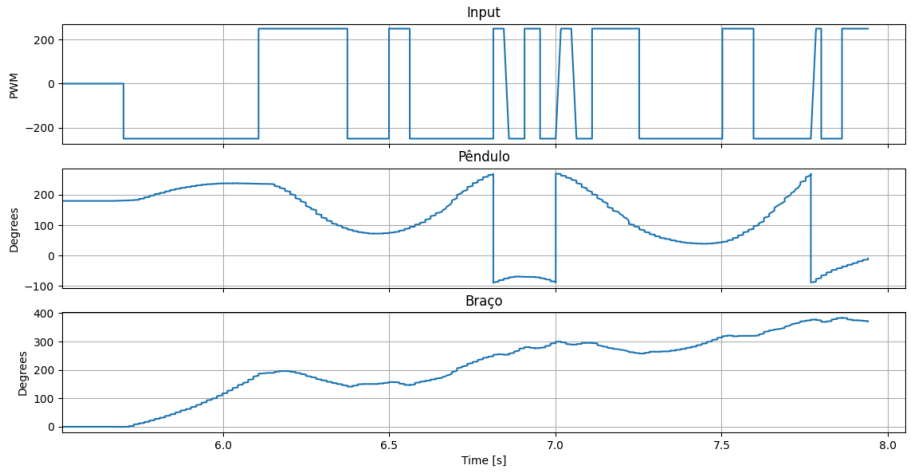


Figura 24: Desempenho obtido na realização da dinâmica de *SwingUp*

Como se pode observar, para o pêndulo chegar na posição de 0 graus, foram necessários aproximadamente 8 segundos.

9.2 Controle clássico

Através da implementação do controlador mencionado na Seção 8.2, ou seja, dois controladores Proporcionais Derivativos que trabalham em paralelo, obteve-se a resposta apresentada pela Figura 25. Pode-se observar um comportamento do pêndulo pouco oscilatório em relação a outros resultados que serão mostrados a seguir. Além disso nota-se o sucesso do controlador em levar o pêndulo até uma posição de equilíbrio estático e o braço até uma posição próxima à zero graus.

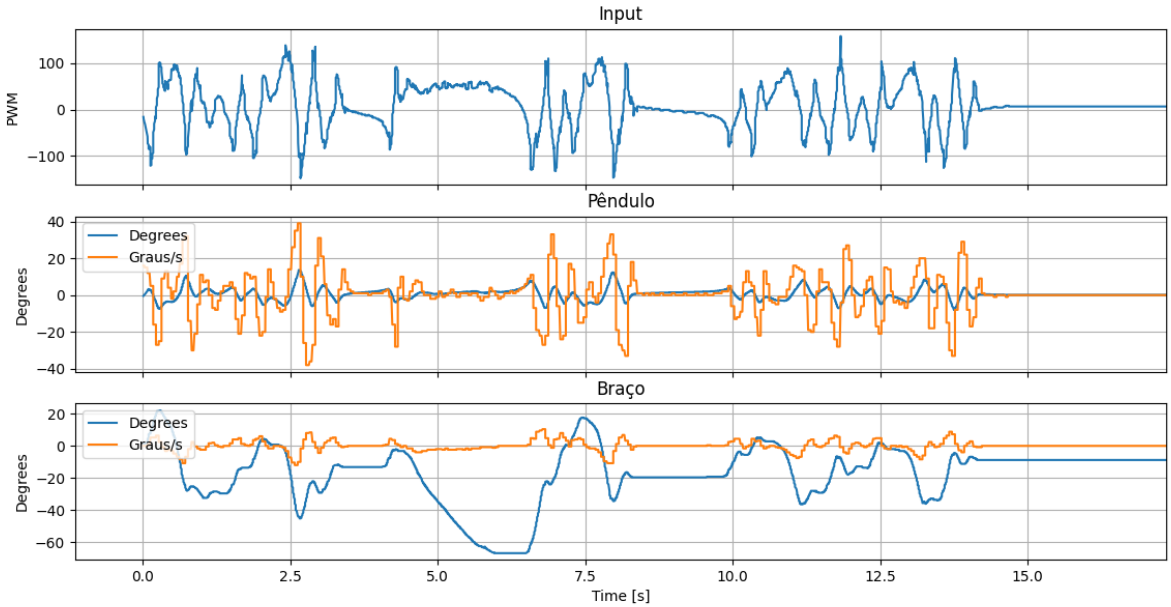


Figura 25: Desempenho obtido através do duplo controlador PD

Vale ressaltar que este resultado foi obtido com os controladores sendo configurados com as seguintes constantes: $P_{Pêndulo} = 15$, $D_{Pêndulo} = 1.6$, $P_{Braço} = 1$, $D_{Braço} = 12$.

9.3 Inteligência Artificial

A implementação de controladores baseados exclusivamente no algoritmo *Q-Learning* não trouxeram resultados satisfatórios, conforme discutido a seguir. Por isso, optou-se pelo desenvolvimento do controlador híbrido, descrito na Seção 8.5.

Uma vez que este ainda possui parte de seu comportamento baseado em inteligência artificial, é possível efetuar a comparação entre controladores proposta neste trabalho.

9.3.1 Q-Learning

A Figura 26 ilustra o desempenho de alguns modelos desenvolvidos de controladores baseados exclusivamente no algoritmo *Q-Learning*, de forma que cada um possui uma função de recompensa e hiperparâmetros específicos.

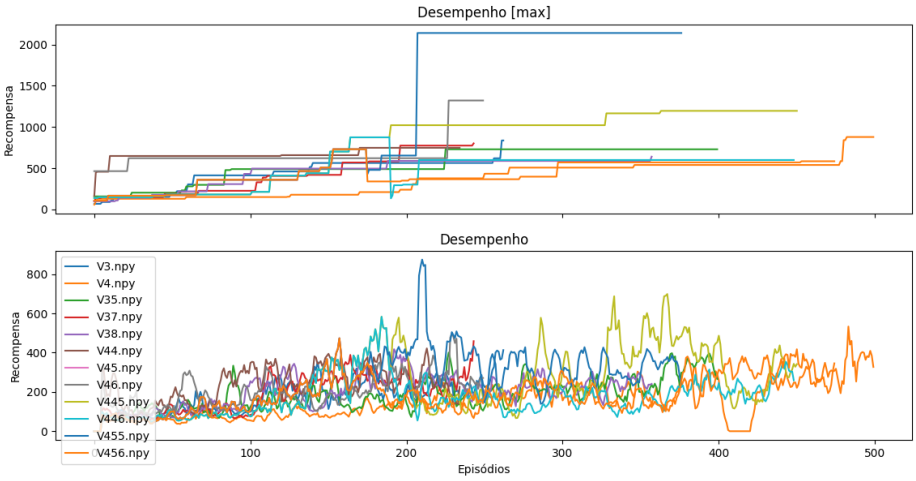


Figura 26: Desempenho de diversos modelos testados

Na Figura 26, o eixo Y expressa a quantidade de iterações realizadas até o episódio se encerrar, ou seja, expressa um índice proporcional ao tempo de duração do episódio. Diante disso, nota-se que de forma geral existe um aumento no desempenho de cada episódio ao longo dos treinamentos, porém nenhum modelo se destaca ou obtém resultados comparáveis aos obtidos com o controlador PD.

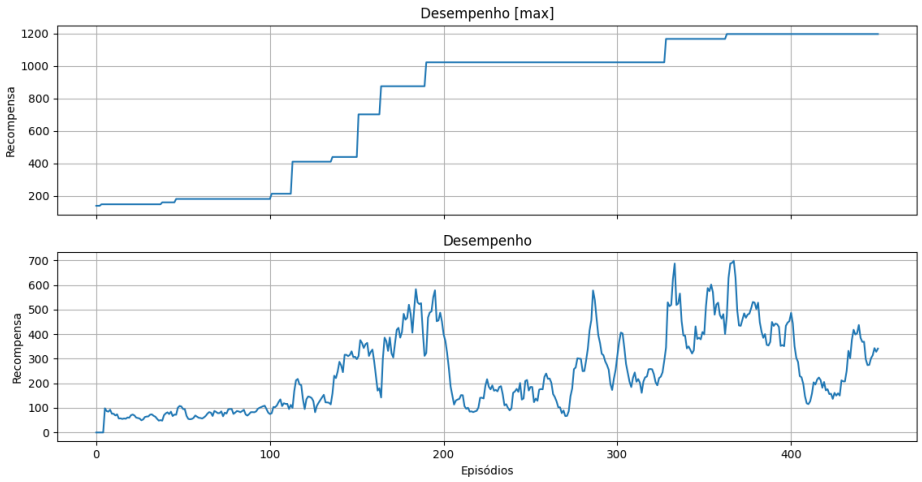


Figura 27: Desempenho do modelo V445

Tomando como exemplo o modelo V445, destacado na Figura 27, nota-se com mais clareza que seu desempenho entre o episódio inicial e o centésimo episódio é pior do que entre os episódios 100 e 200. O que mostra uma evolução no treinamento. Porém logo após o episódio 200 nota-se uma queda abrupta no desempenho, fato que se repete após o episódio 400. Estes resultados evidenciam a dificuldade deste algoritmo em evoluir e obter resultados comparáveis aos obtidos com o controlador PD.

Como o desempenho deste modelo foi destacado com fim de exemplo, destaca-se também sua função de recompensa explicitada em forma de tópicos a seguir:

- Se após 25 iterações, o módulo da posição do pêndulo for menor do que 4 graus e o módulo da velocidade do pêndulo for menor do que 25 graus por segundo, a recompensa atual será: $0.7 - 0.125 * |PP|$. Onde PP é a posição do pêndulo em graus após 25 iterações.
- Se após 25 iterações, o módulo da posição do pêndulo for menor do que 4 graus e o módulo da velocidade do pêndulo for maior do que 25 graus por segundo, a recompensa atual será: $1.5 - 0.125 * |PP| - 0.04 * |VP|$. Onde VP é a velocidade do pêndulo em graus por segundo após 25 iterações.
- Se após 25 iterações, o módulo da posição do pêndulo for maior do que 4 graus, mas inferior ao módulo da posição atual (pêndulo se aproximando da posição de equilíbrio), e o módulo da velocidade for menor do que 25 graus por segundo, a recompensa será: 0.2.
- Se após 25 iterações, o módulo da posição do pêndulo for maior do que 4 graus, mas inferior ao módulo da posição atual (pêndulo se aproximando da posição de equilíbrio), e o módulo da velocidade angular for maior do que 25 graus por segundo, a recompensa será: $1 - 0.04|VP|$.
- Se o módulo da posição do pêndulo após 25 iterações for superior ao módulo da posição atual (pêndulo se afastando da posição de equilíbrio), a recompensa será: $1 - |PP| * 0.25$.

9.3.2 Q-Learning híbrido

A Figura 28 ilustra o desempenho obtido através do algoritmo híbrido. Destaca-se que ela possui as mesmas métricas utilizadas nas figuras apresentadas anteriormente, ou seja, no eixo X tem-se os episódios do treinamento, e no eixo Y o índice de desempenho obtido em cada episódio.

Diante disso, é possível notar que a velocidade de aprendizado em comparação ao modelo anterior foi maior, e a qualidade dos resultados melhor, uma vez que no episódio 133 já se obteve um índice de desempenho na ordem de 6000, o que em termos práticos, significa que o objetivo do modelo foi atingido.

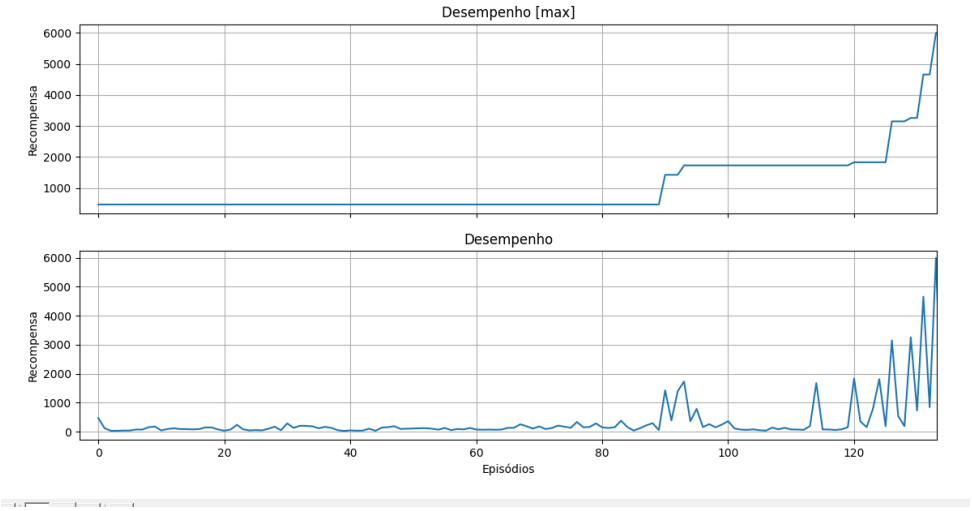


Figura 28: Desempenho obtido através do controlador híbrido

A seguir, com o intuito de explicitar a evolução dos resultados no processo de aprendizagem, e apresentar tal processo com mais detalhes, destaca-se o comportamento do pêndulo durante quatro episódios deste processo: Na Figura 29, em um estágio inicial da aprendizagem, ilustra-se o comportamento do pêndulo no episódio 3. Na Figura 30, ilustra-se o comportamento no episódio 90.

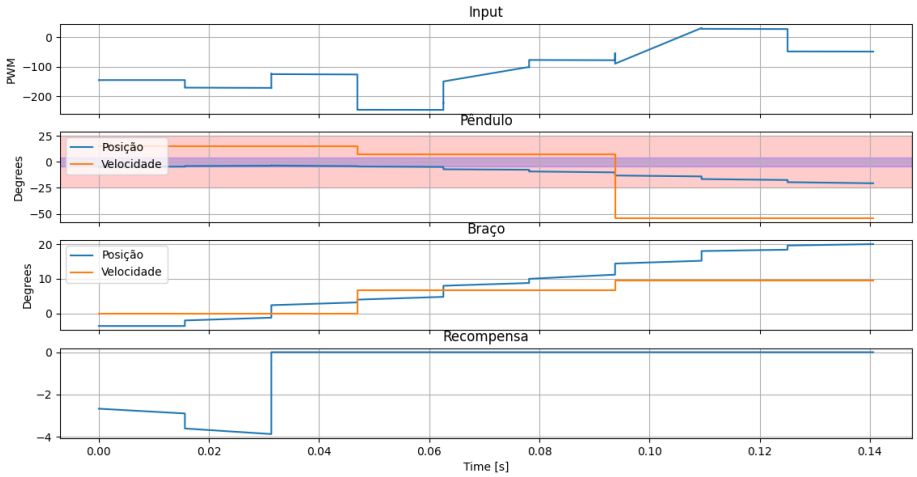


Figura 29: Desempenho controlador híbrido no episódio 3

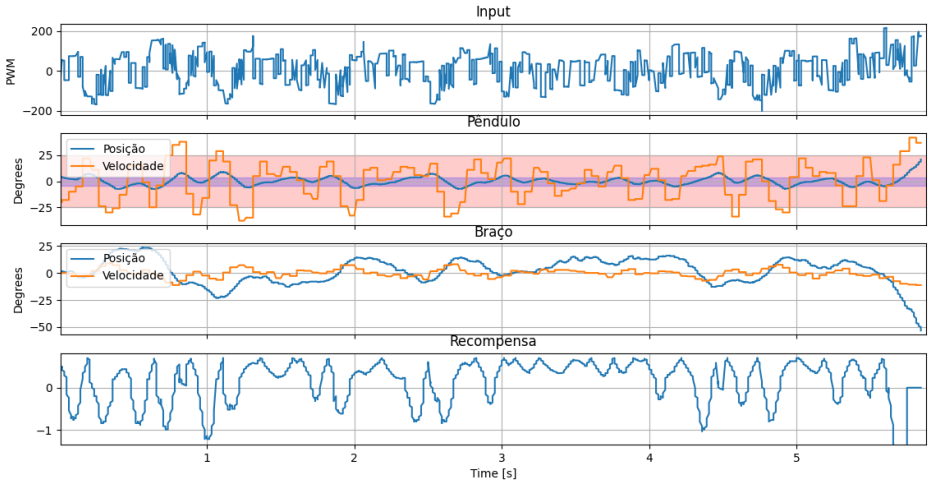


Figura 30: Desempenho controlador híbrido no episódio 90

Na Figura 31, em um estágio final do treinamento, ilustra-se o desempenho no episódio 126. Por fim, a Figura 32 apresenta o desempenho do pêndulo no último episódio do treinamento, que durou aproximadamente 25 segundos e precisou ser interrompido manualmente.

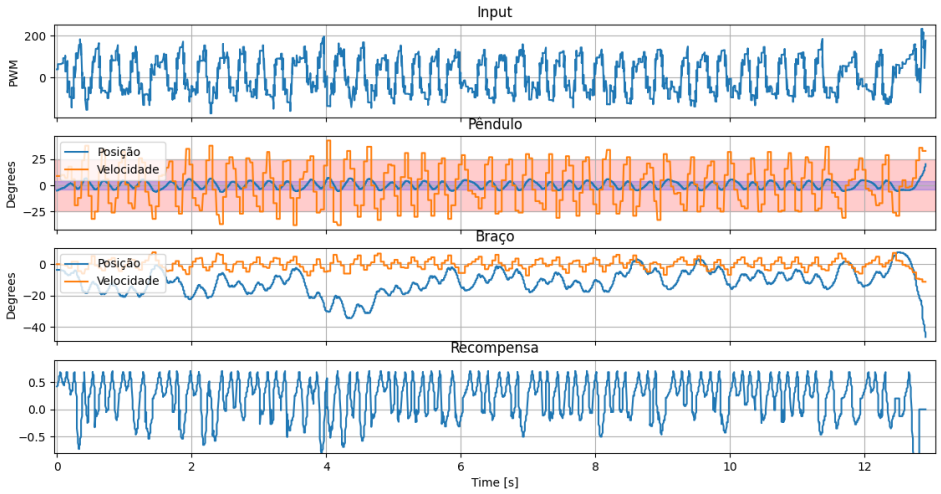


Figura 31: Desempenho controlador híbrido no episódio 126

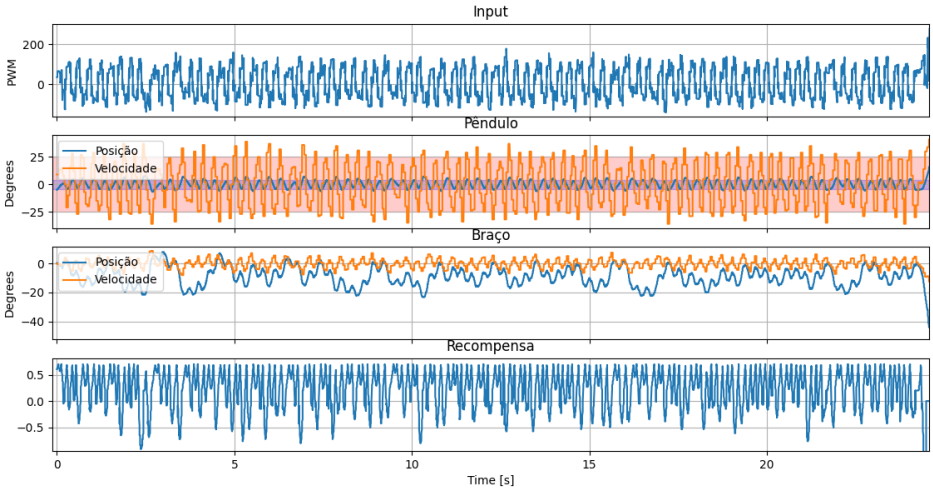


Figura 32: Desempenho controlador híbrido no episódio 133

Nota-se que durante o último episódio, o pêndulo se manteve na maior parte do tempo na região entre 4^o e -4^o, e a velocidade angular entre $25^o/s$ e $-25^o/s$, devido à função de recompensa utilizada, descrita na seção 9.3.1. Destaca-se também que as recompensas dadas ao longo do último episódio são majoritariamente maiores do que zero, ao contrário dos episódios 3, 90 e 126 destacados nas outras figuras.

10 DISCUSSÃO

Conforme apresentado no Capítulo 9, o algoritmo utilizado para a execução da dinâmica de *swing-up* foi capaz de levar o pêndulo da posição de equilíbrio estável até a posição de equilíbrio instável.

Porém, uma vez que a plataforma desenvolvida não possui a capacidade de rotacionar o braço indefinidamente, buscou-se impor restrições em relação aos ângulos máximos de giro desta peça. Como consequência, o algoritmo passou a falhar em sua tarefa, indicando que apesar de ser capaz de executar a rotina de *swing-up*, não se comporta bem quando certas restrições são impostas.

Já em relação ao controlador baseado exclusivamente no algoritmo *Q-Learning*, apesar de demonstrar certa evolução no processo de aprendizagem, não foi capaz de equilibrar o pêndulo indefinidamente, como os controladores híbrido e duplo-PD.

Acredita-se que este insucesso se deu majoritariamente por conta da não utilização de uma função de recompensa adequada, e em menor escala, por não se ter utilizado hiperparâmetros adequados.

No processo de criação e teste de diferentes modelos, notou-se uma melhora significativa no desempenho do algoritmo *Q-Learning*, quando se passou a avaliar a iteração atual com base em um estado futuro. Assim, superou-se a dificuldade imposta pela inércia da planta.

Porém, acredita-se que só esta medida não seja suficiente para avaliar de forma justa e precisa o desempenho de determinada iteração, sendo necessário ainda levar em conta outros fatores que não se conseguiu definir neste projeto.

Além disso, notou-se que os hiperparâmetros, apesar de não terem sido muito explorados neste trabalho, influenciam uma parcela considerável do processo de aprendizagem,

sendo responsáveis por acelerá-lo ou torná-lo mais lento, por exemplo.

Em relação à comparação entre os controladores híbrido e duplo-PD, destaca-se inicialmente o comportamento da posição do pêndulo. As Figuras 33 e 34 destacam 10 segundos dos testes realizados com tais controladores.

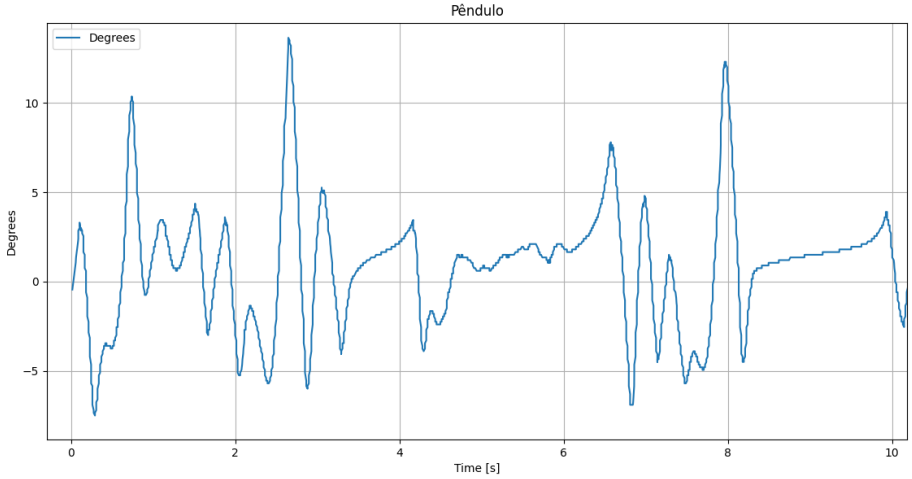


Figura 33: Posição do pêndulo em relação ao tempo (duplo-PD)

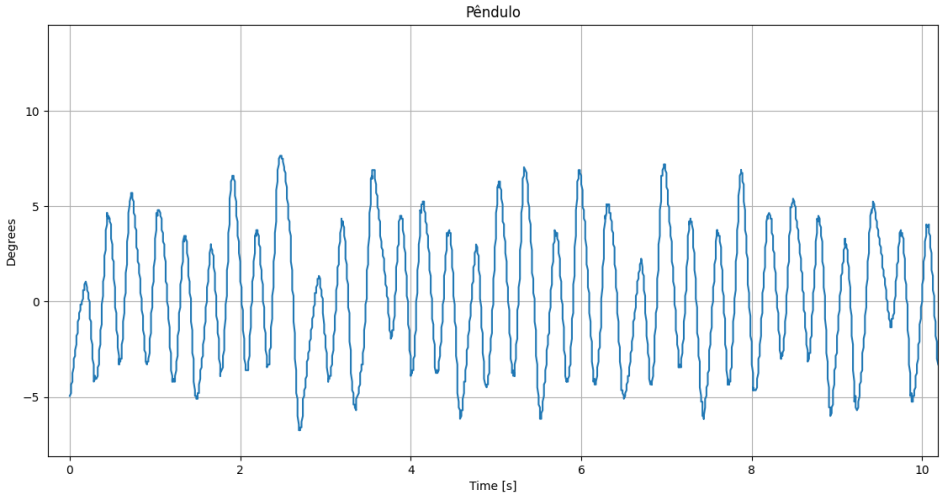


Figura 34: Posição do pêndulo em relação ao tempo (controlador híbrido)

Nota-se que quando o controlador duplo-PD é utilizado, a frequência de oscilação do pêndulo é menor, inclusive não oscilando em alguns momentos. Porém, também pode-se observar que a amplitude das oscilações é menor quando o controlador híbrido é utilizado.

Já em relação à comparação entre o gasto de energia pelos controladores, optou-se por calcular um índice baseado na integração do módulo da porcentagem de largura de pulso imposta ao motor DC, uma vez que sua corrente não foi medida.

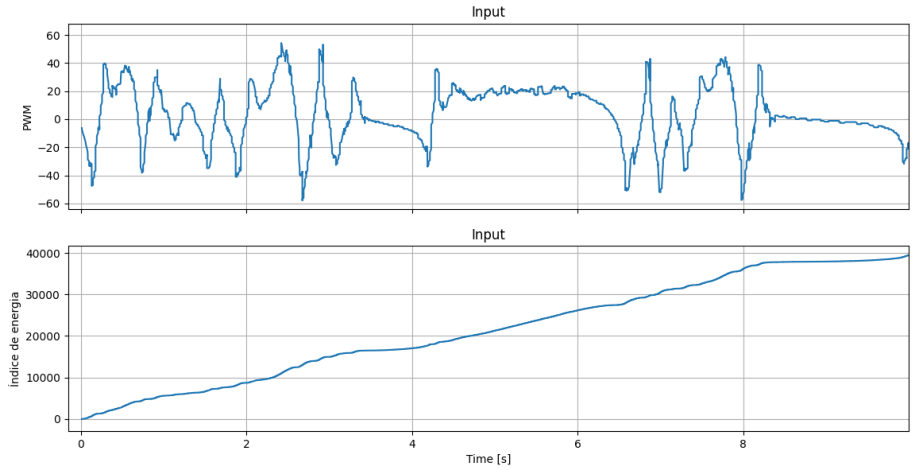


Figura 35: PWM e índice de energia (duplo-PD)

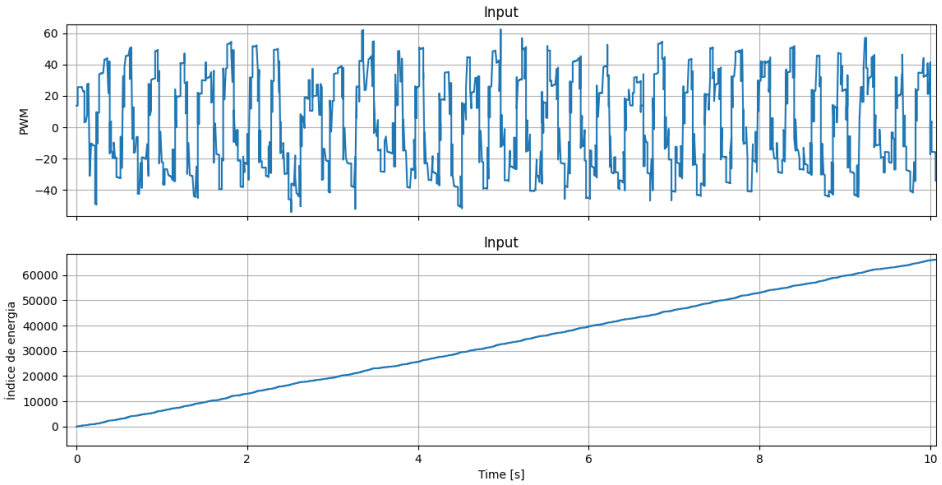


Figura 36: PWM e índice de energia (controlador híbrido)

As Figuras 35 e 36 apresentam em sua parte superior a porcentagem de largura de pulso imposta ao motor, de forma que os valores negativos indicam apenas que a tensão foi aplicada de maneira inversa. E em sua parte inferior, o índice calculado através da integração do módulo desta porcentagem.

Nota-se que após 10 segundos, o índice de energia calculado para o controlador duplo-PD é da ordem de 40.000 unidades, e para o controlador híbrido, de 65.000 unidades. Conclui-se que o segundo gasta mais energia do que o primeiro.

Além das comparações quantitativas realizadas anteriormente, destaca-se uma comparação qualitativa em relação à robustez dos dois métodos em relação a pequenas mudanças mecânicas da plataforma.

Após algum tempo de utilização do aparato, a flange que fixa o braço ao eixo do motor passa a apresentar uma pequena folga, que cresce com a utilização do equipamento. Diante deste fato, notou-se que, mesmo com uma folga relativamente grande, o método híbrido continua se comportando bem, ainda sendo capaz de manter o pêndulo equilibrado. O mesmo não ocorre com o controlador baseado no método clássico, que nestas condições passa a apresentar oscilações e até a incapacidade de equilibrar o pêndulo.

Por fim, destaca-se o comportamento oscilatório apresentado pelo pêndulo quando controlado pelo método híbrido, e a provável relação deste comportamento com a função de recompensa utilizada, descrita na seção 9.3.1. Tal conclusão pode ser inferida conforme apresentado na seção 9, em que na maior parte do tempo o pêndulo oscila na região em que as recompensas são positivas.

11 CONCLUSÃO

A fim de ser utilizado como plataforma didática, este projeto proporciona uma série de fatores que auxiliam na consolidação do ensino de técnicas de controle. Um exemplo prático seria a criação de uma experiência de laboratório.

Com o fornecimento de projetos mecânico, elétrico e de computação implementados e unidos uma estrutura robusta, é possível desenvolver e testar diversos métodos de controle, conforme citado na seção 1.1, além dos executados neste trabalho.

Componente	**Custo**
Motor	142,40 R$
Encoder	170,00 R$
Ponte H	37,30 R$
Acoplamento	30,90 R$
Arduino	30,90 R$
Jumpers	10,00 R$
Chapas e parafusos	52,00 R$
Rolamentos	23,00 R$
Impressão 3D	35,00 R$
Fonte de tensão	48,90 R$
Total	581,3 R$

Tabela 9: Custo dos componentes do projeto.

Em conjunto com os resultados apresentados na seção 9, as Tabelas 9 e 10 evidenciam como os requisitos de projeto definidos na Tabela 2 foram satisfeitos.

Dimensão	**Valor**
Largura	200 [mm]
Comprimento	200 [mm]
Altura	300 [mm]

Tabela 10: Dimensões do protótipo.

Uma vez que os requisitos propostos foram atingidos, torna-se viável replicar a estrutura para atender a demanda de um laboratório de controle. Consequentemente, é possível propor o desenvolvimento de algoritmos e suas análises de resultado.

Vale ressaltar que o processo de treinamento de algoritmos de inteligência artificial requer um tempo consideravelmente maior em relação aos de controle clássico. Dessa forma, é possível seccionar o projeto em etapas distintas.

Assim, seria proposto a realização da experiência com uma fase de implementação de algoritmos e uma de realização dos treinamentos e testes. Ainda seria possível o fornecimento de uma *Q-Table* ou uma rede neural pré-treinada, agilizando todo o procedimento.

Outro fator de sucesso no desenvolvimento da plataforma foi a implementação de comunicação através do Arduino. Com o procedimento via serial, não foi observada qualquer instabilidade na leitura dos encoders e no envio dos esforços de controle ao longo de todo o projeto.

Quanto ao projeto computacional, a implementação dos algoritmos demonstrou ser bem eficaz, inclusive no contexto do *Q-Learning*. Apesar do resultado deste algoritmo individualmente não haver sido satisfatório, foi possível notar a eficiente introdução de todos os algoritmos, com curto tempo de execução e de obtenção de resultados, conciliando com um uso adequado das bibliotecas inseridas. Além disso, o código foi bem seccionado em classes e métodos com tarefas bem definidas, tornando simples sua manutenção e refatoração.

Enfatiza-se que a instabilidade do pêndulo obtida por meio do *Q-Learning* se deu por conta dos hiperparâmetros e principalmente da função de recompensa. Responsável pelo comportamento esperado da inteligência artificial, a função de recompensa deve abranger todos os objetivos a serem atingidos. Por exemplo, notou-se que a energia gasta com o método de controle híbrido foi superior a do controle clássico, mas tal resultado era plausível, uma vez que não foi incluído na rotina uma lógica que otimizasse o uso de energia.

Um fator determinante para o sucesso obtido com o método híbrido apresentado na seção 8.5 foi o *delay* introduzido na avaliação das ações pela função de recompensa. Apenas após sua implementação foi possível equilibrar o pêndulo com a introdução da inteligência artificial, evidenciando como a inércia do sistema impediu o êxito do algoritmo.

Por fim, observou-se que o método híbrido demonstrou ser consideravelmente mais robusto do que o PD, uma vez que é menos suscetível a pequenas mudanças mecânicas. Assim, não se observou necessário reavaliar os ganhos de controle e corrigir as imperfeições mecânicas com a frequência realizada com o controle PD.

11.1 Trabalhos futuros

Uma vez que foi necessário dar prioridade para alguns tópicos em detrimento de outros, e que devido ao cronograma do trabalho não foi possível refazer determinadas estruturas ou testar diferentes algoritmos e variáveis, destaca-se aqui algumas sugestões de trabalhos futuros.

O algoritmo utilizado para a dinâmica de *SwingUp*, apesar de ter sido capaz de realizar sua tarefa, não se comportou bem quando restrições em relação a ângulos máximos de rotação do braço foram impostos. Por isso, sugere-se o desenvolvimento de algoritmos que suportem este tipo de restrição, ou até o treinamento de modelos de inteligência artificial para realizar tal tarefa. Outra solução ainda seria projetar a plataforma de maneira que permitisse a rotação indefinida do braço.

Em relação à mecânica do projeto, destaca-se que a peça desenvolvida para a conexão entre o braço e o eixo do motor passa a apresentar folgas após um certo período de utilização da plataforma. Assim, tornou-se necessário a desmontagem da estrutura do braço e o reaperto dos parafusos ocasionalmente. Desta forma sugere-se a utilização de uma flange comercial ou a fabricação de tal peça em metal.

Em relação ao motor utilizado, destaca-se a presença de folgas em sua redução, o que dificulta a tarefa de equilibrar o pêndulo. Por isso, sugere-se a utilização de um motor sem redução, ou uma redução com menos folga, apesar de ambas opções representarem um aumento no custo do projeto.

Ainda em relação ao motor, sugere-se que sua corrente seja medida para que todos os estados da planta sejam obtidos, sendo assim possível implementar o controlador LQR. Outra solução seria a substituição do motor DC por um motor de passo.

Em relação ao algoritmo *Q-Learning*, sugere-se uma maior exploração dos hiperparâmetros e de um entendimento mais aprofundado da influência de cada um no processo

de aprendizagem. Por fim, poderia ser realizado um estudo da possibilidade de atualizar a *Q-Table* de forma off-line, ou seja, após o fim de cada episódio.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Bruno Augusto Evangélico, Kennedy X. B. Peixoto, Alan Favoretto Rocha, Gabriel Pereira das Neves, and Mateus Mussi Brugnolli. Pêndulo invertido rotacional. `https://sites.usp.br/lca/projetos/pendulo-invertido-rotacional/`, 2017.

[2] NASA/Bill Ingalls. File:soyuz tma-9 launch.jpg. `https://commons.wikimedia.org/wiki/File:Soyuz_TMA-9_launch.jpg`, 2019.

[3] R. Sritharan, M. Sivapalanirajan, and M. WilljuiceIruthayarajan. Mathematical modeling and control of qnet rotary inverted pendulum in matlab and real time implementation in lab view using elvis. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1452–1457, 2018.

[4] Markus Hehn and Raffaello D'Andrea. A flying inverted pendulum. In *2011 IEEE International Conference on Robotics and Automation*, pages 763–770, 2011.

[5] Xinrong Zhang, Jie Ma, Lian Lin, and Lele Wang. Study on swing-up control of rotary inverted pendulum based on energy feedback. In *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, pages 994–998, 2018.

[6] Ju-Bong Kim, Hyun-Kyo Lim, Chan-Myung Kim, Min-Suk Kim, Yong-Geun Hong, and Youn-Hee Han. Imitation reinforcement learning-based remote rotary inverted pendulum control in openflow network. *IEEE Access*, 7:36682–36690, 2019.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[8] Ioannis Kafetzis and Lazaros Moysis. Inverted pendulum: A system with innumerable applications. 03 2017.

[9] T-P Azevedo Perdicoúlis and P. Lopes Dos Santos. The secrets of segway revealed to students: Revisiting the inverted pendulum. In *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*, pages 43–48, 2018.

[10] Soukaina Krafes, Zakaria Chalh, and Abdelmjid Saka. Review: Linear, nonlinear and intelligent controllers for the inverted pendulum problem. In *2016 International Conference on Electrical and Information Technologies (ICEIT)*, pages 136–141, 2016.

[11] Joseph Peltroche and Anibal E. Morales Zambrana. Advanced mathematics for control system design: Guidance, navigation, and control (gnc) studies. `https://ntrs.nasa.gov/api/citations/20190002679/downloads/20190002679.pdf`, 2019.

[12] N.D. Anh, H. Matsuhisa, L.D. Viet, and M. Yasuda. Vibration control of an inverted pendulum type structure by passive mass–spring-pendulum dynamic vibration absorber. *Journal of Sound and Vibration*, 307(1):187–201, 2007.

[13] Aiman Omer, Kenji Hashimoto, Hun-ok Lim, and Atsuo Takanishi. Study of bipedal robot walking motion in low gravity: Investigation and analysis. *International Journal of Advanced Robotic Systems*, 11:1, 09 2014.

[14] Masahiro Wada, Masahiro Tanaka, Tomohiro Umetani, and Minoru Ito. Framework of control and stabilization system for an inverted pendulum moving robot. *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, 2011, 05 2011.

[15] Kashika Chhabra and Mohd Rihan. Design of linear quadratic regulator for rotary inverted pendulum using labview. pages 1–5, 04 2016.

[16] Binita Prakash, Binoy Krishna Roy, and Raj Kumar Biswas. Design, implementation and comparison of different controllers for a rotary inverted pendulum. In *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, pages 1–6, 2016.

[17] Abhishek Kathpal and Ashish Singla. Simmechanics™ based modeling, simulation and real-time control of rotary inverted pendulum. In *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, pages 166–172, 2017.

[18] Ju-Bong Kim, Do-Hyung Kwon, Yong-Geun Hong, Hyun-Kyo Lim, Min-Suk Kim, and Youn-Hee Han. Deep q-network based rotary inverted pendulum system and its monitoring on the edgex platform. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 034–039, 2019.

[19] Inc. Quanser. In *Qnet Rotpen - User Manual*, pages 1–26, 2011.

[20] Dominic Brown and Martin Strube. Design of a neural controller using reinforcement learning to control a rotational inverted pendulum. In *2020 21st International Conference on Research and Education in Mechatronics (REM)*, pages 1–5, 2020.

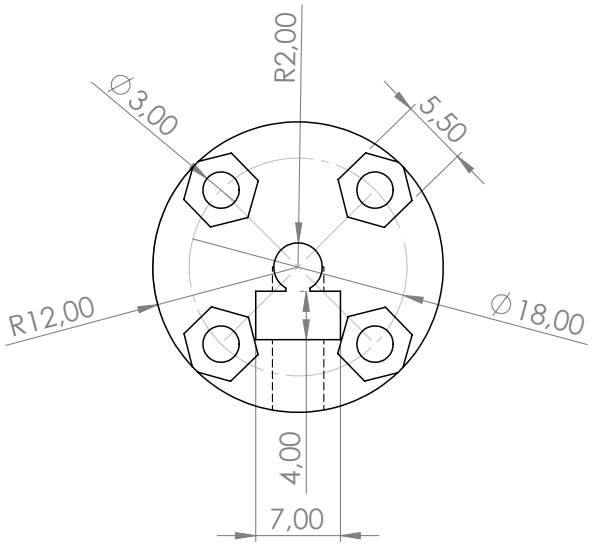
[21] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

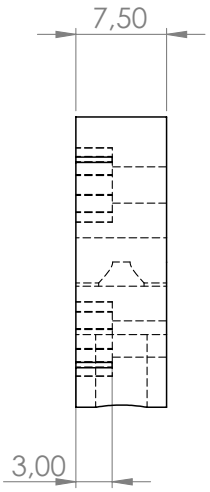
[22] Peter Foy. Deep reinforcement learning: Guide to deep q-learning. `https://www.mlq.ai/deep-reinforcement-learning-q-learning/`, 2021.

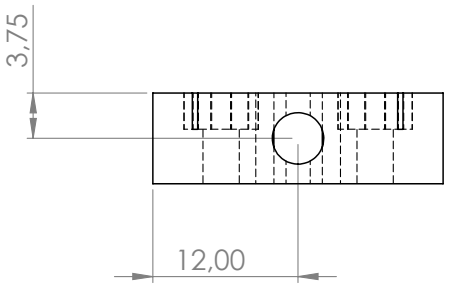
[23] Zoltán Lőrincz. A brief overview of imitation learning. `https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c`, 2019.

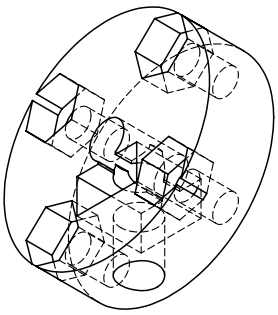
[24] Quanser Inc. Rotary pendulum workbook. `https://www.quanser.com/products/rotary-inverted-pendulum/`, 2011.

APÊNDICE A

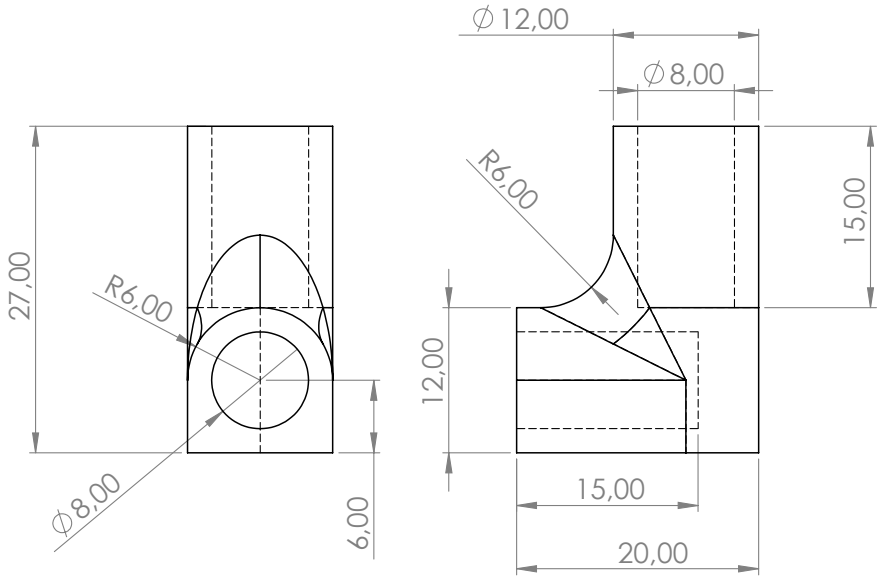


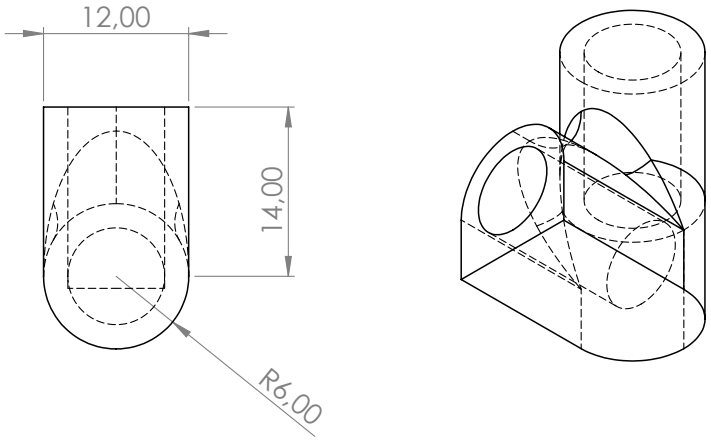




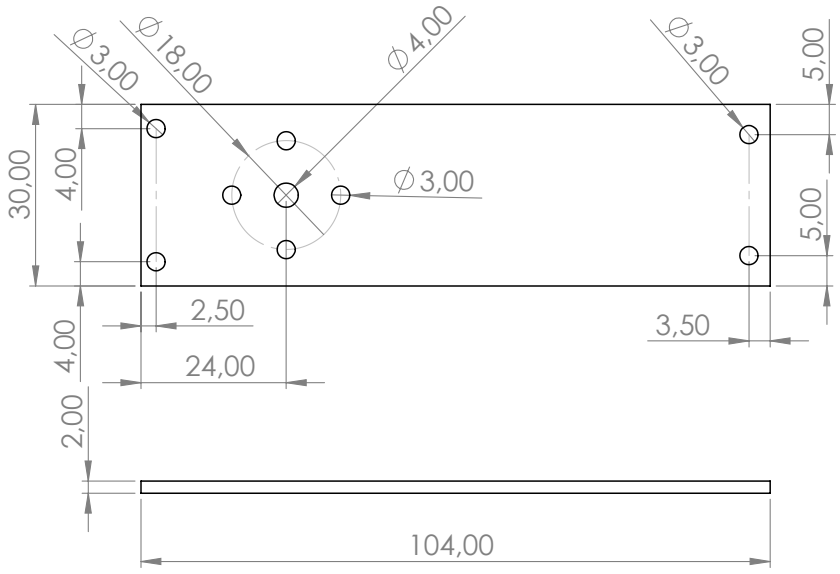


	NOME	ASSINATURA	DATA				TÍTULO: Flange	
DESEN.								
VERIF.								
APROV.								
MANUF.								
QUALID				MATERIAL: PLA (Impressão 3D)			DES. Nº	A4
				PESO:			ESCALA:2:1	FOLHA 1 DE 1

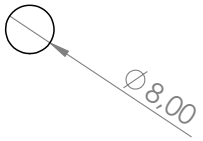


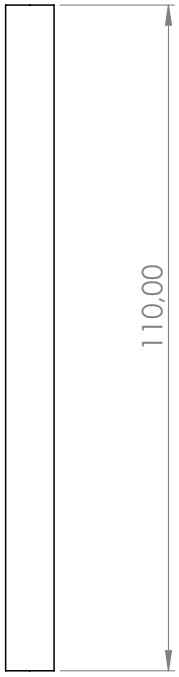


		ACABAMENTO:				REVISÃO	
	NOME	ASSINATURA	DATA			TÍTULO:	
DESEN.						Acoplamento Pêndulo - Eixo	
VERIF.							
APROV.							
MANUF.							
QUALID				MATERIAL: PLA (Impressão 3D)		DES. Nº	A4
				PESO:		ESCALA:2:1	FOLHA 1 DE 1

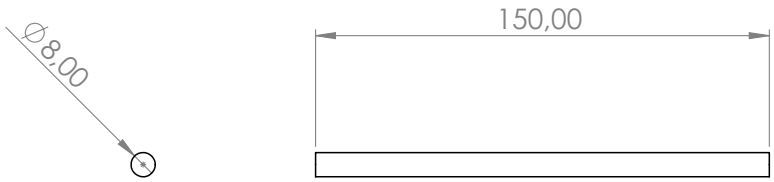


SE NÃO ESPECIFICADO: DIMENSÕES EM MILÍMETROS ACABAM. SUPERFÍCIE: TOLERÂNCIAS: LINEAR: ANGULAR:	ACABAMENTO:						
	NOME	ASSINATURA	DATA			TÍTULO:	
DESEN.						Braço	
VERIF.							
APROV.							
MANUF.							
QUALID.				MATERIAL:		DES. Nº	A4
				MDF			
				PESO:		ESCALA:1:1	FOLHA 1 DE 1

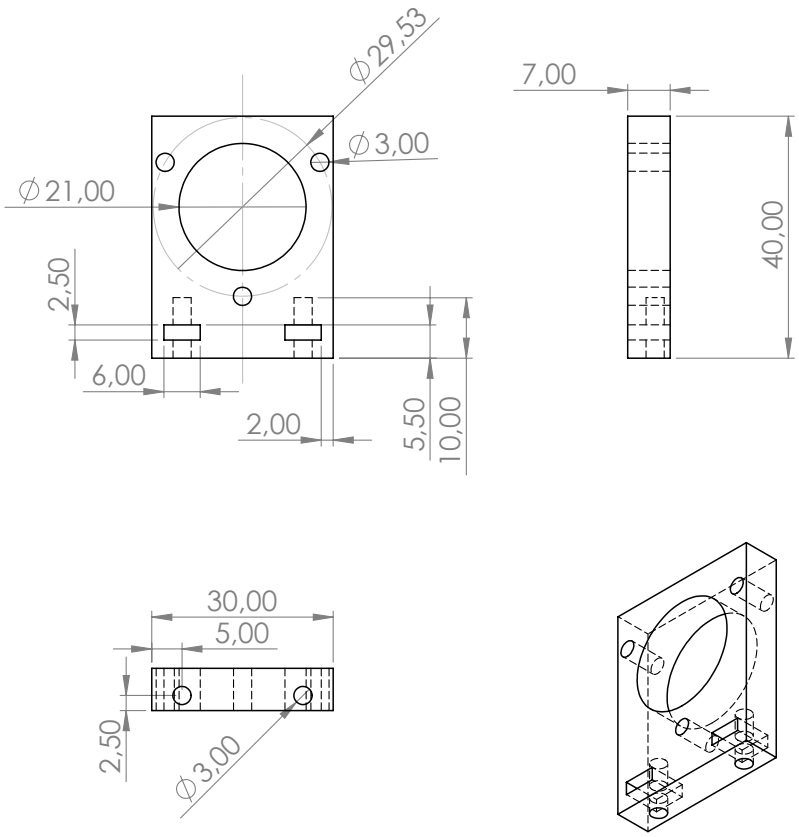




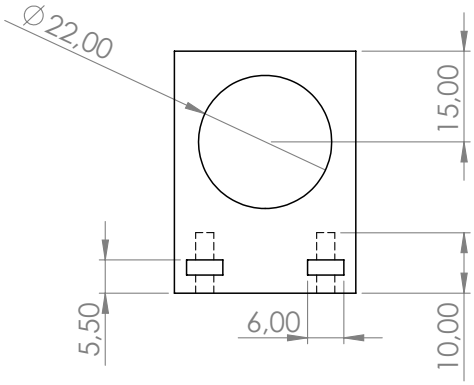
		ACABAMENTO:				REVISÃO	
	NOME	ASSINATURA	DATA			TÍTULO:	
DESEN.						Eixo do encoder	
VERIF.							
APROV.							
MANUF.							
QUALID.				MATERIAL:		DES. Nº	A4
				Aço			
				PESO:		ESCALA:1:1	FOLHA 1 DE 1

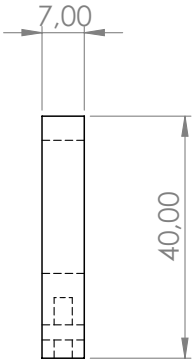


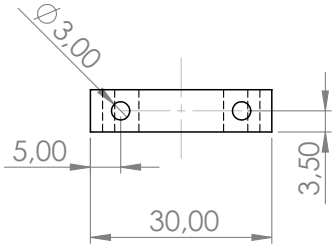
		ACABAMENTO:				REVISÃO	
	NOME	ASSINATURA	DATA			TÍTULO:	
DESEN.						Pêndulo	
VERIF.							
APROV.							
MANUF.							
QUALID.				MATERIAL:		DES. Nº	A4
				Aço			
				PESO:		ESCALA:1:2	FOLHA 1 DE 1

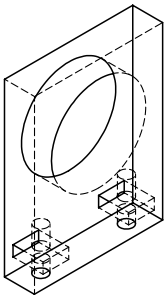


		ACABAMENTO:				REVISÃO	
	NOME	ASSINATURA	DATA			TÍTULO: Suporte Encoder	
DESEN.							
VERIF.							
APROV.							
MANUF.							
QUALID.				MATERIAL: PLA (Impressão 3D)		DES. Nº	A4
				PESO:		ESCALA:1:1	FOLHA 1 DE 1









		ACABAMENTO:				REVISÃO	
	NOME	ASSINATURA	DATA			TÍTULO:	
DESEN.						Suporte Rolamento	
VERIF.							
APROV.							
MANUF.							
QUALID.				MATERIAL: PLA (Impressão 3D)		DES. Nº	A4
				PESO:		ESCALA:1:1	FOLHA 1 DE 1