UNIVERSIDADE FEDERAL DO NORTE DO TOCANTINS
CAMPUS UNIVERSITÁRIO DE ARAGUAÍNA
MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA

Rafael Medeiros de Freitas

EXPLORANDO A DINÂMICA NÃO LINEAR EM SISTEMAS OSCILANTES: UM GUIA DIDÁTICO PARA O ENSINO DE FÍSICA

PRODUTO EDUCACIONAL

Araguaína - TO
2025

Rafael Medeiros de Freitas

EXPLORANDO A DINÂMICA NÃO LINEAR EM SISTEMAS OSCILANTES: UM GUIA DIDÁTICO PARA O ENSINO DE FÍSICA

Produto de Mestrado apresentado ao Programa de Pós-Graduação em Ensino de Física, no Curso de Mestrado Profissional de Ensino de Física (MNPEF), como parte dos requisitos necessários à obtenção do título de Mestre em Ensino de Física.

Orientador(a): Profa. Dra. Liliana Yolanda Ancalla Davila
Coorientador: Dr. Nilo Maurício Sotomayor Choque

Araguaína - TO
2025

LISTA DE ILUSTRAÇÃO

LISTA DE TABELAS

SUMÁRIO

1 APRESENTAÇÃO

Os desafios do ensino de Física no Ensino Médio permanecem como um dos principais obstáculos para o desenvolvimento pleno dos estudantes. Grande parte dessas dificuldades decorre do distanciamento entre a teoria e a prática, comprometendo a motivação e o engajamento dos alunos. Pensando nisso, este produto educacional foi desenvolvido com o objetivo de integrar a experimentação, a simulação computacional e o uso de recursos digitais, especialmente no campo da Mecânica, aproximando essas dimensões que, juntas, enriquecem o processo de aprendizagem.

O produto educacional inclui em um sistema experimental formado por dois pêndulos acoplados por uma mola, com sensores de posição (encoders) conectados a placas Arduíno, permitindo a coleta precisa dos dados de deslocamento angular. Para viabilizar o funcionamento do aparato, foram desenvolvidos códigos específicos para o Arduíno, responsáveis pela leitura dos dados dos encoders, além de códigos implementados no LabVIEW para aquisição, processamento e visualização gráfica dos dados em tempo real. A construção física do sistema conta ainda com desenhos em AutoCAD, fornecendo detalhes técnicos para garantir precisão e qualidade no processo de montagem.

Além do aparato experimental, o produto contempla um manual detalhado que orienta passo a passo a construção e a calibração do sistema, além de apresentar uma descrição formal e abrangente do seu funcionamento. Esse manual também integra simulações computacionais desenvolvidas com foco tanto nos aspectos numéricos quanto na visualização gráfica, utilizando códigos em Python que permitem a reprodução, análise e interpretação dos fenômenos simulados.

O material é complementado por um e-book, que organiza e sistematiza todos os elementos do produto educacional, servindo como um guia teórico, prático e metodológico. Juntamente com o e-book, foi elaborada uma Sequência de Ensino Investigativo (SEI), estruturada para conduzir o uso do sistema experimental e das simulações em sala de aula. Essa sequência foi planejada com o objetivo de promover o aprendizado ativo por meio da exploração, formulação de hipóteses e investigação científica, incentivando a participação ativa dos estudantes. Para potencializar a aplicação prática do produto, também foram desenvolvidas aulas teóricas e práticas integradas, questionário diagnóstico e atividades complementares que orientam o aprofundamento dos conceitos abordados.

Desenvolvido com propósitos didáticos, o produto educacional promove a integração entre a sala de aula e as atividades práticas em laboratórios de ensino de Física, servindo como uma ferramenta para motivar os estudantes e dar suporte às aulas teóricas. Sua aplicação ocorre, especificamente, em aulas de Mecânica no Ensino Médio, incentivando a prática experimental e o uso de tecnologias computacionais como complementos essenciais ao aprendizado teórico. A proposta busca, portanto, aproximar a teoria e a prática no ensino de Física, reforçando os conceitos estudados e desenvolvendo habilidades investigativas nos alunos.

Dessa forma, o produto educacional integra o sistema experimental, o manual detalhado, as simulações computacionais, os códigos em Arduíno, LabVIEW e Python, os desenhos em AutoCAD, as aulas teóricas e práticas, a sequência investigativa e o questionário de avaliação diagnóstica, compondo um conjunto completo e articulado de

materiais que enriquecem o ensino e a aprendizagem da Mecânica no Ensino Médio.

Rafael Medeiros de Freitas

2 SISTEMA EXPERIMENTAL - DOIS PÊNDULOS ACOPLADOS POR UMA MOLA

A concepção deste trabalho baseia-se em uma pesquisa bibliográfica robusta, que consolida o uso de dispositivos tecnológicos acessíveis, como o Arduíno, em demonstrações e investigações de fenômenos físicos. A relevância do aparato desenvolvido nesta pesquisa está alinhada a diversas iniciativas anteriores, destacando-se, contudo, por sua proposta inovadora. Esse diferencial agrega valor tanto à Ciência quanto ao avanço tecnológico, além de fortalecer a inclusão de ferramentas e recursos didáticos acessíveis em salas de aula de diferentes contextos.

Além da bibliografia considerada, fez-se necessário um vasto estudo sobre a plataforma programável, sobre o código necessário para o funcionamento, a montagem do circuito e os elementos constituintes. Paralelamente a isso, a abordagem exigiu o desenvolvimento de saberes relacionados a desenhos técnicos estruturais em AutoCAD, adaptação ao ambiente de desenvolvimento e linguagem de programação gráfica LabVIEW para exibição em tempo real dos gráficos, além do desenvolvimento de código computacional em Python para integração numérica das equações do movimento pendular e para a visualização gráfica.

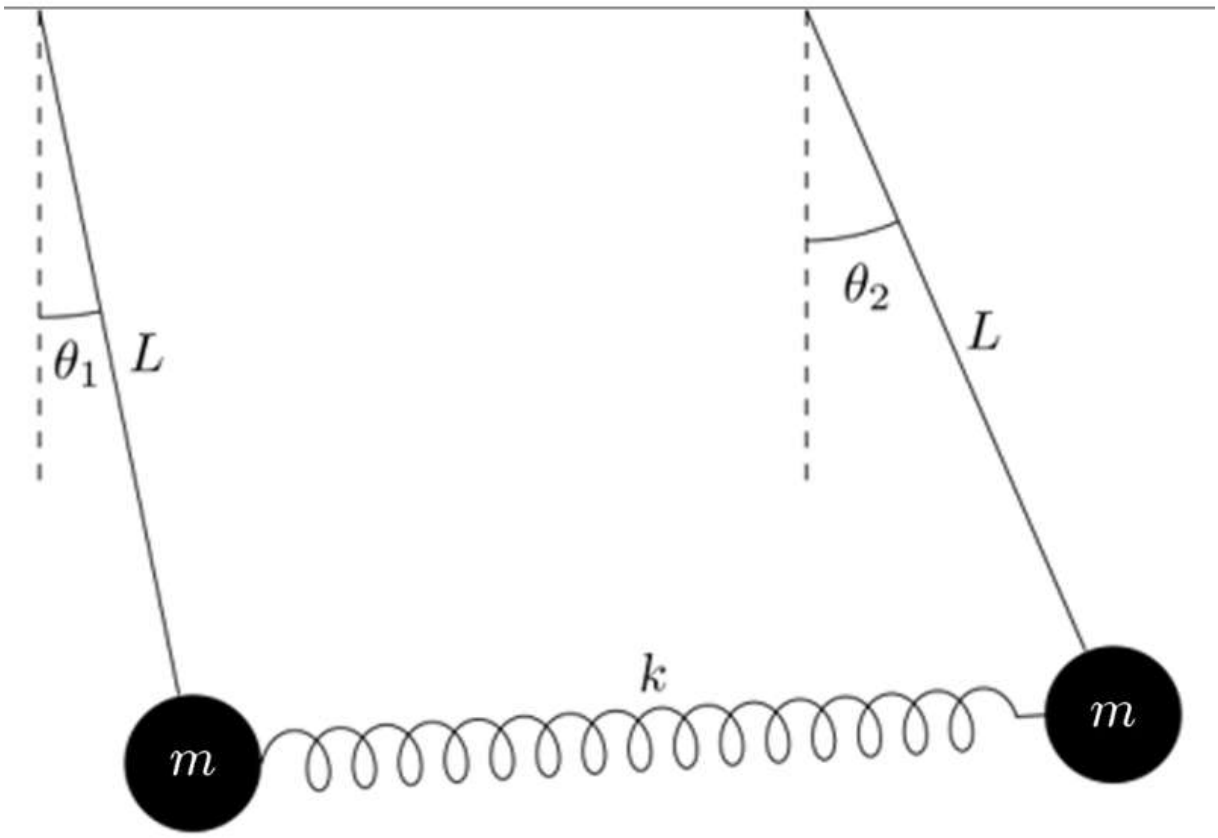
A construção do sistema dos dois pêndulos acoplados por uma mola foi desenvolvida quase que integralmente no Laboratório de Materiais para Aplicações em Dispositivos Eletrônicos LABMADE da Universidade Federal do Norte do Tocantins. A unidade conta com uma infraestrutura física de aproximadamente 120 m² dividida em 6 salas. Conta também com diversas facilidades de pesquisa, entre elas microcomputadores, ferramentas, máquinas, softwares de desenho técnico, softwares para instrumentação, bancadas, entre outras. Também conta com tecnologia, desenvolvida através de vários anos, em instrumentação, automação, controle de sistemas e aquisição analógico-digital de dados de grandezas físicas.

Em relação ao sistema físico em estudo, o LABMADE tem orientado trabalhos sobre sistemas dinâmicos não lineares. Além da descrição física de sistemas mecânicos, foi enfatizado também o estudo do formalismo matemático, principalmente em relação a espaços topológicos e variedades.

A motivação para o presente trabalho parte do fato de que boa parte dos objetos materiais, em especial os sólidos, podem ser considerados como um conjunto muito grande de osciladores harmônicos simples acoplados entre eles (átomos e moléculas em materiais sólidos). É de interesse para os físicos, do ponto de vista teórico, experimental e didático, compreender como esse efeito de acoplamento pode afetar o comportamento de cada um dos osciladores individuais.

Um sistema simples que pode ajudar a entender esse comportamento complexo pode ser construído a partir de dois pêndulos idênticos de comprimento L interconectados por uma mola com alguma constante elástica k, como mostra a figura 2.1. Pode-se estudar o movimento de cada um usando a formulação de Lagrange da mecânica clássica (THORNTON; MARION, 2019).

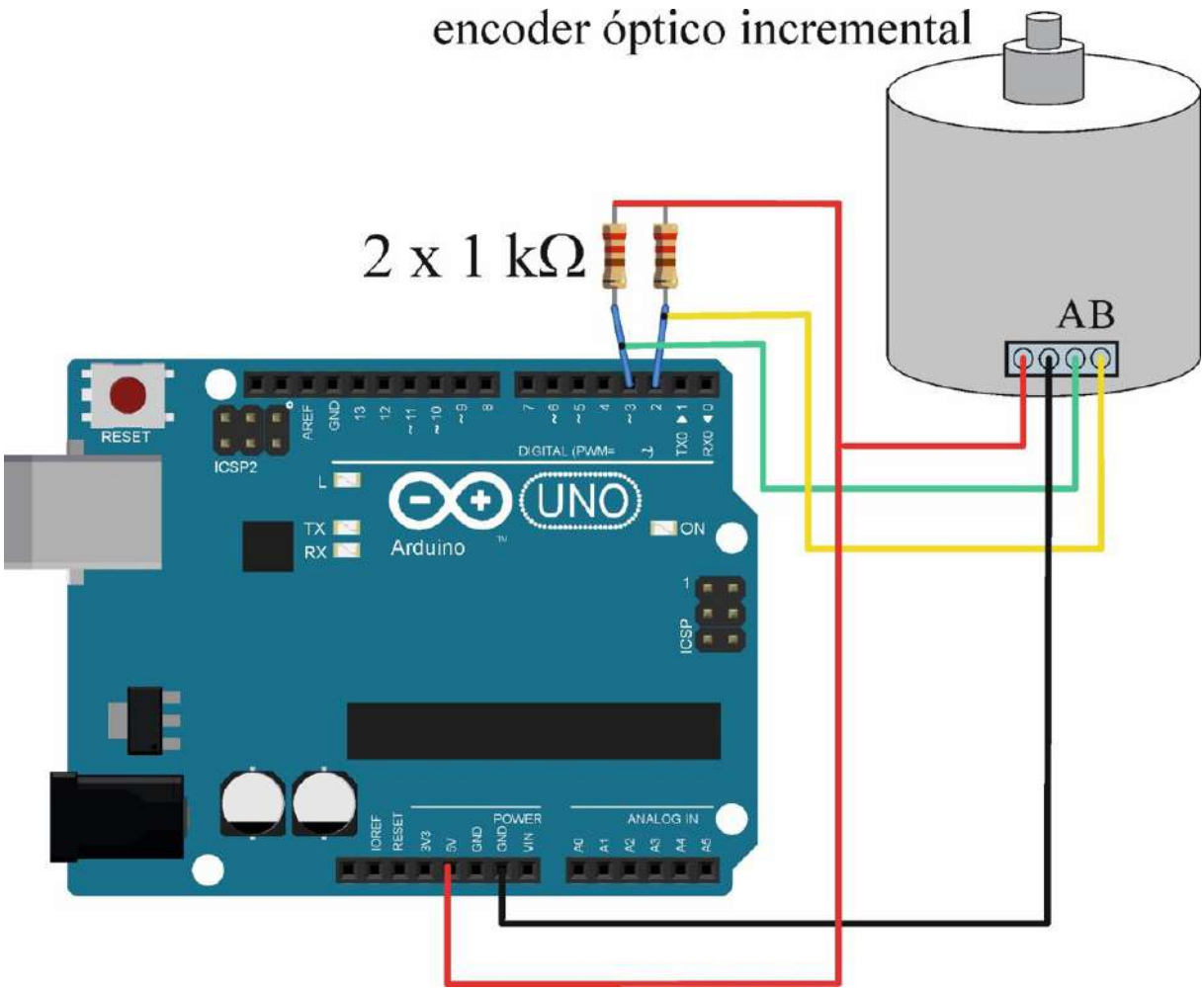
Figura 2.1 – Esquematização de dois pêndulos acoplados por uma mola



Fonte: N. M. Sotomayor

Dessa forma, o aparato físico constitui-se basicamente dos elementos citados acima (dois pêndulos acoplados por uma mola de pequena constante elástica). Uma estrutura de madeira foi desenvolvida a fim de fixar os pêndulos minimizando alterações verticais. Tal estrutura conta também com um trilho que permite variar as posições horizontais de cada pêndulo. A captação das oscilações mecânicas é realizada por meio de encoders ópticos e posteriormente transformadas em sinais digitais com o intermédio da plataforma programável Arduíno, como é possível visualizar na figura 2.2, a qual apresenta a configuração genérica da conexão entre um encoder e uma placa Arduíno UNO.

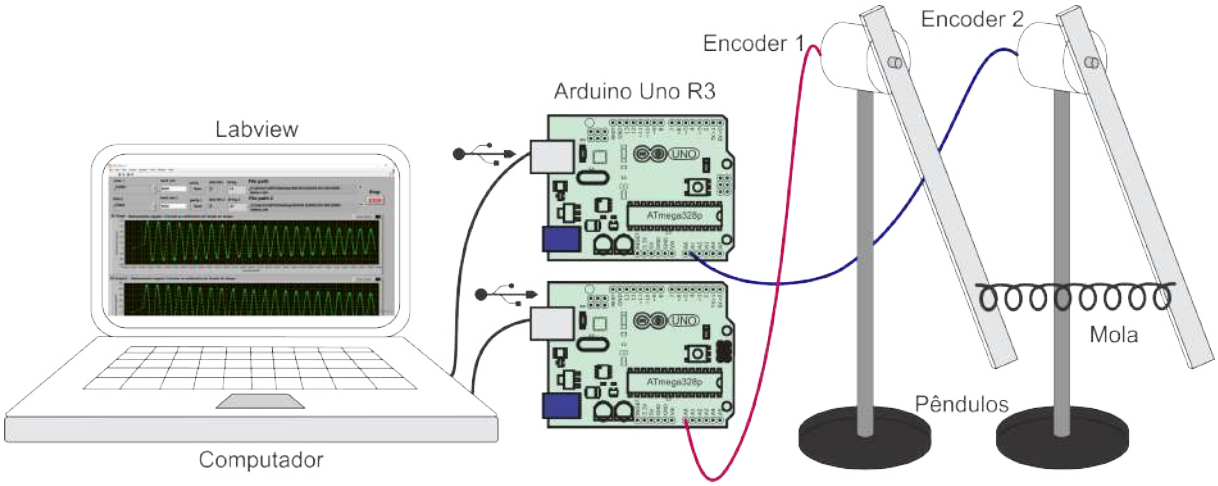
Figura 2.2 – Conexão encoder - Arduíno



Fonte: N. M. Sotomayor

A figura 2.3 mostra a esquematização do sistema experimental utilizado para medir o deslocamento angular de dois pêndulos acoplados por uma mola, com aquisição automática de dados. Na imagem, observa-se os dois pêndulos conectados por uma mola, representando um sistema de osciladores acoplados, os encoders ópticos instalados no topo de cada pêndulo para medir o deslocamento angular, dois módulos Arduíno Uno R3, responsáveis pela leitura dos dados dos encoders e envio das informações para o computador e o computador com software LabVIEW para o processamento e exibição dos dados recebidos dos Arduínos em gráficos de deslocamento angular em função do tempo.

Figura 2.3 – Sistema de aquisição de dados de dois pêndulos acoplados com mola, utilizando encoders, Arduínos e o software LabVIEW para análise do movimento.



Fonte: N. M. Sotomayor

2.1 Componentes e materiais utilizados

Para a construção do sistema experimental proposto, foi necessário selecionar cuidadosamente os componentes e materiais que possibilitam sua montagem, funcionamento e coleta de dados de forma eficiente e precisa. Cada item foi escolhido com base em sua relevância e precisão para o estudo das propriedades físicas envolvidas, bem como sua compatibilidade com o sistema de aquisição automática de dados. Abaixo, apresenta-se a lista de itens utilizados no experimento, seguida por uma descrição detalhada de cada componente, destacando suas especificações técnicas e papel no experimento, além de justificativas para sua inclusão no sistema.

Lista de materiais utilizados:

- Portal de madeira;
- Guia linear;
- Blocos deslizantes;
- Encoders óptico;
- Suporte de poliamida;
- Bloco de madeira;
- Hastes de acrílico;
- Flanges;
- Parafusos;
- Mola;

- Placas programáveis Arduíno UNO;
- Cabos USB tipo B.

2.1.1 **Portal de madeira**

Figura 2.4 – Portal de madeira

Fonte: Autoria própria

A estrutura foi construída em madeira do tipo cedro, visando proporcionar uma base estável e confiável para a fixação dos pêndulos utilizados no experimento. O arranjo inclui uma base retangular com 1 metro de comprimento, duas colunas verticais de 90 centímetros de altura cada, e uma viga superior, também com 90 centímetros de comprimento, onde são instalados os encoders. A configuração foi projetada com o propósito de maximizar a estabilidade do conjunto, minimizando quaisquer movimentos indesejados que poderiam interferir na precisão das medições e na observação dos fenômenos físicos investigados. A estrutura apresenta uma massa aproximada de 15 kg, combinando robustez e funcionalidade. A escolha da madeira como material estrutural alia baixo custo financeiro, durabilidade e

facilidade de manipulação, contribuindo significativamente para a eficiência do aparato experimental.

O portal foi confeccionado em uma marcenaria localizada na cidade de Imperatriz, Maranhão. A construção foi realizada com base em especificações detalhadas fornecidas ao marceneiro, garantindo que o design atendesse aos requisitos técnicos do projeto.

2.1.2 **Guia Linear e blocos deslizantes**

Figura 2.5 – Trilho guia de eixo duplo: Projetada para movimento linear suave e preciso em máquinas e sistemas de automação, com trilhos paralelos e elementos rolantes.

Fonte: Autoria própria

Figura 2.6 – Detalhe dos elementos rolantes

Fonte: Strodox Tools Store

Figura 2.7 – Guia linear em perspectiva

Fonte: Strodox Tools Store

Na viga superior do portal foi instalada uma guia linear (figura 2.5), que se estende de uma extremidade à outra. Esse componente, semelhante a um trilho, tem a função de permitir o deslocamento horizontal dos pêndulos ao longo da estrutura de madeira. Essa característica possibilita a alteração das posições relativas entre os pêndulos, permitindo a observação de diferentes padrões oscilatórios quando essa condição é modificada.

A guia linear, com comprimento original de 1 metro, foi adquirida por meio do e-commerce *Aliexpress*. Para ajustar suas dimensões às proporções da viga superior, foi

necessário reduzir seu tamanho em 10 centímetros. Além da guia, foram adquiridos dois blocos deslizantes (figura 2.6) com sistema de bloqueio, que permitem tanto o movimento horizontal quanto a fixação precisa das posições desejadas. Os encoders estão fixados nesses blocos deslizantes, o que possibilita sua movimentação controlada e sua estabilização em pontos específicos, garantindo a flexibilidade e a precisão do experimento. A figura 2.8 fornece uma visão detalhada dos blocos deslizantes adquiridos.

Figura 2.8 – Bloco deslizante com sistema de travamento integrado.

Fonte: Strodox Tools Store

2.1.3 **Encoder óptico**

Um encoder óptico incremental é um dispositivo eletrônico utilizado para converter movimentos mecânicos em sinais digitais, permitindo a medição de variáveis como distâncias, velocidades, número de rotações, ângulos, entre outros. O funcionamento desse dispositivo baseia-se em um disco com marcações (pulsos), acompanhado por um emissor e um receptor de luz. À medida que o disco gira, os pulsos gerados são detectados pelo receptor, que os converte em sinais digitais passíveis de análise. Nas figuras 2.9, 2.10 e 2.11, é possível observar o conjunto de encoders empregados no aparato experimental.

Figura 2.9 – Encoders óptico rotativos com conectores e cabos destacados.

Fonte: Autoria própria

Figura 2.10 – Encoders óptico - especificações de conexão.

Fonte: Autoria própria

Figura 2.11 – Vista lateral dos encoders ópticos incrementais.

Fonte: Autoria própria

As especificações do modelo utilizado neste experimento são as seguintes:

- **Resolução:** 1000 p/r (100 pulsos por revolução em fase única);
- **Fonte de alimentação:** DC 5-24V;
- **Diâmetro do eixo:** 6 mm;
- **Comprimento do eixo:** 13 mm;
- **Dimensões:** 38 mm de diâmetro e 40 mm de altura (sem considerar o eixo);

- **Saída:** sinal de pulso ortogonal em duas fases (AB), com saída do tipo coletor aberto NPN;
- **Velocidade mecânica máxima:** 5000 rotações por minuto (R/min);
- **Frequência de resposta:** 0-20 kHz;
- **Comprimento do cabo:** 1,5 metros.

Neste experimento, foi necessário utilizar dois encoders ópticos incrementais, um para cada pêndulo, de forma a possibilitar a coleta de dados independentes para cada sistema oscilatório. Ambos os dispositivos foram adquiridos por e-commerce *Casa da Robótica* (www.casadarobotica.com), localizado no estado da Bahia. Essa escolha levou em consideração não apenas as especificações técnicas que atendem aos requisitos do experimento, mas também a confiabilidade e a acessibilidade do fornecedor.

2.1.4 **Suporte em poliamida**

Figura 2.12 – Suporte semicircular em Nylon para fixação precisa dos encoders.

Fonte: Autoria própria

A fixação dos encoders ópticos ocorreu através de suportes semicirculares fabricados em poliamida, comercialmente conhecida como Nylon. A estrutura é composta por duas partes ajustáveis, fixadas entre si com parafusos metálicos, permitindo que o suporte seja preso firmemente em torno dos encoders, como pode ser visualizado na figura 2.12. O suporte apresenta um diâmetro de aproximadamente 10 centímetros e uma altura de 5 centímetros.

O material utilizado na fabricação pode ser o Nylon 6 (Poli(ϵ-caprolactama)) ou o Nylon 6,6 (Poli(hexametileno adipamida)), ambos amplamente empregados na manufatura subtrativa de peças mecânicas devido à sua excelente resistência mecânica, durabilidade e estabilidade dimensional.

2.1.5 **Bloco de madeira**

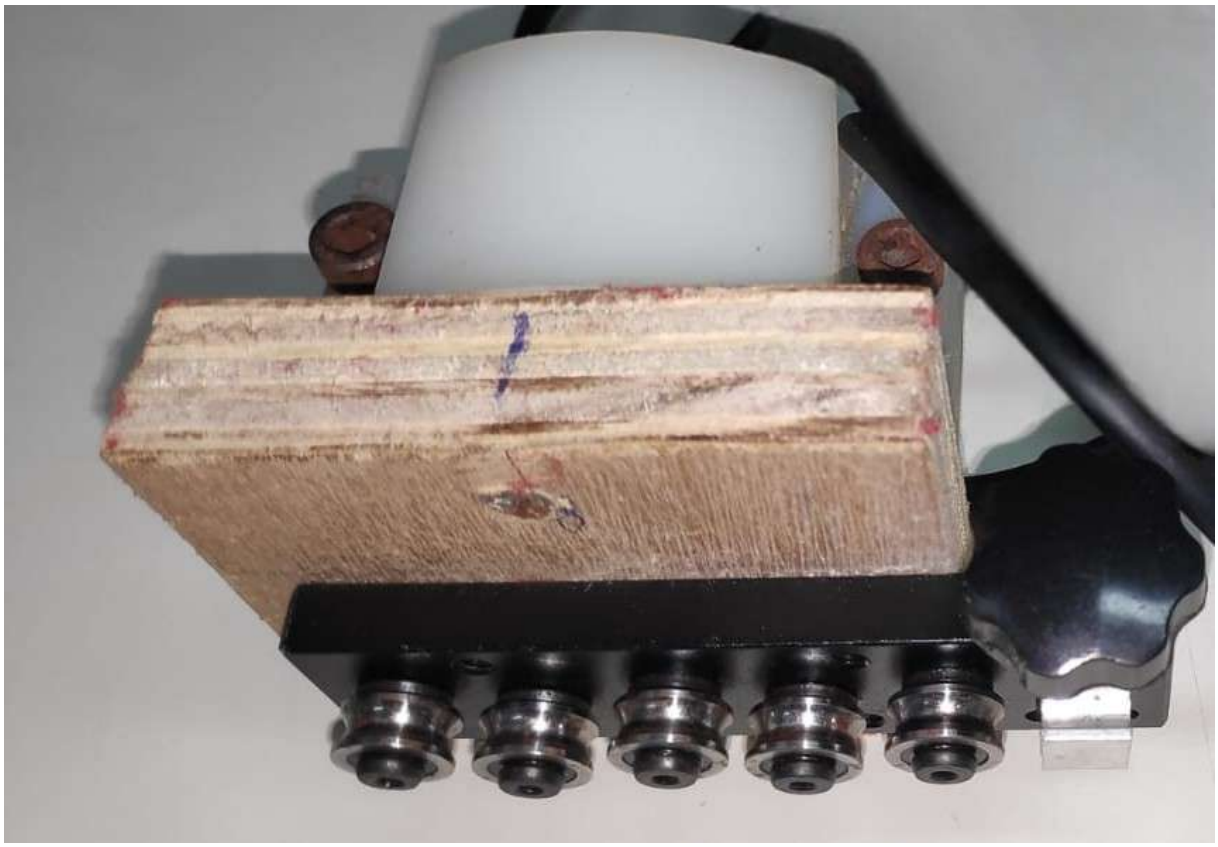
Figura 2.13 – Bloco de madeira utilizado como elemento de acoplamento entre suporte de poliamida e blocos deslizantes em AutoCAD.

Fonte: Autoria própria

Para fixar os encoders ópticos aos blocos deslizantes, além dos suportes de poliamida, também foi necessária a utilização de blocos de madeira fabricados a partir de madeira compensada (ou MDF). Estes blocos possuem formatos quadrados, tendo 10 centímetros de lado e espessura de aproximadamente 1,5 centímetros. A figura 2.13 mostra o desenho em AutoCAD dessa peça.

Como é possível ver na figura 2.14, o acoplamento do bloco de madeira ao suporte de poliamida é mostrado na parte frontal, enquanto a conexão com o bloco deslizante está localizada na parte posterior. Ambas as peças foram fixadas utilizando parafusos metálicos, assegurando a rigidez estrutural e funcionalidade do conjunto.

Figura 2.14 – Bloco de madeira fixado ao bloco deslizante e ao suporte de poliamida através de parafusos metálicos



Fonte: Autoria própria

2.1.6 **Hastes de acrílico**

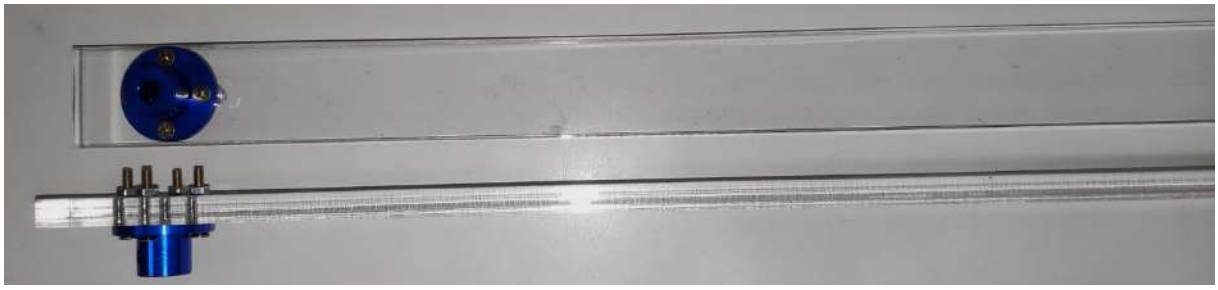
Figura 2.15 – Hastes de acrílico transparente.

Fonte: Autoria própria

A constituição dos pêndulos é composta por duas barras de acrílico, material plástico derivado de polímeros do ácido acrílico ou metacrílico, cada uma com 80 centímetros de comprimento. Essas barras são acopladas diretamente ao eixo dos encoders, permitindo a captação precisa do movimento mecânico e sua conversão em dados digitais. As hastes,

cujo desenho CAD foi realizado em Corel Draw e convertido em arquivo .obj, permitindo a leitura da cortadora, foram projetadas e fabricadas em acrílico com 8 milímetros de espessura, utilizando o corte a laser para garantir maior precisão dimensional. A estrutura das barras inclui quatro furos na parte superior, destinados à fixação dos flanges que as conectam aos encoders, e três furos na parte inferior, posicionados com espaçamento de 5 centímetros entre si, permitindo ajustes na posição da mola que as interliga. A figura 2.15 mostra as hastes de acrílico confeccionadas dispostas paralelamente sobre uma superfície. Na imagem, é possível ver os furos distribuídos ao longo de sua extensão inferior utilizados para o acoplamento da mola entre elas, bem como os da parte superior onde são acoplados os flanges na montagem, como é possível visualizar na imagem abaixo (figura 2.16).

Figura 2.16 – Hastes de acrílico com flanges metálicos.



Fonte: Autoria própria

2.1.7 **Flanges e parafusos**

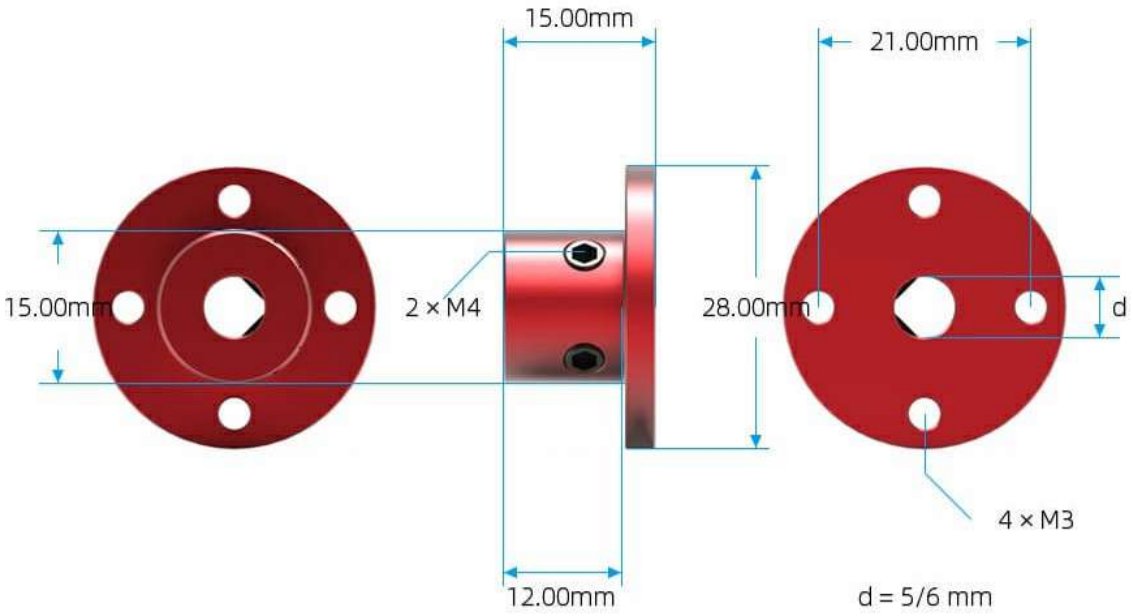
Figura 2.17 – Flanges metálicos em perspectiva frontal e lateral, destacando os furos para fixação.

Fonte: Autoria própria

Para fixar as hastes aos encoders, foram utilizados flanges de alumínio, pequenas peças com orifícios centrais projetados para acoplamento a eixos como os presentes nos encoders. A fixação ocorre com parafusos perpendiculares ao corpo da peça, proporcionando estabilidade durante o funcionamento e maior precisão nas medidas. A figura 2.17 mostra as peças utilizadas no aparato.

Os flanges utilizados também foram adquiridos de forma online, através do e-commerce *Aliexpress*. A figura abaixo (figura 2.18) mostra as dimensões dos flanges utilizados.

Figura 2.18 – Dimensões dos flanges.



Fonte: GKTOOLS Official Store

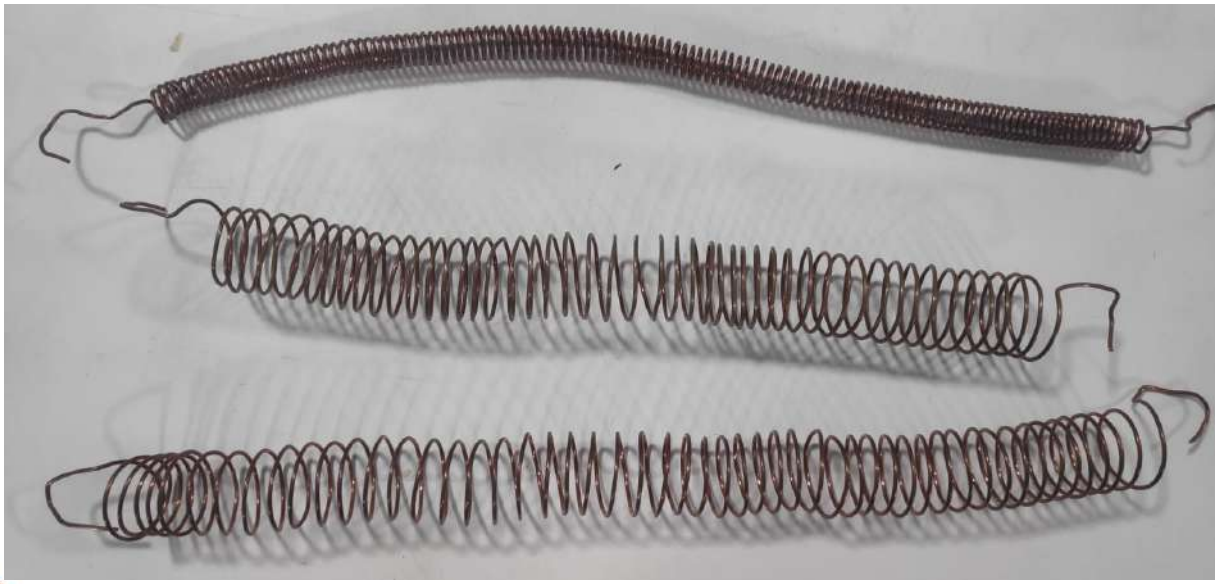
Para conectar os flanges às hastes, foram utilizados parafusos de aproximadamente 2 centímetros de comprimento e 5 milímetros de diâmetro, acompanhados de porcas sextavadas, assegurando uma fixação firme e confiável (figura 2.19).

Figura 2.19 – Parafusos utilizados no acoplamento dos flanges às hastes.

Fonte: Autoria própria

2.1.8 **Mola**

Figura 2.20 – molas metálicas de diferentes tamanhos e espessuras utilizados no experimento.

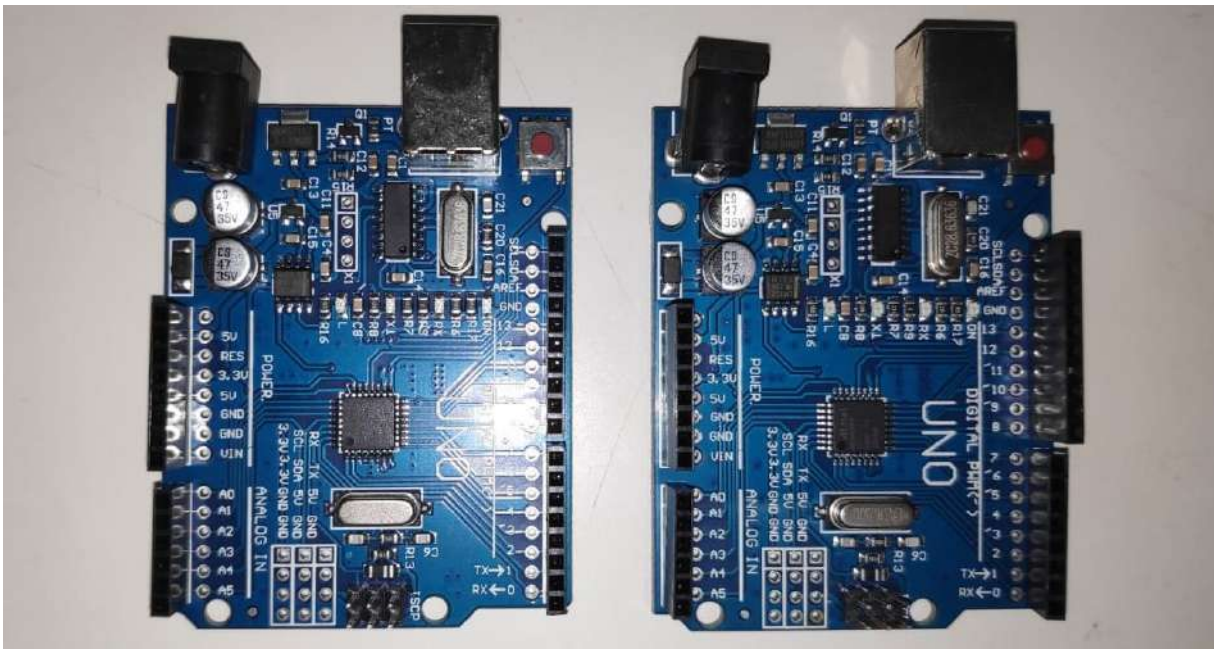


Fonte: Autoria própria

É um objeto elástico flexível de metal que tem como objetivo, nesse aparato, acoplar um pêndulo ao outro. Foram utilizadas três molas com comprimentos entre 30 e 50 centímetros, e diâmetros entre 1 e 2 centímetros, todas com leve constante elástica e construídas a partir de fios de cobre de aproximadamente 4 metros de comprimento e 1 milímetro de espessura. Graças à mobilidade permitida pelos blocos deslizantes, há a possibilidade de utilização de molas de diferentes comprimentos. A figura 2.20 mostra as molas utilizadas na aplicação do produto.

2.1.9 **Placas Arduíno**

Figura 2.21 – Microcontroladores Arduíno UNO, mostrando componentes eletrônicos como portas USB, pinos de entrada e saída, e conectores de alimentação.



Fonte: Autoria própria

o Arduíno é um microcontrolador programável de prototipagem eletrônica. Possui código aberto, o que possibilita a dinamização do seu uso. Ao receber o sinal do encoder óptico, converte e envia para a porta serial. Os dados dessa conversão podem ser visualizados em programas capazes de estabelecer comunicação serial ou mesmo no próprio monitor serial da IDE do Arduíno.

Essas placas são amplamente disponíveis no mercado, podendo ser adquiridas com facilidade em lojas de materiais eletrônicos ou plataformas de e-commerce. As especificações técnicas do modelo utilizado neste experimento estão detalhadas a seguir:

- **Microcontrolador:** ATmega328P (28 pinos);
- **Tensão de operação:** 5V;
- **Tensão de entrada (recomendada):** 7-12V;

- **Pinos digitais de entrada/saída (I/O):** 14 (dos quais 6 podem ser usados como saídas PWM);
- **Pinos analógicos:** 6;
- **Corrente máxima por pino I/O:** 20 mA;
- **Memória flash:** 32 KB (dos quais 0,5 KB usados pelo bootloader);
- **SRAM:** 2 KB;
- **EEPROM:** 1 KB;
- **Velocidade do clock:** 16 MHz;
- **Conectividade USB:** Tipo B (para comunicação e alimentação);
- **Conector de energia:** Barril de 2,1 mm (centro positivo);
- **Conector ICSP:** Para gravação do firmware no microcontrolador;
- **Dimensões:** 68,6 mm x 53,4 mm;

2.1.10 Cabos USB

Figura 2.22 – Cabos USB tipo B

Fonte:Autoria própria

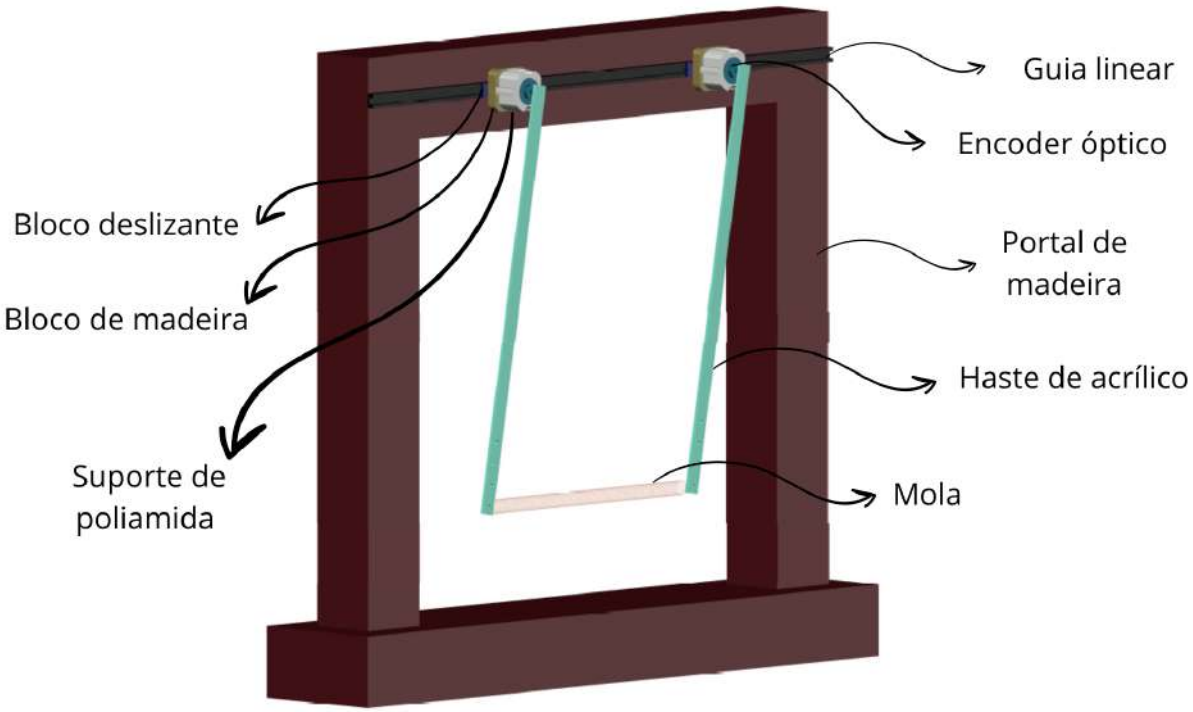
Foram utilizados dois cabos USB tipo B de 1,5 metros cada (figura 2.22), para conectar as placas Arduíno ao computador. Os cabos têm por função transmitir os dados entre os dispositivos utilizados no aparato.

2.2 Montagem e Estrutura

A montagem do sistema experimental foi projetada para garantir simplicidade, funcionalidade e precisão na realização dos experimentos. A configuração busca facilitar o manuseio do aparato, proporcionando praticidade sem comprometer a confiabilidade da coleta de dados.

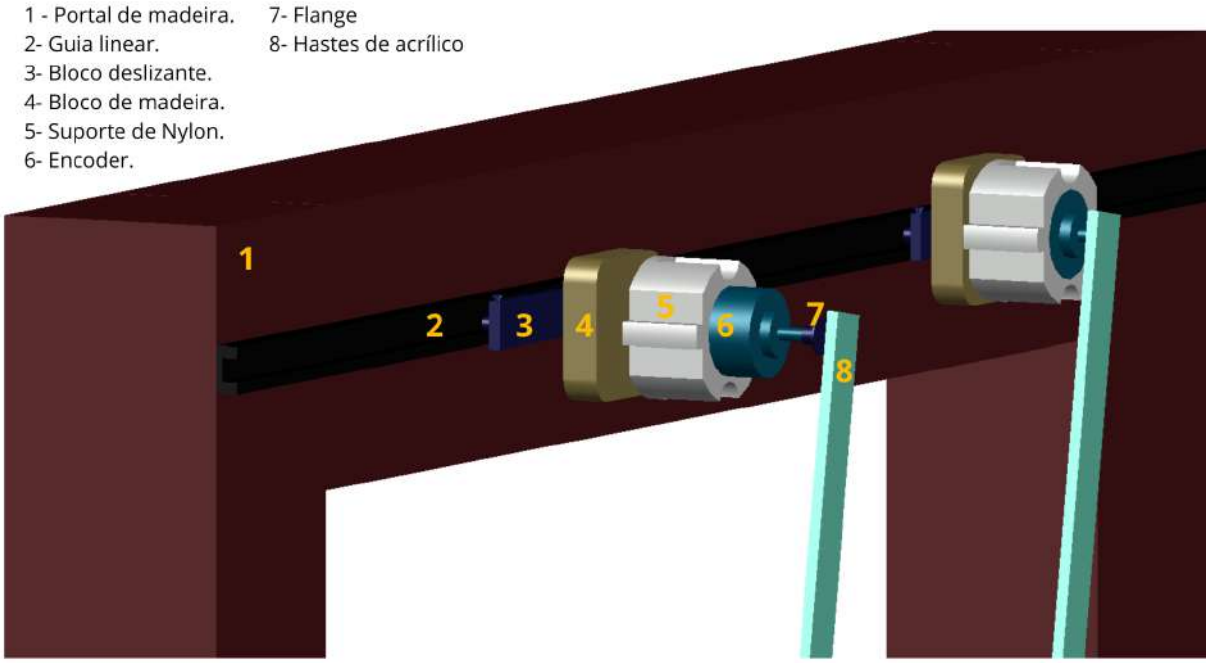
As Figuras 2.23 e 2.24 apresentam o esquema projetado da estrutura do sistema físico com indicação das principais peças constituintes, elaborado utilizando o software de desenho técnico AutoCAD. Esse esquema fornece uma visão clara e precisa dos componentes e da disposição geométrica do sistema, servindo como base para sua construção e montagem.

Figura 2.23 – Estrutura física em AutoCAD com indicação dos principais elementos.



Fonte: Autoria própria

Figura 2.24 – Representação ampliada da estrutura física com foco nos principais componentes.



Fonte: Autoria própria

Este tópico descreve de forma detalhada o processo de montagem, abordando o posicionamento dos componentes, as conexões necessárias entre eles e os ajustes que garantem o funcionamento adequado.

2.2.1 **Etapa 1 - Fixação da guia ao portal**

Figura 2.25 – Guia linear fixada ao portal de madeira.

Fonte: Autoria própria

Figura 2.26 – Detalhe lateral da montagem da guia linear sobre portal de madeira.

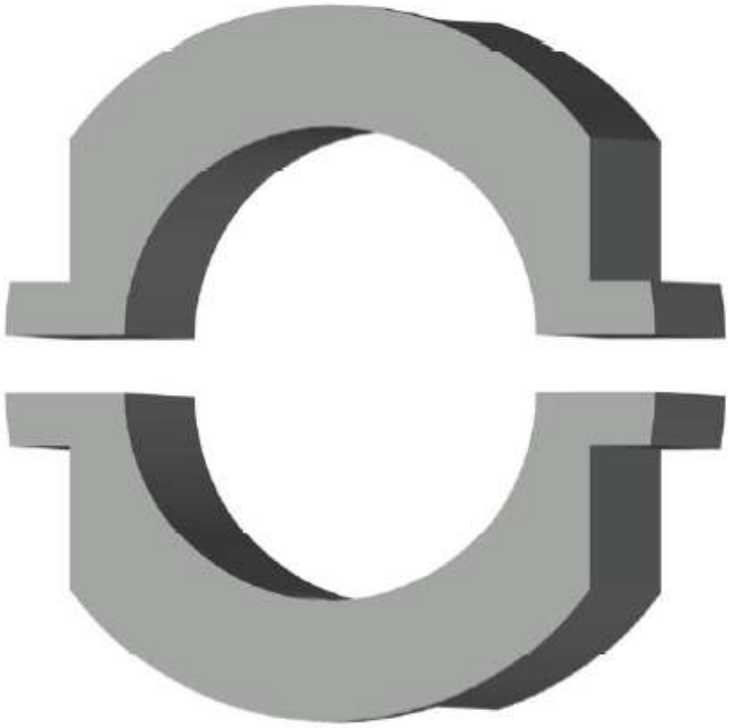
Fonte: Autoria própria

A primeira etapa da montagem do equipamento envolve a fixação da guia linear ao portal de madeira. Para facilitar esse processo, a guia linear possui furos distribuídos ao longo de sua extensão, conforme ilustrado nas Figuras 2.25 e 2.26. Para garantir estabilidade e segurança na montagem, foram utilizados parafusos como elemento de fixação.

2.2.2 **Etapa 2 - Acoplamento dos encoders aos blocos deslizantes**

Os encoders ópticos foram fixados à guia linear, que, por sua vez, está presa ao portal de madeira. Para essa montagem, foram utilizados suportes de poliamida (mostrado na figura 2.12), fabricados em uma tornearia, com base em especificações elaboradas previamente em desenho técnico (figura 2.27). Esses suportes possuem cerca de 10 centímetros de diâmetro e 5 centímetros de altura, dimensões projetadas para atender às necessidades do sistema. Cada suporte foi dividido em duas partes, permitindo o encaixe preciso e seguro dos encoders. Após o encaixe, as duas metades foram unidas novamente por parafusos.

Figura 2.27 – Suporte para os encoders.



Fonte: Autoria própria

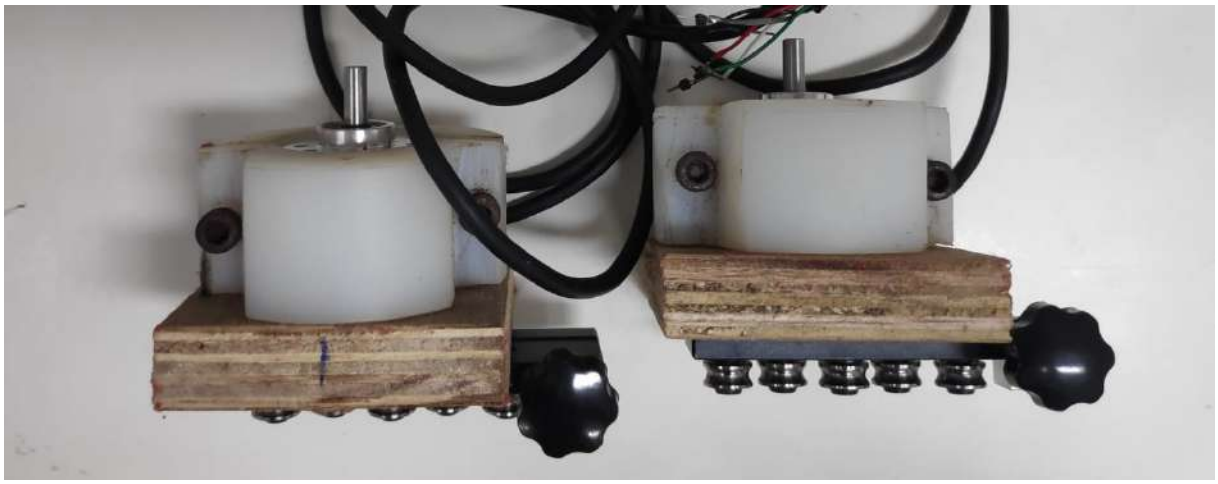
Os suportes de poliamida foram fixados a blocos de madeira MDF, confeccionados em uma marcenaria. Cada placa possui formato quadrado com aproximadamente 10 centímetros de lado. Essas placas foram acopladas ao bloco deslizante da guia linear (figura 2.14), utilizando parafusos, permitindo uma movimentação suave e controlada.

As Figuras 2.28, 2.29 e 2.30 apresentam a organização detalhada desses elementos, destacando sua disposição no sistema e evidenciando como cada componente contribui para o funcionamento eficiente do aparato experimental.

Figura 2.28 – Acoplamento dos encoders aos blocos deslizantes.

Fonte: Autoria própria

Figura 2.29 – Acoplamento dos encoders aos blocos deslizantes - vista superior dos componentes.



Fonte: Autoria própria

Figura 2.30 – Acoplamento dos encoders aos blocos deslizantes - vista frontal e posterior.

Fonte: Autoria própria

Feito o acoplamento, os blocos deslizantes foram restituídos à guia linear. A figura 2.31 mostra a disposição desses elementos fixos ao portal de madeira.

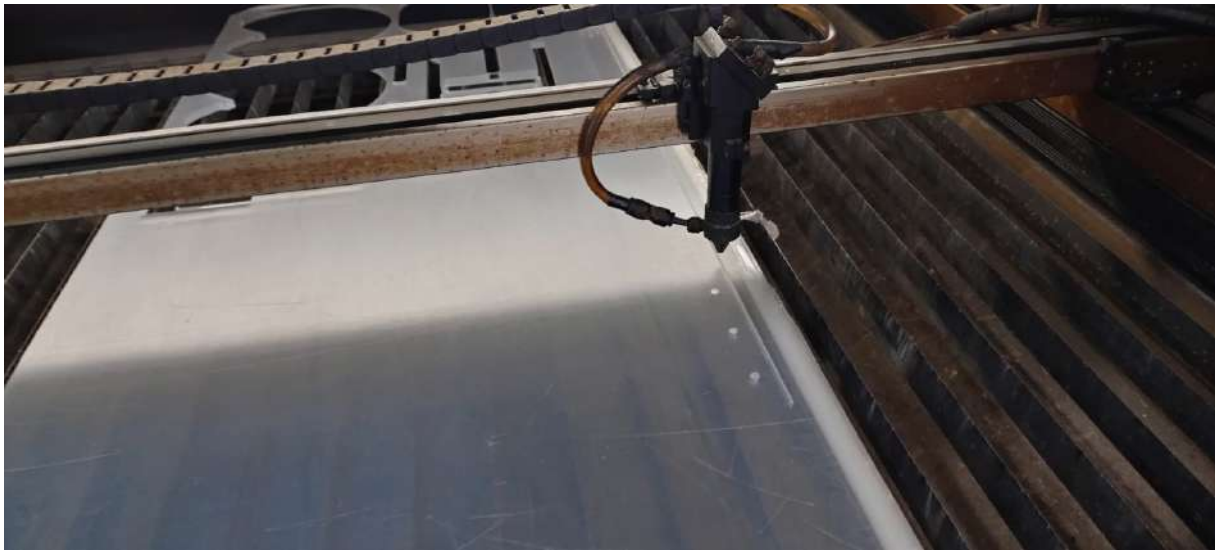
Figura 2.31 – Montagem dos encoders ópticos na guia linear.

Fonte: Autoria própria

2.2.3 **Etapa 3 - Montagem das hastes**

Conforme mencionado anteriormente, a concepção dos pêndulos contou com a utilização de duas hastes de acrílico, cada uma com 80 centímetros de comprimento. Essas hastes foram projetadas e fabricadas por meio de corte a laser, conforme ilustrado na figura 2.32. Esse método de fabricação foi escolhido por sua alta precisão, assegurando dimensões exatas e acabamento de qualidade, essenciais para o desempenho adequado do sistema experimental.

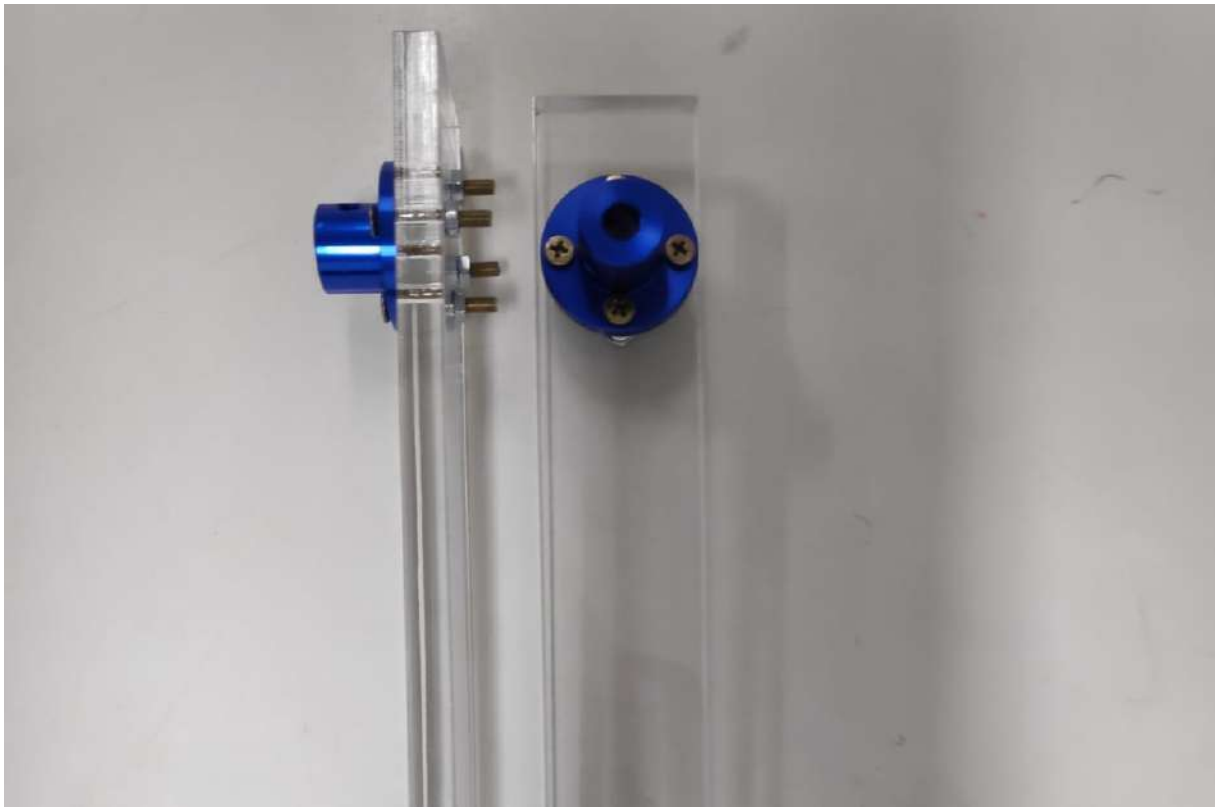
Figura 2.32 – Hastes de acrílico sendo manufaturadas na cortadora laser.



Fonte: Autoria própria

A fixação das hastes aos encoders é realizada por meio dos flanges de alumínio descritos no tópico 2.1.7. A Figura 2.33 ilustra a disposição detalhada dos flanges, hastes e parafusos, destacando como esses elementos foram organizados.

Figura 2.33 – Hastes acopladas aos flanges por parafusos.

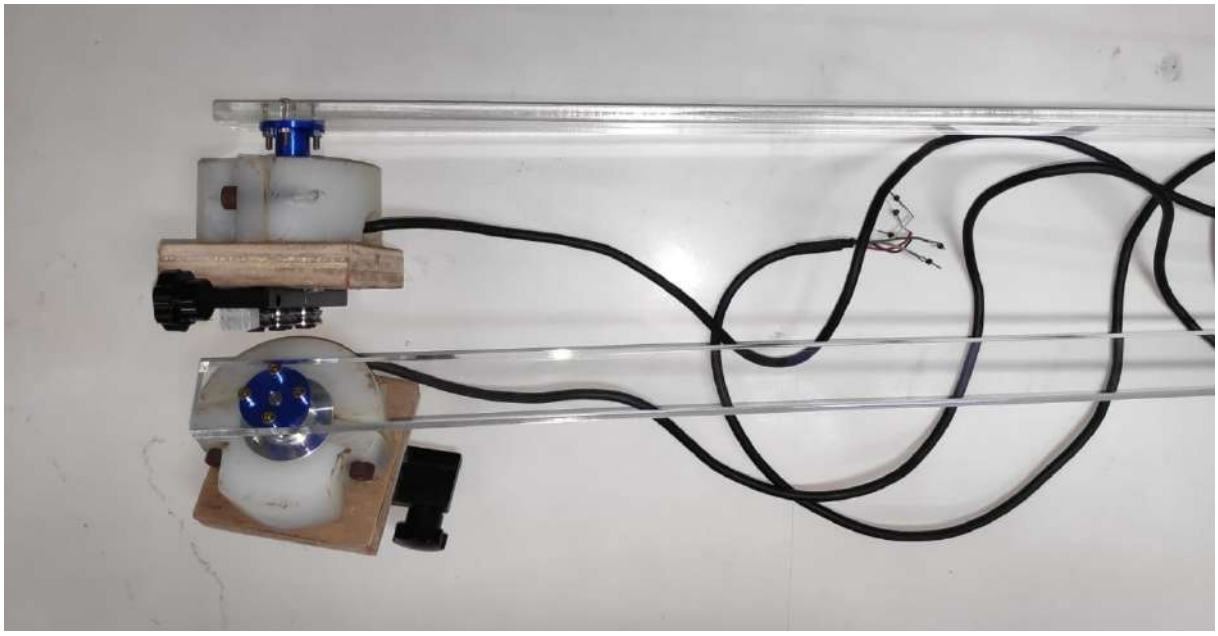


Fonte: Autoria própria

Os flanges foram acoplados ao eixo dos encoders utilizando parafusos reguladores,

inseridos nos orifícios integrados à estrutura dos próprios flanges. Essa configuração assegura um ajuste preciso e firme, como é visto na imagem 2.34.

Figura 2.34 – Hastes acopladas aos encoders.



Fonte: Autoria própria

2.2.4 **Etapa 4 - Posicionamento da mola**

A última etapa antes da inclusão dos elementos eletrônicos é o posicionamento da mola que conecta os dois pêndulos. Para isso, as hastes foram projetadas com pequenos orifícios em suas extremidades, especialmente concebidos para acomodar a mola e viabilizar o acoplamento. A mola utilizada pode ter tamanhos variados, pois as hastes deslizam nos trilhos, permitindo ajustar a distância entre elas de 0 a 90 centímetros.

Recomenda-se a utilização de molas com diferentes tamanhos, diâmetros e constantes elásticas, ampliando as possibilidades experimentais e permitindo a exploração de uma maior variedade de cenários físicos. A configuração final dessa etapa estrutural, antes da inclusão dos componentes eletrônicos, é apresentada nas figuras 2.35 e 2.36.

Figura 2.35 – Disposição final dos componentes mecânicos.

Fonte: Autoria própria

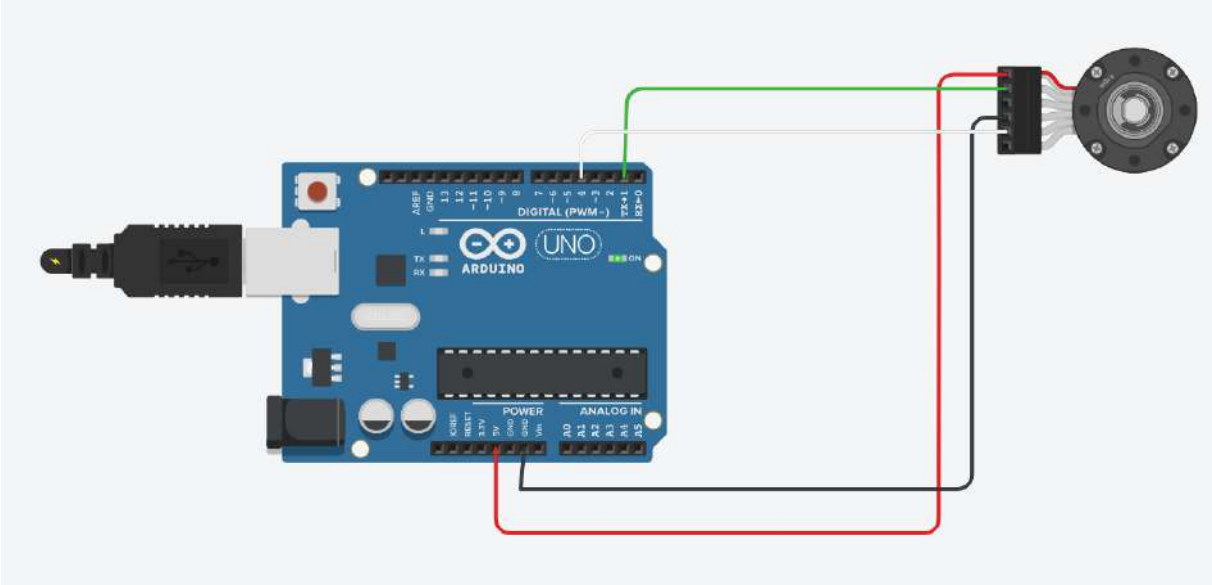
Figura 2.36 – Vista lateral da estrutura de madeira com os elementos inseridos .

Fonte: Autoria própria

2.2.5 **Etapa 5 - Conexão das placas Arduíno**

A conexão da placa Arduíno ao encoder ocorre através dos próprios fios do encoder, sem a necessidade de mecanismos intermediários, como protoboards. A imagem 2.37 mostra uma esquematização de como ocorrem essas conexões.

Figura 2.37 – Ligação entre encoder e placa Arduíno.

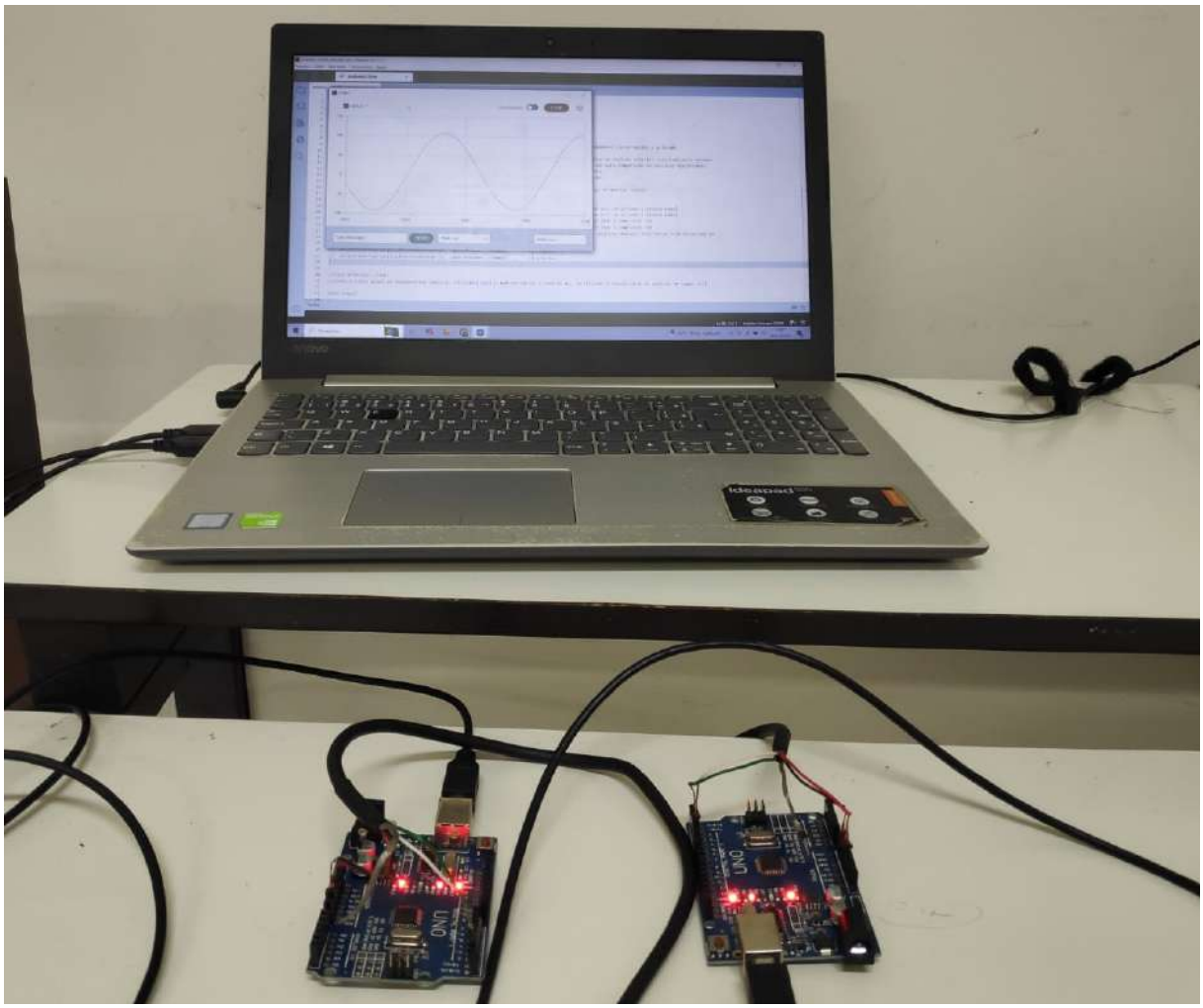


Fonte: Autoria própria

O fio vermelho (VCC ou +5V/+24V) fornece a alimentação elétrica ao encoder, ou seja, é responsável por energizar o dispositivo. Esse fio é conectado ao pino 5V da placa Arduíno. O fio preto (GND) serve como referência de terra para o circuito, devendo ser conectado ao terra (GND) do sistema para completar o circuito de alimentação elétrica. O fio branco (canal A) emite pulsos digitais que indicam o deslocamento angular do eixo no primeiro canal de saída. Esses pulsos são usados para determinar o movimento e a velocidade do eixo. Esse fio deve ser conectado a um dos pinos digitais com capacidade de interrupção, como os pinos 2 ou 3 no Arduíno, para monitorar os pulsos do encoder em tempo real. Esse pino é definido no código. Por fim, o fio verde (canal B) tem a função de fornecer pulsos deslocados em 90° em relação ao canal A, permitindo identificar a direção do movimento. A combinação dos sinais dos canais A e B possibilita detectar o sentido de rotação do eixo. Esse fio deve ser conectado a outro pino digital, que também pode ser lido pelo Arduíno e estará definido no código.

Feita a conexão dos encoders às placas Arduíno, estas são conectadas ao computador através dos cabos USB tipo B (figura 2.22). A figura 2.38 mostra o *setup* experimental envolvendo a conexão entre o computador e as duas placas Arduíno.

Figura 2.38 – *Setup* experimental com Arduíno para coleta automatizada de dados do sistema oscilatório desenvolvido.



Fonte: Autoria própria

Finalizando esta etapa, o equipamento está pronto para o uso. A visualização gráfica dos dados pode ser realizada na própria ferramenta integrada à IDE do Arduíno, o Serial Plotter. No entanto, existem softwares com interfaces melhoradas que podem ser utilizados para esta função e que podem atuar em conjunto com a IDE, como o software de programação gráfica LabVIEW. Esses softwares são detalhados na seção seguinte.

3 AUTOMAÇÃO DA AQUISIÇÃO DE DADOS

A revolução tecnológica vivida nos últimos anos transformou o *modus operandi* dos principais setores da sociedade. A integração de mecanismos que facilitam a vida humana ou otimizam seus esforços tornou-se uma realidade. Na indústria, por exemplo, os processos de produção estão passando por uma transição. Alguns ainda são operados manualmente, enquanto outros seguem o caminho inevitável da automação (RIFKIN, 2011).

Um processo automatizado pode ser entendido como um sistema operacional que requer pouca ou nenhuma intervenção humana. Nesses sistemas físicos, são utilizados atuadores, controladores e sensores para desenvolver tarefas que, muitas vezes, são repetitivas ou exigem um nível de precisão que o ser humano não é capaz de alcançar. Outro aspecto que justifica a automação de processos é a eficiência e segurança na realização das tarefas. Um processo automatizado, quando operado sob parâmetros precisos, não abre margem para as incorreções típicas da atividade humana (GROOVER, 2016).

Na indústria, a automação dos sistemas de produção pode ser agrupada em duas categorias: automação da própria produção e informatização dos sistemas de apoio à produção. Apesar dessa divisão, as duas categorias estão intimamente relacionadas, visto que os sistemas de produção fabris automatizados na operação são implementados por sistemas informatizados, e estes, por sua vez, são incorporados aos sistemas de apoio à produção (LEE; BAGHERI; KAO, 2015).

Para além da indústria, a automação dos processos está presente em áreas que demandam a coleta de dados precisos para análise e controle, como em pesquisas científicas, onde é indispensável para garantir a precisão e a reprodutibilidade dos experimentos. Muitos desses métodos de automação começam na própria instrumentação, que se caracteriza pelo uso de dispositivos para realizar medições, monitoramento e controle das variáveis mensuradas. Esses mecanismos instrumentais, assim como todo o processo de automação, integram sensores, controladores, transdutores e outros dispositivos semelhantes (DOEBELIN; MANIK, 2007).

3.1 Hardware e Software

Esta seção apresenta os componentes físicos e as ferramentas digitais utilizadas no desenvolvimento e funcionamento do sistema experimental. Nessa abordagem, detalha-se o hardware empregado (sensores e microcontroladores) e demais elementos estruturais, bem como os softwares responsáveis pela aquisição automática de dados, simulação computacional e controle do sistema. Essa integração entre hardware e software é essencial para garantir precisão, funcionalidade e eficiência nos experimentos realizados, além de proporcionar uma interface acessível para o usuário.

3.1.1 Hardware

O aprimoramento da visualização de fenômenos físicos está intimamente relacionado ao desenvolvimento de tecnologias acessíveis e de fácil manuseio. Com a automatização da aquisição de dados, é possível realizar uma análise detalhada e precisa, já que a captação pode ocorrer em tempo real, permitindo o ajuste das informações de acordo com os parâmetros estabelecidos dos osciladores.

A automação de sistemas pendulares acoplados pode ser realizada utilizando sensores disponíveis no mercado, que possuem preços acessíveis e ampla compatibilidade com mecanismos de aquisição de dados. Os encoders ópticos incrementais são exemplos de mecanismos que desempenham essa função, pois são capazes de captar fenômenos físicos, como posição, direção, velocidade linear e angular, entre outros (BISINOTTO; CANCIAN; DE OLIVEIRA, 2015).

3.1.1.1 ***Funcionamento do encoder óptico incremental***

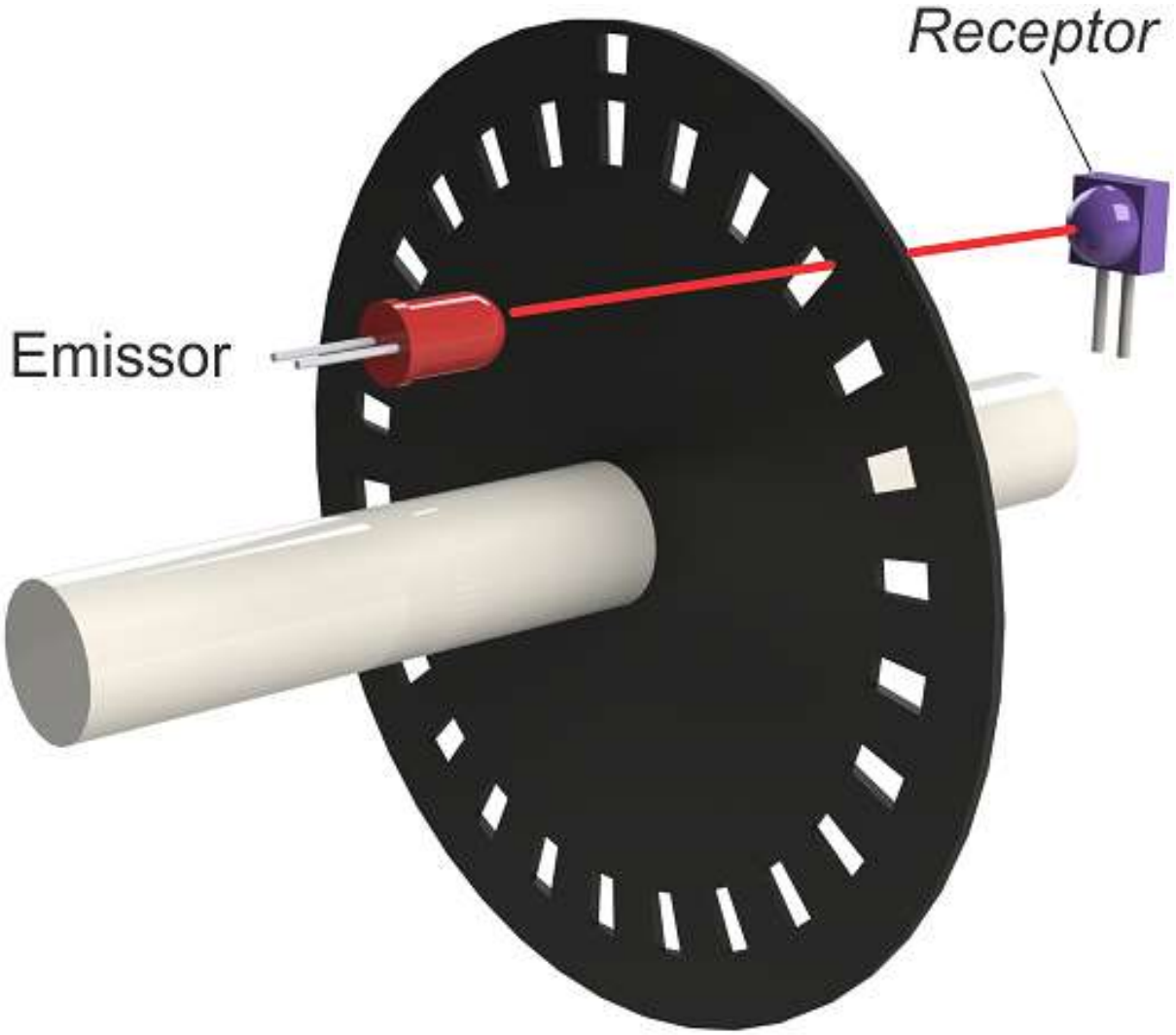
Quando se busca controle e precisão em sistemas automáticos, os encoders ópticos incrementais são uma escolha viável, especialmente para análise de posições ou movimentos angulares. Sua função principal é converter sinais mecânicos lineares ou rotativos em pulsos analógicos ou digitais. São amplamente utilizados para medir rotações por minuto (RPM) de motores elétricos, como motores de passo, e também para monitorar posição, velocidade, movimento e direção em outros dispositivos (FERRAZ; OLIVEIRA JUNIOR, 1992). A figura 3.1 mostra um encoder incremental óptico de 1000 pulsos.

Figura 3.1 – Encoder incremental óptico.

Fonte: Autoria própria

Esse mecanismo é composto, basicamente, por um disco com marcações que bloqueia ou libera o feixe de luz de um LED para o fotodetector, o qual inclui um emissor e um receptor (figura 3.2). As marcações no disco estão relacionadas à resolução do equipamento, pois, à medida que o disco gira, um sinal em forma de onda quadrada (clock) é enviado pelo fotodetector e pelo circuito eletrônico para as saídas do encoder. Esse sinal é proporcional ao número de marcações no disco, determinando sua resolução. A quantidade de pulsos está diretamente relacionada à precisão na captura da posição. No entanto, fatores como erro de excentricidade, desalinhamento do sensor, variações na largura das marcações do disco, ruído eletrônico e interferências ópticas podem introduzir erros sistêmicos na medição. Nessas condições, mesmo uma alta resolução não será suficiente para corrigir tais imprecisões. (INÁCIO, 2009).

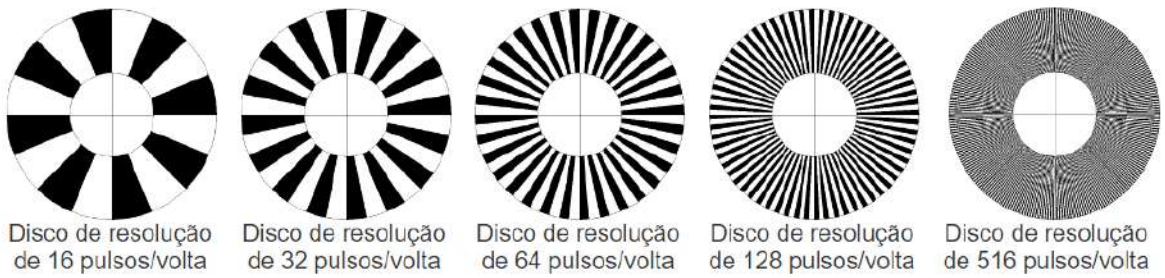
Figura 3.2 – Elementos constituintes do encoder.



Fonte: HI Tecnologia

Na figura 3.3 é possível visualizar alguns exemplos de discos e sua quantidade de marcações (pulsos). Esse valor faz referência ao número de pulsos elétricos gerados a cada rotação completa em torno do eixo no qual o encoder está instalado.

Figura 3.3 – Resoluções de encoders.

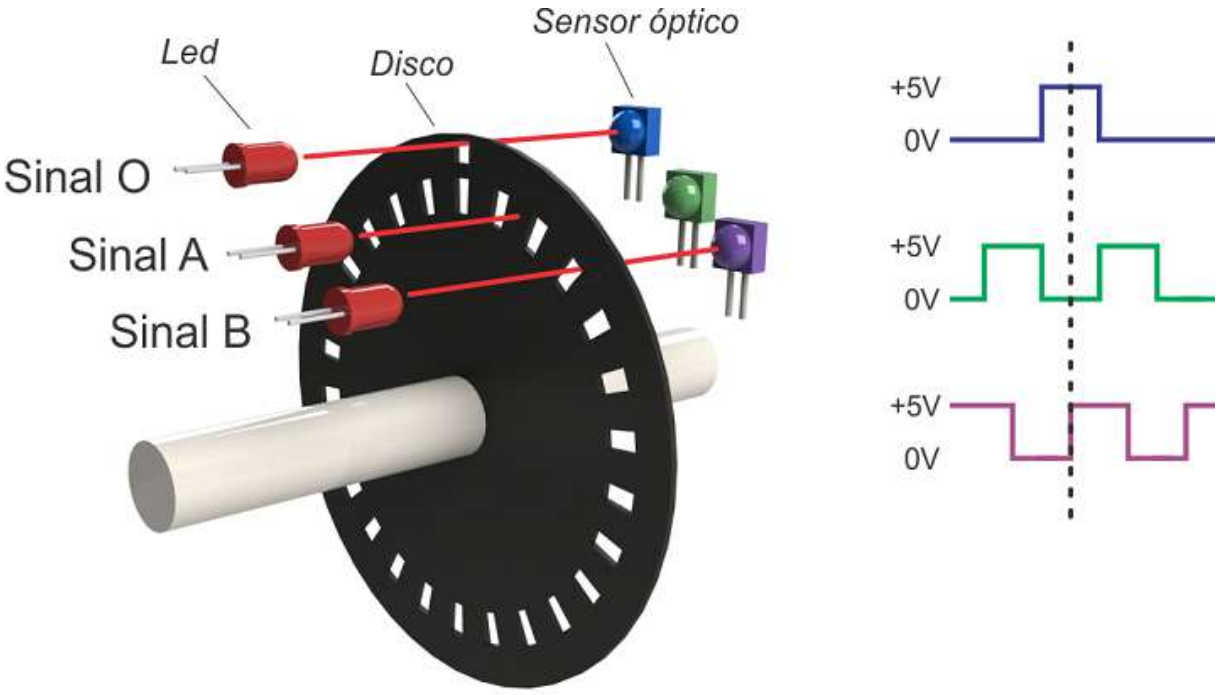


Fonte: HI Tecnologia

A particularidade dos encoders ópticos incrementais, amplamente utilizados, reside

nas suas múltiplas saídas. Os modelos mais comuns possuem três saídas: “A”, “B” e “O”. Essas saídas estão associadas, respectivamente, ao ângulo de rotação, medido pelos pulsos conforme o disco gira, ao sentido da rotação, com defasagem angular de ±90°, e à indicação de quando uma volta se encerra e outra começa (TIAGO, 2013). A figura 3.4 mostra a estrutura esquematizada dos principais elementos desse tipo de encoder.

Figura 3.4 – Estrutura do encoder óptico incremental.



Fonte: HI Tecnologia

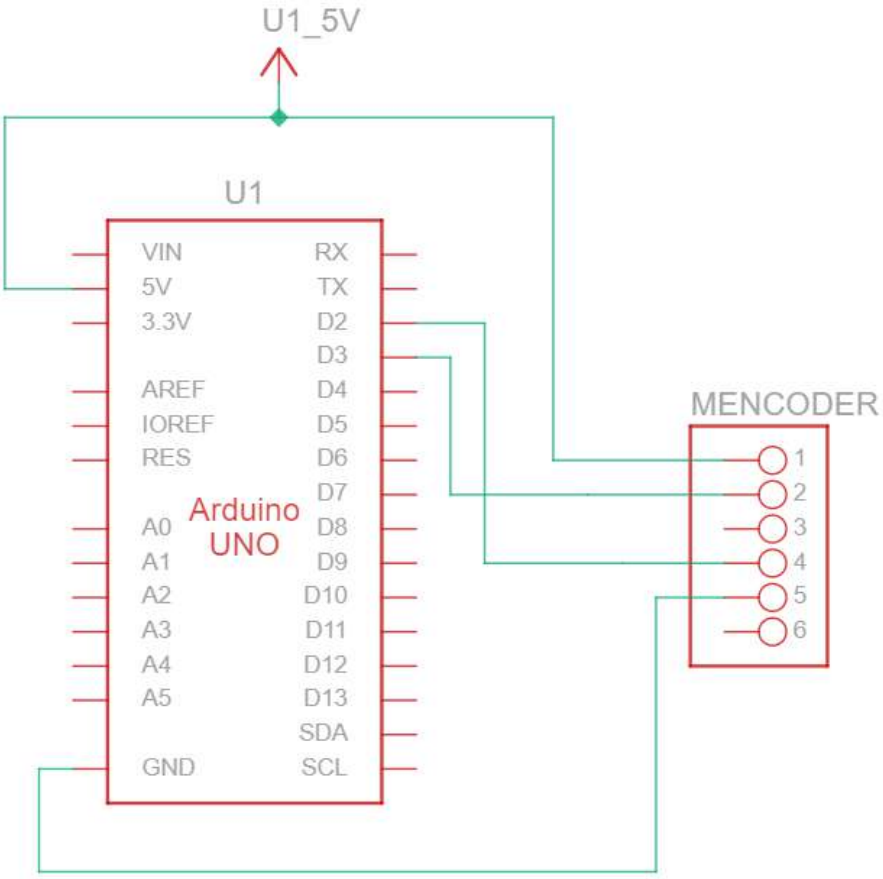
3.1.1.2 ***Placa Arduíno UNO e microcontrolador***

A placa programadora Arduíno é uma plataforma *open-source*, o que permite que qualquer indivíduo possa contribuir com o seu desenvolvimento. Seu código fonte está disponível publicamente e a programação física baseada em uma placa com microcontrolador simples, garante uma democratização de objetivos desejados pelos utilizadores por benefícios que incluem não somente a customização do código, mas também colaboração entre desenvolvedores, transparência e custo. A vasta compatibilidade com sensores e interruptores permite que a placa Arduíno seja utilizada no desenvolvimento de objetos interativos e atue na visualização e controle de fenômenos associados ao funcionamento de motores, luzes, entre outros dispositivos físicos. Desta forma, é possível constatar as potencialidades integrativas entre computador e mundo físico obtidas a partir dessa ferramenta poderosa e versátil (SILVA; ARAUJO; CAVALCANTE, 2019).

Para realizar medidas comportamentais de osciladores acoplados, por exemplo, a placa programável Arduino recebe os sinais gerados pelo encoder a partir da movimentação dos pêndulos. A partir desses dados, ela é capaz de monitorar as posições angulares em tempo real. Com base nas informações obtidas, é possível analisar o comportamento completo do sistema, como velocidade, variações de ângulo, frequência e as características específicas do acoplamento.

No geral, os encoders ópticos incrementais possuem 4 ou 5 pinos e a ligação entre o sensor e a placa Arduíno é bastante intuitiva. A figura 3.5 mostra uma esquematização de como ocorre a conexão entre a plataforma Arduíno e o encoder. Como já mencionado no tópico 2.2.5, o pino VCC (Voltage at the Common Collector) é responsável pela alimentação do circuito. Normalmente, apresenta uma tensão de 5V e é ligado diretamente à placa. Já o terra, utilizado como ponto de referência de tensão, é ligado na porta GND. Os canais A e B são responsáveis pela captação da rotação do eixo do encoder; quando o movimento acontece, enviam pulsos indicando a orientação do giro. Esses pinos são ligados aos canais digitais da placa Arduíno responsáveis pela leitura dos pulsos recebidos.

Figura 3.5 – Vista esquemática do circuito.



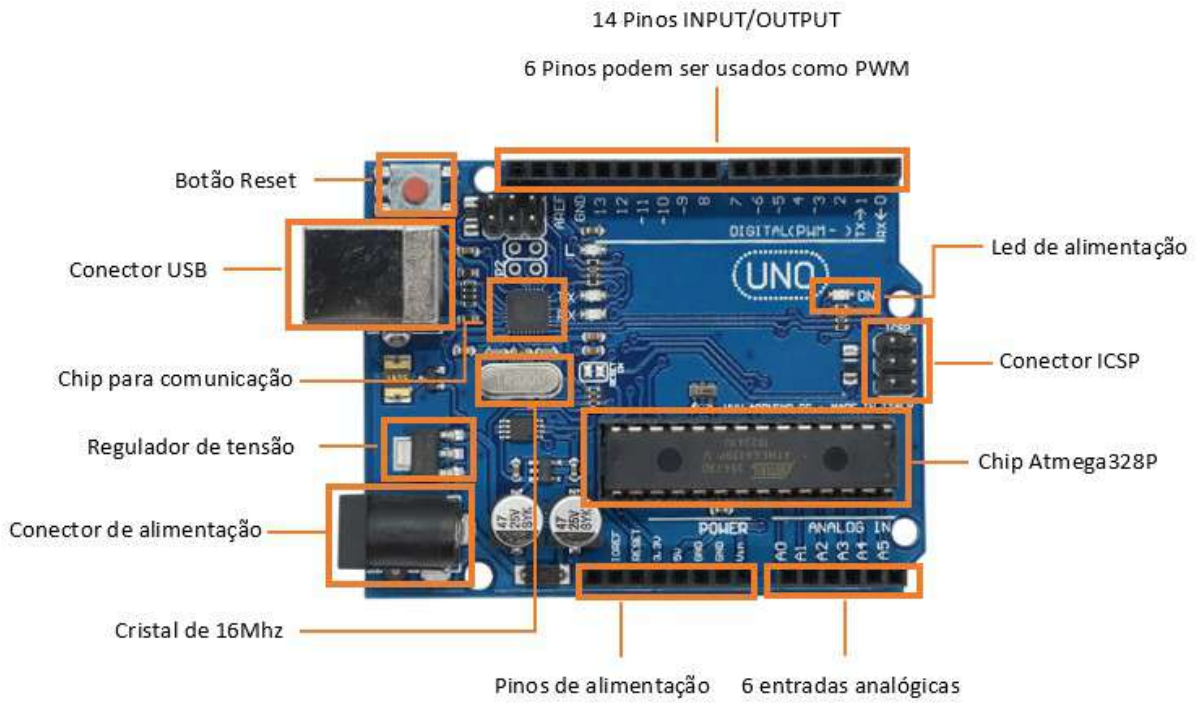
Fonte: Autoria própria

O componente principal da placa Arduino é o microcontrolador central, além dos pinos de entrada/saída analógicos e digitais, o que permite a comunicação com uma grande variedade de sensores. O Arduino UNO, um dos modelos mais utilizados atualmente, por exemplo, utiliza um controlador ATmel ATmega328p, um dispositivo de 8 bits da classe AVR, com "arquitetura RISC avançada e encapsulamento DIP28."Ele conta com 32 kB de Flash (sendo 512 bytes utilizados pelo bootloader), 2 kB de RAM e 1 kB de EEPROM (MCROBERTS, 2015).

A plataforma possui hardware open source, permitindo que qualquer pessoa crie seu próprio dispositivo Arduino, podendo modificá-lo ou adaptá-lo conforme suas necessidades. O software utiliza uma linguagem baseada em C/C++ e conta com uma interface simplificada (IDE), o que possibilita o desenvolvimento e a execução do código diretamente na placa (MCROBERTS, 2015).

A figura 3.6 mostra os principais elementos de uma placa de prototipagem Arduíno, modelo UNO.

Figura 3.6 – Placa Arduíno UNO.



Fonte: Embarcados

3.1.2 Software

O desenvolvimento de sistemas experimentais modernos não se limita à utilização de componentes físicos; o software desempenha um papel igualmente crucial ao proporcionar ferramentas para aquisição, processamento e análise dos dados coletados. A programação personalizada permite não apenas automatizar tarefas, mas também adaptar os parâmetros de medição às necessidades específicas do experimento. Nesse contexto, plataformas como o Arduíno, associadas a linguagens de programação como Python, oferecem recursos versáteis para integrar hardware e software, garantindo precisão e eficiência nos experimentos realizados.

3.1.2.1 ***Linguagem desenvolvimento Python***

A linguagem de programação Python é amplamente utilizada em aplicações práticas na área acadêmica e fora dela. Isso se dá, principalmente, devido à sua versatilidade e

simplicidade de utilização. Trata-se de uma linguagem de código aberto de alto nível com sintaxe intuitiva, o que beneficia seu aprendizado e implementação, especialmente em ambientes educacionais. Essa ferramenta possui uma vasta biblioteca de pacotes que acarreta na ampliação de possibilidades como simulações numéricas e visualizações gráficas que, associadas à experimentação, podem representar um artifício poderoso para a observação de fenômenos e análise de dados.

Criada no final da década de 1980 pelo programador holandês Guido Van Rossum, funcionário do Centrum Wiskunde & Informatica (CWI) em Amsterdã, a linguagem de programação Python surgiu com a proposta inicial de ser uma linguagem de código aberto, de fácil leitura e escrita. A primeira versão foi lançada em fevereiro de 1991 e tinha como inspiração a linguagem de ensino ABC, a linguagem C, Perl e Lisp no intuito de unir simplicidade e poder de expressão. (ROSSUM, s.d.)

A simplicidade proposta na criação da linguagem Python tinha por intuito a descentralização dos esforços, geralmente associados à complexidade da sintaxe, e na concentração maior na resolução de problemas. O termo Python, tem origem curiosa, é inspirado no programa de TV da BBC intitulado "Monty Python's Flying Circus", um humorístico do qual Guido Van Rossum era um grande fã. (VAN ROSSUM et al., 2007)

Atualmente o Python segue em constante evolução, apoiado por diversos colaboradores e mantido pela Python Software Foundation (PSF), uma organização sem fins lucrativos que fomenta a evolução da linguagem e suas diversas aplicações. A diversidade de pacotes disponíveis e compatíveis à linguagem Python possibilita a aplicação em áreas diversas como automação, inteligência artificial, ciência de dados e desenvolvimento web. (VAN ROSSUM; DRAKE JR, 1995)

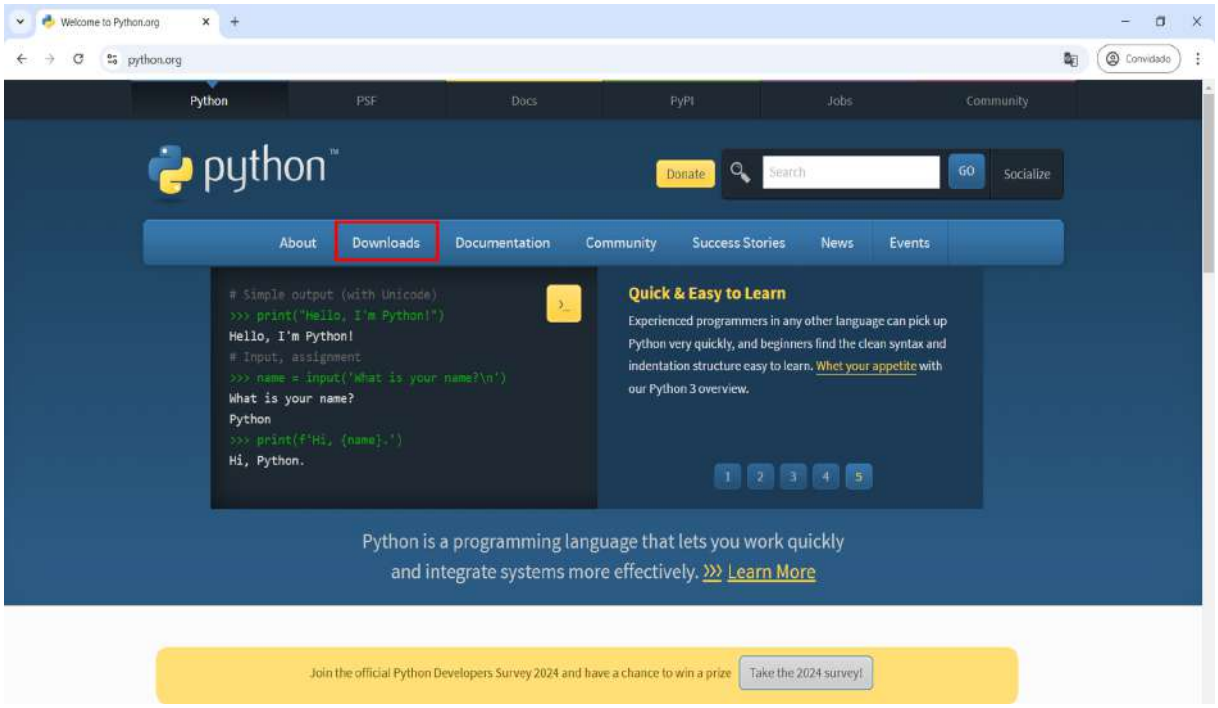
3.1.2.1.1 **Instalando o Python no computador**

A instalação do Python é bastante intuitiva e não requer grandes habilidades para ser realizada. Basta seguir os seguintes passos:

- 1º passo: acessar o site oficial da Python Software Foundation.

Em um computador, com acesso à internet, é necessário acessar o endereço da Python Foundation (https://www.python.org/). Após o acesso, será exibida a home page da fundação (figura 3.7). A partir daí, basta seguir os passos seguintes.

Figura 3.7 – Home page da Python Foundation.

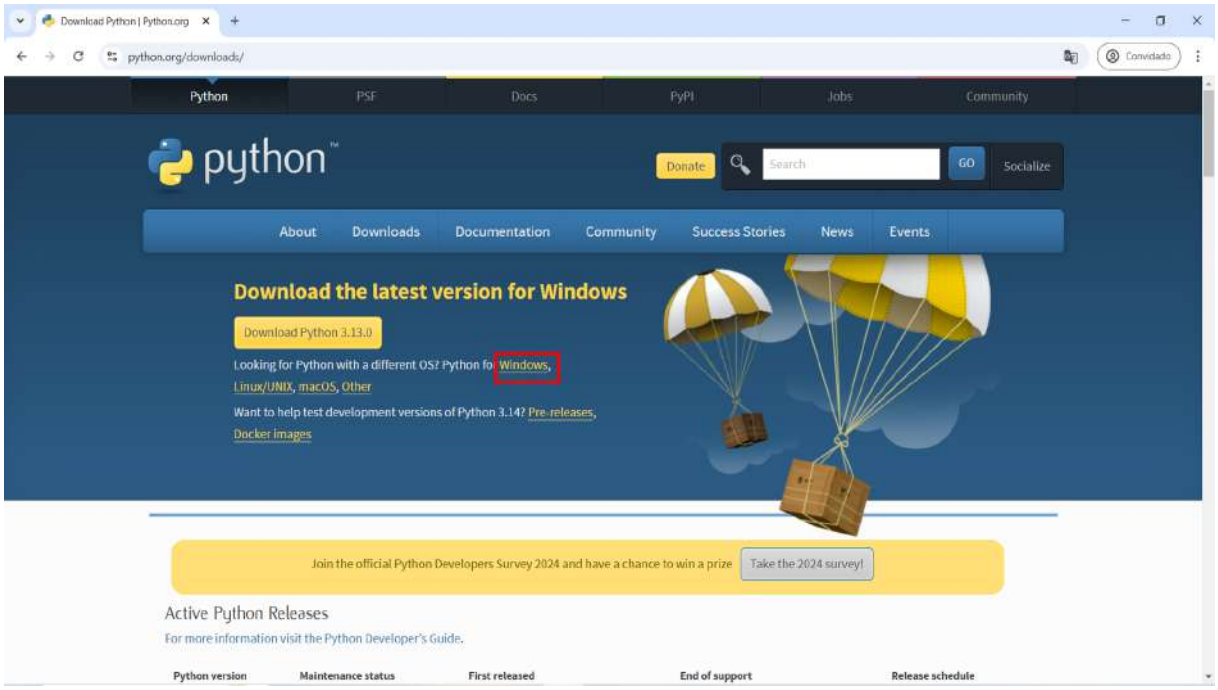


Fonte: Autoria própria

- 2º passo: Escolher o software de acordo com o sistema operacional do computados em que se deseja instalar o Python.tion.

Ao clicar em download na tela inicial, o usuário é direcionado à tela seguinte onde é possível escolher o sistema operacional ao qual deseja instalar o programa (figura 3.8). Para esse tutorial, será utilizado um computador com sistema Windows, mas há disponibilidade para Linux, macOS e outros.

Figura 3.8 – Sistemas operacionais disponíveis.

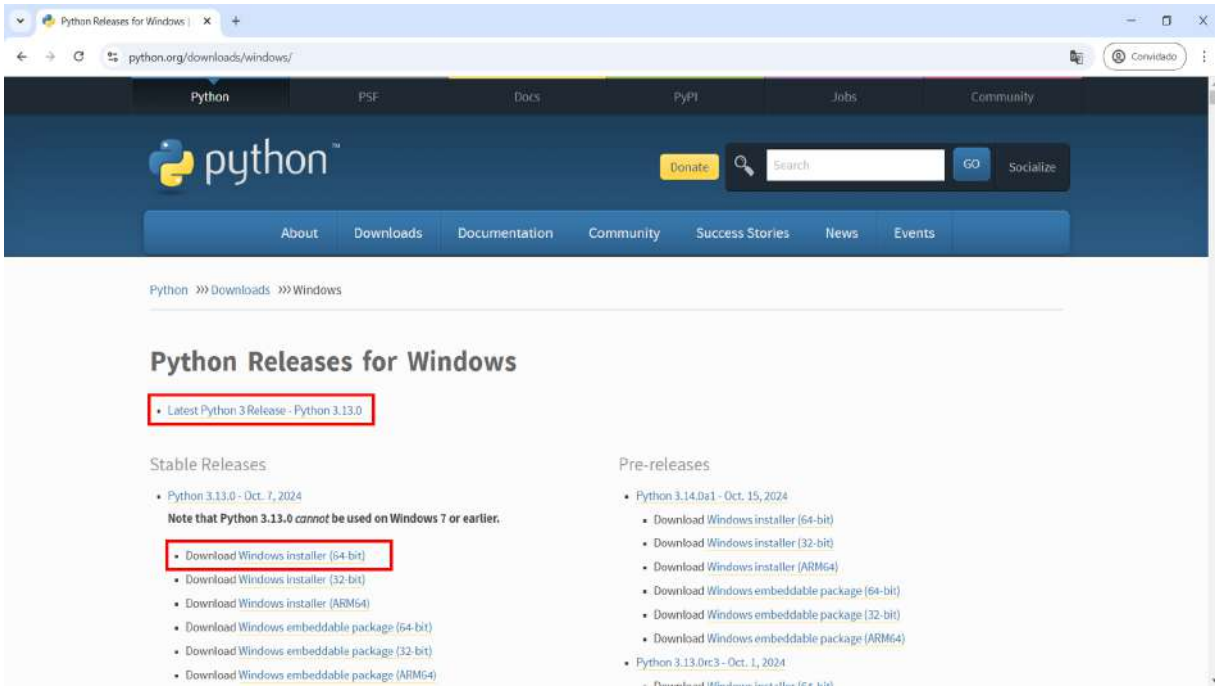


Fonte: Autoria própria

- 3º passo: Escolher a versão que deseja instalar.

Escolhido o sistema operacional, abrirá uma tela com todas as versões do programa disponíveis para download (figura 3.9) . Nessa página, deve-se escolher a versão compatível com a máquina em que se está trabalhando. Aconselha-se utilizar sempre a versão mais recente compatível. Aqui utilizou-se a versão mais recente (3.13.0) para um computador com a versão de 64 bits do Windows.

Figura 3.9 – Versões do Python.



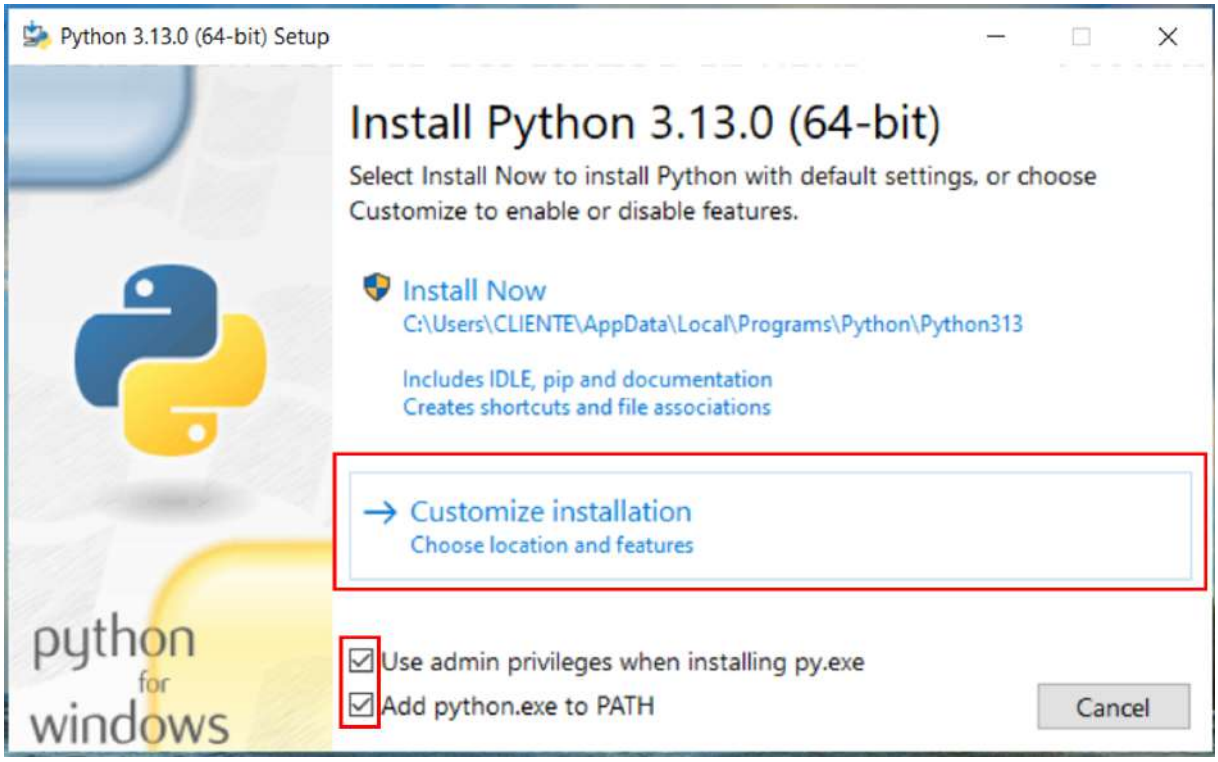
Fonte: Autoria própria

Ao clicar na versão escolhida, o download começará automaticamente. Após a conclusão do download, o arquivo está pronto para ser instalado.

- 4º passo: Instalação.

O passo seguinte é instalar o programa no computador. Para isso, é necessário localizar o arquivo na pasta de downloads e clicar duas vezes com o botão esquerdo do mouse. Ao fazer isso, surgirá uma tela com duas possibilidades de instalação (figura 3.10). É importante certificar-se de que as caixas de seleção na parte inferior da tela estão marcadas. A primeira opção inicia a instalação imediatamente e a segunda permite customizar a instalação. Sugere-se optar pela segunda opção.

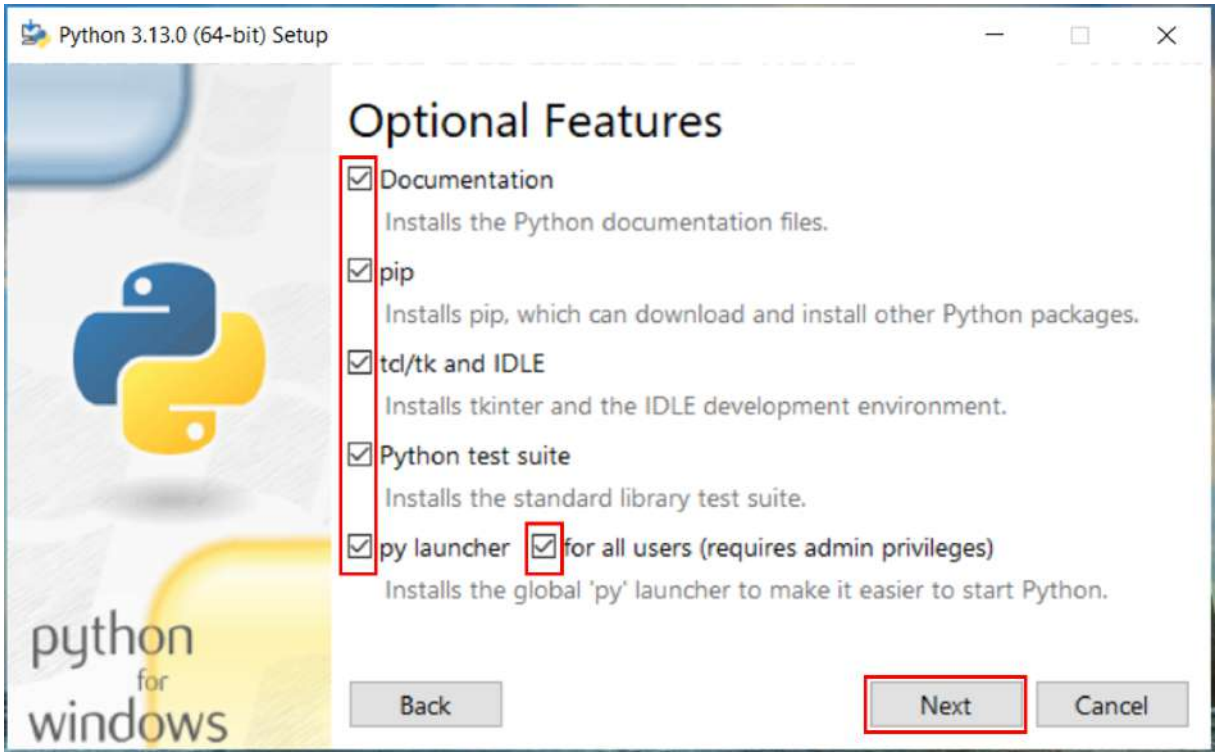
Figura 3.10 – Opçoes de instalação.



Fonte: Autoria própria

Feita a escolha de customizar a instalação, surgirá mais uma tela (figura 3.11). Nessa tela, deve-se conferir se todas as caixas de seleção estão marcadas. Feito isso, clica-se em “next” para dar prosseguimento à instalação.

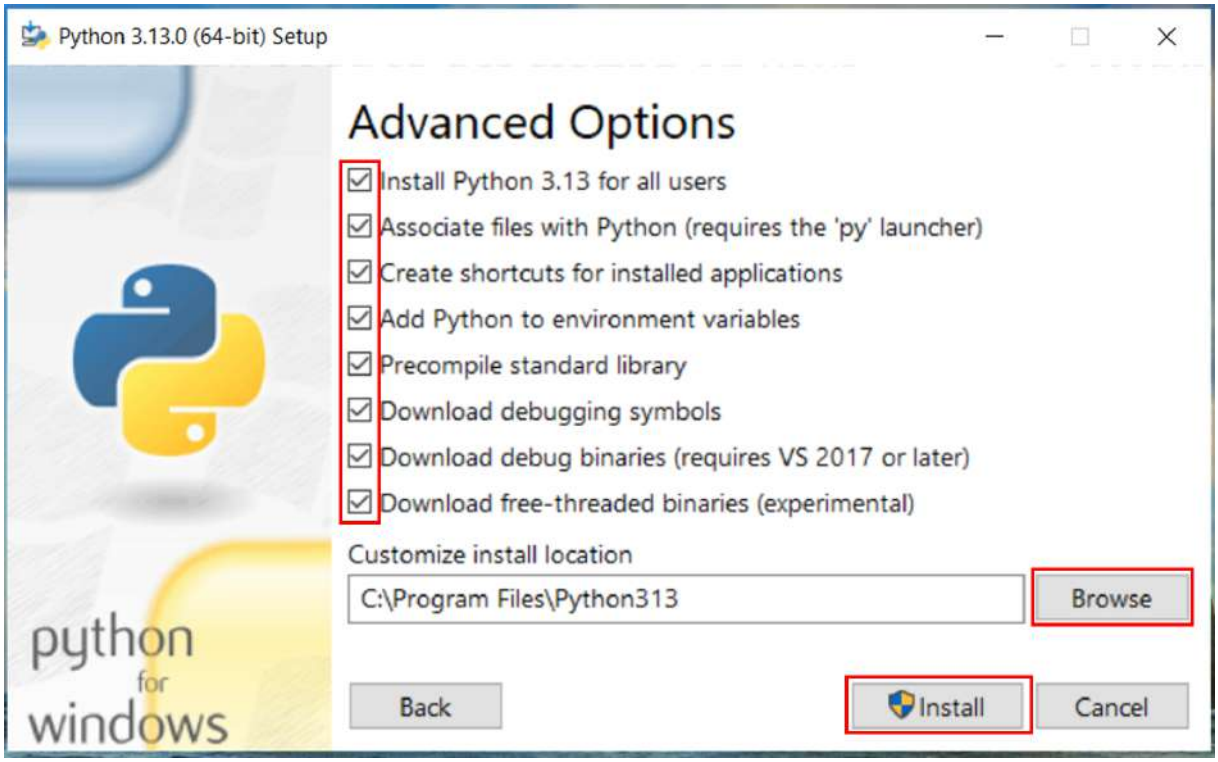
Figura 3.11 – Recursos opcionais.



Fonte: Autoria própria

A última tela conclui o processo de instalação. Mais uma vez, é necessário selecionar todas as caixas, como mostrado na imagem abaixo (figura 3.12). Nessa tela, ainda é possível escolher onde o programa será instalado clicando em "browse". Ao clicar em "Install" dá-se por finalizada a instalação do Python.

Figura 3.12 – Opções avançadas.



Fonte: Autoria própria

Feito todo esse processo, é necessário aguardar a barra de carregamento finalizar e clicar em "concluir".

3.1.2.1.2 **Simulações**

Antecedendo a visualização prática do funcionamento do aparato experimental, foram desenvolvidos quatro códigos em Python para simular diferentes sistemas oscilatórios: **o pêndulo simples**, **o pêndulo simples amortecido**, **o pêndulo simples forçado** e **dois pêndulos simples acoplados por uma mola**, sendo este último um modelo semelhante ao sistema físico desenvolvido. Cada uma dessas simulações computacionais foi projetada para reproduzir o comportamento esperado dos sistemas reais, possibilitando a análise detalhada de suas dinâmicas antes da montagem física. Essa abordagem permitiu uma compreensão prévia das variáveis envolvidas, bem como a identificação de possíveis desafios experimentais, tornando o processo de construção e calibração do aparato mais eficiente.

A importância dessa etapa inicial reside no fato de que a simulação computacional proporciona um ambiente controlado e seguro para a exploração de conceitos fundamentais, como oscilações, amortecimento, ressonância e acoplamento de sistemas. Além disso, a manipulação de parâmetros nos códigos oferece percepções sobre o impacto de fatores como massa, comprimento e coeficientes de amortecimento na resposta dinâmica dos sistemas. Esse processo não apenas fortalece a base teórica dos experimentos, mas também garante uma preparação mais sólida para a aplicação prática, alinhando o desenvolvimento

experimental com objetivos didáticos e de pesquisa.

Nesta subseção, são detalhados os códigos desenvolvidos, com uma descrição de suas funcionalidades e dos objetivos específicos de cada um, destacando suas contribuições para a compreensão e análise dos sistemas oscilatórios simulados.

I - SIMULAÇÃO COMPUTACIONAL DO PÊNDULO SIMPLES

Introdução

Este documento apresenta o desenvolvimento de um aplicativo em Python para simular numericamente a dinâmica de um pêndulo simples. Utilizando o método de Runge-Kutta de quarta ordem (RK-4), o programa permite calcular a evolução temporal do sistema, bem como visualizar graficamente aspectos relevantes, como a trajetória angular, a energia cinética, a energia potencial e o vetor velocidade do bob. Além disso, o aplicativo inclui controles interativos que permitem ajustar parâmetros físicos do sistema em tempo real.

O aplicativo foi concebido como uma ferramenta educativa, oferecendo uma interface gráfica intuitiva que possibilita explorar os efeitos de diferentes condições iniciais e parâmetros físicos. Entre as funcionalidades, destacam-se a animação do movimento do pêndulo, a exibição de gráficos de energia e trajetória, e a manipulação interativa de variáveis como massa e comprimento do fio.

Descrição do Código

O código inicializa com a definição das constantes físicas do sistema, como a aceleração da gravidade $g = 9,81\,\mathrm{m/s^2}$, e os parâmetros ajustáveis: massa do bob m, comprimento do fio L e deslocamento angular inicial $\theta_0 = 60°$. O intervalo de tempo para a integração numérica é definido como $\Delta t = 0,01\,\mathrm{s}$, garantindo alta precisão na simulação.

A equação diferencial que descreve o sistema é dada por:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0,$$

e é resolvida numericamente pelo método RK-4. A função `derivs` calcula as derivadas de estado θ e $\dot{\theta}$, enquanto a função `run_simulation` executa a integração numérica para o tempo total definido pelo usuário. Os resultados, θ (deslocamento angular) e $\dot{\theta}$ (velocidade angular), são armazenados para posterior visualização.

Visualização Gráfica e Animação

A interface gráfica é implementada utilizando a biblioteca `matplotlib` e oferece duas áreas principais:

- **Animação do Pêndulo:** Mostra a evolução do movimento em tempo real. O bob (massa pendular) é representado por um círculo vermelho, conectado por uma haste ao ponto de suspensão. Um vetor azul indica a velocidade tangencial do bob.
- **Gráfico da Posição Angular:** Apresenta a evolução da posição angular do bob em função do tempo, permitindo analisar o comportamento oscilatório do sistema.

Além disso, o aplicativo inclui barras verticais que exibem a energia cinética e a energia potencial do sistema, normalizadas em relação à energia total.

Controles Interativos

O aplicativo incorpora funcionalidades interativas para personalização da simulação:

- **Sliders:** Permitem ajustar os valores de massa m e comprimento do fio L durante a execução, alterando dinamicamente a simulação.
- **Botões:** Controles para parar (`stop_animation`) e reiniciar (`restart_animation`) a animação.

A função `update` é acionada ao modificar os sliders, recalculando as trajetórias com os novos valores e atualizando as visualizações em tempo real.

Conclusão

Este aplicativo é uma ferramenta robusta para o ensino de física clássica, especialmente no estudo do movimento oscilatório. Ele combina a precisão de métodos numéricos com uma interface interativa, proporcionando uma experiência de aprendizado visual e prática. O código pode ser expandido para incluir outros fenômenos, como amortecimento ou forças externas, ampliando seu potencial educacional e de pesquisa.

As figuras apresentadas a seguir ilustram o código desenvolvido.

Figura 3.13 – Código da simulação de um pêndulo simples - página 1 de 4

um-pendulo-whole.py - C:\Users\ADM\Desktop\CODES\SINGLE\um-pendulo-whole.py (3.11.4)

File Edit Format Run Options Window Help

```
# APLICATIVO PARA INTEGRAÇÃO NUMÉRICA DA DINÂMICA DE UM PÊNDULO SIMPLES
# CODIGO COMPLETO
# MÉTODO RK-4
# GRÁFICO DA TRAJETÓRIA EM FUNÇÃO DO TEMPO
# AJUSTE INTERATIVO DOS PARÂMETROS DE SIMULAÇÃO
# ANIMAÇÃO DA EVOLUÇÃO TEMPORAL DO PÊNDULO
# VISUALIZAÇÃO DO VETOR VELOCIDADE
# GRÁFICO DA EVOLUÇÃO DAS ENERGIAS POTENCIAL E CINÉTICA
# N. M. SOTOMAYOR; RAFAEL MEDEIROS DE FREITAS; L. Y. A. DAVILA V. MAIO 2024.
# EQUAÇÃO DO PÊNDULO
#\frac{d^2\theta}{dt^2} + \frac{g}{L} \sin\theta = 0

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
import matplotlib.animation as animation

# Constantes iniciais
initial_m = 1
initial_L = 1
g = 9.81  # aceleração devido à gravidade (m/s^2)

# Deslocamento angular inicial (rad) e velocidade tangencial (m/s)
theta0, v0 = np.radians(60), 0

# Tempo de passo para integração numérica da equação de movimento (s).
dt = 0.01

# Inicializa as posições angulares e as velocidades tangenciais.
theta, v = [theta0], [v0]

# Número de periodos desejados
num_periods = 16  # Ajuste conforme necessário

# Tempo total de simulação
total_simulation_time = num_periods * (2 * np.pi * np.sqrt(initial_L / g))

def derivs(state, m, L):
    theta, omega = state
    dtheta_dt = omega
    domega_dt = -(g / L) * np.sin(theta)
    return np.array([dtheta_dt, domega_dt])

def run_simulation(m, L):
    theta, v = [theta0], [v0]
    state = np.array([theta0, v0])

    t = 0
    while t < total_simulation_time:
        k1 = derivs(state, m, L)
        k2 = derivs(state + 0.5 * dt * k1, m, L)
        k3 = derivs(state + 0.5 * dt * k2, m, L)
        k4 = derivs(state + dt * k3, m, L)
        state += dt / 6.0 * (k1 + 2 * k2 + 2 * k3 + k4)
        theta.append(state[0])
        v.append(state[1])
        t += dt
    return theta, v

theta, v = run_simulation(initial_m, initial_L)    #PAGINA 1
```

Fonte: Autoria própria

Figura 3.14 – Código da simulação de um pêndulo simples - página 2 de 4

um-pendulo-whole.py - C:\Users\ADM\Desktop\CODES\SINGLE\um-pendulo-whole.py (3.11.4)

File Edit Format Run Options Window Help

```
theta, v = run_simulation(initial_m, initial_L)     #PAGINA 1

def get_coords(th, L):
    return L * np.sin(th), -L * np.cos(th)

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 5))

# Ajustar a posição dos eixos da animação e do gráfico
ax1.set_position([0.27, 0.33, 0.6, 0.65])
ax2.set_position([0.27, 0.05, 0.6, 0.2])

x0, y0 = get_coords(theta0, initial_L)
line1, = ax1.plot([0, x0], [0, y0], lw=3, c='k')
bob_radius = 0.12
circle1 = ax1.add_patch(plt.Circle(get_coords(theta0, initial_L), bob_radius, fc='r', zorder=4))

# Adicionando as linhas dos eixos x = 0 e y = 0
ax1.axhline(0, color='green')
ax1.axvline(0, color='green')

# Adicionando as barras verticais
ax_potential_energy = plt.axes([0.82, 0.65, 0.05, 0.2], facecolor='lightgoldenrodyellow')
ax_kinetic_energy = plt.axes([0.9, 0.65, 0.05, 0.2], facecolor='lightgoldenrodyellow')
potential_bar = ax_potential_energy.bar(0, 0, width=1, color='blue')
kinetic_bar = ax_kinetic_energy.bar(0, 0, width=1, color='red')

ax_potential_energy.set_xlim(-0.5, 0.5)
ax_kinetic_energy.set_xlim(-0.5, 0.5)

ax_potential_energy.set_ylim(0, 0.5)
ax_kinetic_energy.set_ylim(0, 0.5)

ax_potential_energy.set_xticks([])
ax_kinetic_energy.set_xticks([])

ax_potential_energy.set_yticks([])
ax_kinetic_energy.set_yticks([])

ax_potential_energy.set_title('Energia Potencial')
ax_kinetic_energy.set_title('Energia Cinética')

# Configurar a razão de aspecto para ser igual
ax1.set_aspect('equal')

ax1.set_xlim(-initial_L * 1.2, initial_L * 1.2)
ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)

# Inicializando o vetor velocidade
quiver = ax1.quiver(0, 0, 0, 0, angles='xy', scale_units='xy', scale=1, color='blue', zorder=5)

bob_positions = []

# Fator de escala para ajustar o comprimento do vetor de velocidade
scale_factor = 0.2

def animate(i):
    x, y = get_coords(theta[i], initial_L)
    line1.set_data([0, x], [0, y])
    circle1.set_center((x, y))                    # PAGINA 2
```

Fonte: Autoria própria

Figura 3.15 – Código da simulação de um pêndulo simples - página 3 de 4

um-pendulo-whole.py - C:\Users\ADM\Desktop\CODES\SINGLE\um-pendulo-whole.py (3.11.4)

File Edit Format Run Options Window Help

```
    circle1.set_center((x, y))                    # PAGINA 2

    # Calcular a velocidade do bob
    vx = scale_factor * initial_L * v[i] * np.cos(theta[i])
    vy = scale_factor * initial_L * v[i] * np.sin(theta[i])

    # Atualizar o vetor de velocidade como uma seta na frente do bob
    quiver.set_offsets([x, y])
    quiver.set_UVC(vx, vy)

    bob_positions.append(x)

    # Calculando as energias potencial e cinética
    kinetic_energy = 0.5 * initial_m * (initial_L * v[i])**2
    potential_energy = initial_m * g * (initial_L * (1 - np.cos(theta[i])))

    # Normalizando para exibição
    max_energy = initial_m * g * initial_L  # Energia potencial máxima
    normalized_kinetic = kinetic_energy / max_energy
    normalized_potential = potential_energy / max_energy

    potential_bar[0].set_height(normalized_potential)
    kinetic_bar[0].set_height(normalized_kinetic)

    plot_position_vs_time(i)

    return line1, circle1, potential_bar, kinetic_bar

nframes = len(theta)
interval = dt * 1000
ani = animation.FuncAnimation(fig, animate, frames=nframes, repeat=True, interval=interval)

def plot_position_vs_time(i):
    ax2.clear()
    time_values = np.arange(0, i) * dt
    ax2.plot(time_values, bob_positions[:i], label='Bob')
    ax2.set_xlabel('Tempo (s)')
    ax2.set_ylabel('Posição')
    ax2.set_title('Evolução da Posição do Bob em Função do Tempo')
    ax2.legend()
    ax2.grid(True)

def stop_animation(event):
    ani.event_source.stop()
    plot_position_vs_time(len(theta))

def restart_animation(event):
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    bob_positions.clear()
    ax2.clear()
    ani.event_source.start()

# Criação dos botões interativos
stop_button_ax = plt.axes([0.01, 0.3, 0.2, 0.04])
stop_button = Button(stop_button_ax, 'Parar')
stop_button.on_clicked(stop_animation)

restart_button_ax = plt.axes([0.01, 0.35, 0.2, 0.04])
restart_button = Button(restart_button_ax, 'Reiniciar')
restart_button.on_clicked(restart_animation)            # PAGINA 3
```

Fonte: Autoria própria

Figura 3.16 – Código da simulação de um pêndulo simples - página 4 de 4

um-pendulo-whole.py - C:\Users\ADM\Desktop\CODES\SINGLE\um-pendulo-whole.py (3.11.4)

File Edit Format Run Options Window Help

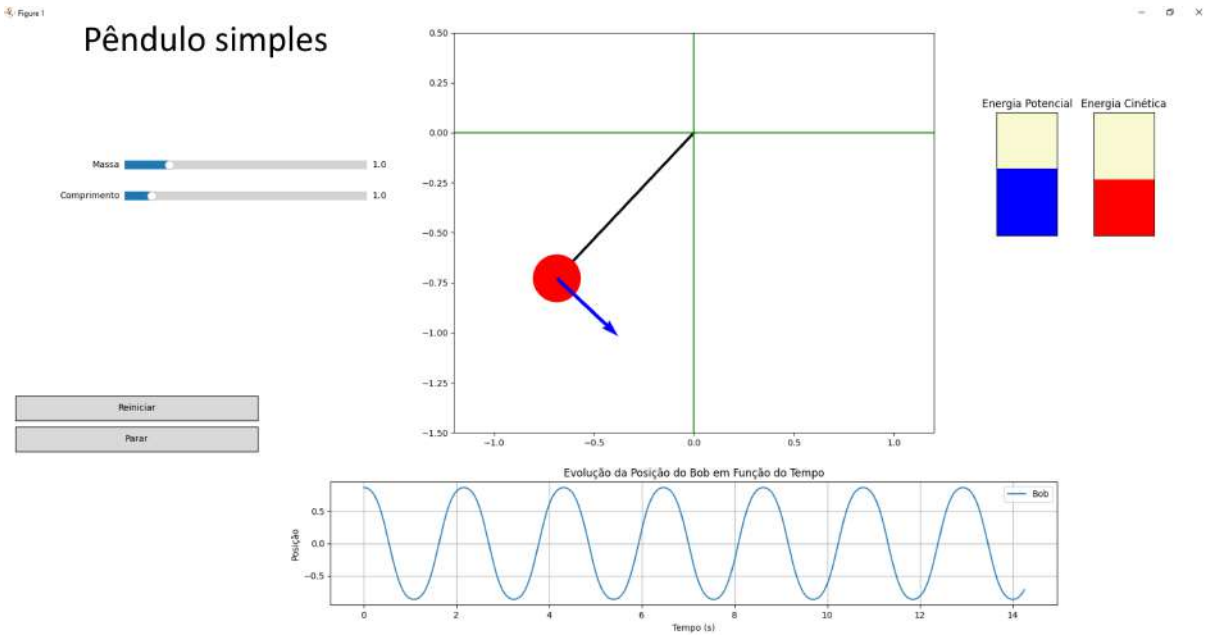
```
    plot_position_vs_time(i)

    return line1, circle1, potential_bar, kinetic_bar

nframes = len(theta)
interval = dt * 1000
ani = animation.FuncAnimation(fig, animate, frames=nframes, repeat=True, interval=interval)

def plot_position_vs_time(i):
    ax2.clear()
    time_values = np.arange(0, i) * dt
    ax2.plot(time_values, bob_positions[:i], label='Bob')
    ax2.set_xlabel('Tempo (s)')
    ax2.set_ylabel('Posição')
    ax2.set_title('Evolução da Posição do Bob em Função do Tempo')
    ax2.legend()
    ax2.grid(True)

def stop_animation(event):
    ani.event_source.stop()
    plot_position_vs_time(len(theta))

def restart_animation(event):
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    bob_positions.clear()
    ax2.clear()
    ani.event_source.start()

# Criação dos botões interativos
stop_button_ax = plt.axes([0.01, 0.3, 0.2, 0.04])
stop_button = Button(stop_button_ax, 'Parar')
stop_button.on_clicked(stop_animation)

restart_button_ax = plt.axes([0.01, 0.35, 0.2, 0.04])
restart_button = Button(restart_button_ax, 'Reiniciar')
restart_button.on_clicked(restart_animation)            # PAGINA 3

# Sliders para ajustar massa e comprimento do pêndulo
ax_mass = plt.axes([0.1, 0.75, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_length = plt.axes([0.1, 0.7, 0.2, 0.03], facecolor='lightgoldenrodyellow')

slider_mass = Slider(ax_mass, 'Massa', 0.1, 5.0, valinit=initial_m)
slider_length = Slider(ax_length, 'Comprimento', 0.5, 5.0, valinit=initial_L)

def update(val):
    global theta, v, initial_m, initial_L
    initial_m = slider_mass.val
    initial_L = slider_length.val
    theta, v = run_simulation(initial_m, initial_L)
    ax1.set_xlim(-initial_L * 1.2, initial_L * 1.2)
    ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    ani.event_source.start()

slider_mass.on_changed(update)
slider_length.on_changed(update)

plt.show()
                                                    # PAGINA 4
```

Fonte: Autoria própria

A interface do aplicativo é apresentada a seguir: uma janela interativa para

simulação do movimento de um pêndulo simples, com gráficos e controles ajustáveis. Na área superior esquerda, a animação exibe o movimento do pêndulo, com um ponto de massa representado por um círculo vermelho e um vetor azul indicando a velocidade do bob. Próximo à animação, barras verticais ilustram a energia potencial (azul) e cinética (vermelha), variando conforme o movimento. Abaixo, um gráfico de posição versus tempo registra a trajetória do bob ao longo da simulação. No lado esquerdo, controles interativos, como sliders para ajustar a massa e o comprimento do pêndulo, e botões para parar e reiniciar a animação, permitem ao usuário modificar os parâmetros e observar o impacto no sistema em tempo real. A interface oferece uma visualização dinâmica e intuitiva dos conceitos de física envolvidos, com atualizações instantâneas baseadas nas interações do usuário.

Figura 3.17 – Interface da simulação de um pêndulo simples



Fonte: Autoria própria

II - SIMULAÇÃO COMPUTACIONAL DO PÊNDULO SIMPLES COM AMORTECIMENTO

Introdução

Este documento apresenta a descrição de um programa interativo desenvolvido para simular a dinâmica de um pêndulo simples submetido a forças gravitacionais e de amortecimento proporcional à velocidade angular. A integração numérica das equações de movimento é realizada utilizando o método de Runge-Kutta de quarta ordem (RK-4). A aplicação inclui animação da trajetória do pêndulo, gráficos interativos da posição, velocidade e energia, e controle em tempo real dos parâmetros físicos.

Descrição do Código

As constantes iniciais definidas incluem a aceleração da gravidade $g = 9,81\,\mathrm{m/s^2}$, a massa do pêndulo m, o comprimento do fio L e o coeficiente de amortecimento γ. O deslocamento angular inicial θ_0 é configurado em radianos e a velocidade tangencial inicial v_0 em metros por segundo. O programa considera um intervalo de tempo discreto $\Delta t = 0,01\,\mathrm{s}$ para a integração.

As equações de movimento, que incluem o efeito do amortecimento, são resolvidas iterativamente pela função `run_simulation`. O método calcula as derivadas do estado atual θ e ω usando a função `derivs`.

Visualização Gráfica

O programa utiliza o `matplotlib` para gerar a animação e os gráficos do sistema. A animação exibe o pêndulo como uma linha conectada a uma massa (bob) representada por um círculo. Gráficos adicionais mostram a evolução da posição, velocidade e energias cinética e potencial ao longo do tempo.

Uma barra de energia na interface ilustra a relação entre energia potencial e cinética, normalizada em relação à energia total. As posições do pêndulo são convertidas em coordenadas cartesianas para visualização.

Interatividade

O programa incorpora elementos interativos que permitem controlar a simulação em tempo real. Os botões permitem pausar ou reiniciar a animação. Sliders ajustam os valores de m, L e γ, com atualizações instantâneas na dinâmica do sistema. A função `update` garante que as alterações nos sliders recalculam as trajetórias e gráficos correspondentes.

Conclusão

Esta aplicação é uma ferramenta educacional que demonstra a interação entre forças gravitacionais e de amortecimento em um sistema físico simples. Sua interface interativa permite explorar os efeitos de diferentes parâmetros em tempo real, tornando o aprendizado de conceitos de física clássica mais dinâmico e acessível. O programa pode ser ampliado para incluir novos efeitos, como forças externas ou não-linearidades adicionais.

As figuras abaixo exibem o código desenvolvido para realizar esta simulação.

Figura 3.18 – Código da simulação de um pêndulo simples amortecido - página 1 de 3

simples-amortecido.py - C:\Users\ADM\Desktop\CODES\SIMPLES-AMORTECIDO\simples-amortecido.py (3.11.4)

File Edit Format Run Options Window Help

```
#Este programa implementa uma simulação interativa da dinâmica de um pêndulo simples,
#utilizando o Método Runge-Kutta de 4ª ordem (RK-4) para integração numérica das equações
#de movimento. A força de amortecimento considerada é proporcional à velocidade angular
#do pêndulo, com o coeficiente de amortecimento representado por gamma. A aplicação
#inclui visualização gráfica, ajuste de parâmetros em tempo real e animação da trajetória
#do pêndulo, além de gráficos que mostram a evolução da posição, velocidade e energia
#durante a simulação.
#versão dezembro 2024. N. M. SOTOMAYOR; RAFAEL MEDEIROS DE FREITAS; L. Y. A. DAVILA
#$\dfrac{d^2\theta}{dt^2} = -\dfrac{g}{L} \sin(\theta) - \gamma \frac{d\theta}{dt}$


import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
import matplotlib.animation as animation

# Constantes iniciais
initial_m = 1
initial_L = 1
g = 9.81  # aceleração devido à gravidade (m/s^2)
initial_damping = 0.1  # Força de amortecimento inicial

# Deslocamento angular inicial (rad) e velocidade tangencial (m/s)
theta0, v0 = np.radians(60), 0

dt = 0.01

theta, v = [theta0], [v0]
num_periods = 16
total_simulation_time = num_periods * (2 * np.pi * np.sqrt(initial_L / g))

def derivs(state, m, L, damping):
    theta, omega = state
    dtheta_dt = omega
    domega_dt = -(g / L) * np.sin(theta) - damping * omega
    return np.array([dtheta_dt, domega_dt])

def run_simulation(m, L, damping):
    theta, v = [theta0], [v0]
    state = np.array([theta0, v0])
    t = 0
    while t < total_simulation_time:
        k1 = derivs(state, m, L, damping)
        k2 = derivs(state + 0.5 * dt * k1, m, L, damping)
        k3 = derivs(state + 0.5 * dt * k2, m, L, damping)
        k4 = derivs(state + dt * k3, m, L, damping)
        state += dt / 6.0 * (k1 + 2 * k2 + 2 * k3 + k4)
        theta.append(state[0])
        v.append(state[1])
        t += dt
    return theta, v

theta, v = run_simulation(initial_m, initial_L, initial_damping)

def get_coords(th, L):
    return L * np.sin(th), -L * np.cos(th)

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 5))
ax1.set_position([0.27, 0.33, 0.6, 0.65])
ax2.set_position([0.27, 0.05, 0.6, 0.2])                # PAGINA 1
```

Fonte: Autoria própria

Figura 3.19 – Código da simulação de um pêndulo simples amortecido - página 2 de 3

simples-amortecido.py - C:\Users\ADM\Desktop\CODES\SIMPLES-AMORTECIDO\simples-amortecido.py (3.11.4)

File Edit Format Run Options Window Help

```
ax2.set_position([0.27, 0.05, 0.6, 0.2])                # PAGINA 1

x0, y0 = get_coords(theta0, initial_L)
line1, = ax1.plot([0, x0], [0, y0], lw=3, c='k')
bob_radius = 0.12
circle1 = ax1.add_patch(plt.Circle(get_coords(theta0, initial_L), bob_radius, fc='r', zorder=4))
ax1.axhline(0, color='green')
ax1.axvline(0, color='green')

ax_potential_energy = plt.axes([0.82, 0.65, 0.05, 0.2], facecolor='lightgoldenrodyellow')
ax_kinetic_energy = plt.axes([0.9, 0.65, 0.05, 0.2], facecolor='lightgoldenrodyellow')
potential_bar = ax_potential_energy.bar(0, 0, width=1, color='blue')
kinetic_bar = ax_kinetic_energy.bar(0, 0, width=1, color='red')

ax_potential_energy.set_xlim(-0.5, 0.5)
ax_kinetic_energy.set_xlim(-0.5, 0.5)

ax_potential_energy.set_ylim(0, 0.5)
ax_kinetic_energy.set_ylim(0, 0.5)
ax_potential_energy.set_xticks([])
ax_kinetic_energy.set_xticks([])
ax_potential_energy.set_yticks([])
ax_kinetic_energy.set_yticks([])
ax_potential_energy.set_title('Energia Potencial')
ax_kinetic_energy.set_title('Energia Cinética')

ax1.set_aspect('equal')
ax1.set_xlim(-initial_L * 1.2, initial_L * 1.2)
ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)
quiver = ax1.quiver(0, 0, 0, 0, angles='xy', scale_units='xy', scale=1, color='blue', zorder=5)
bob_positions = []
scale_factor = 0.2

def animate(i):
    x, y = get_coords(theta[i], initial_L)
    line1.set_data([0, x], [0, y])
    circle1.set_center((x, y))
    vx = scale_factor * initial_L * v[i] * np.cos(theta[i])
    vy = scale_factor * initial_L * v[i] * np.sin(theta[i])
    quiver.set_offsets([x, y])
    quiver.set_UVC(vx, vy)
    bob_positions.append(x)
    kinetic_energy = 0.5 * initial_m * (initial_L * v[i])**2
    potential_energy = initial_m * g * (initial_L * (1 - np.cos(theta[i])))
    max_energy = initial_m * g * initial_L
    normalized_kinetic = kinetic_energy / max_energy
    normalized_potential = potential_energy / max_energy
    potential_bar[0].set_height(normalized_potential)
    kinetic_bar[0].set_height(normalized_kinetic)
    plot_position_vs_time(i)
    return line1, circle1, potential_bar, kinetic_bar

nframes = len(theta)
interval = dt * 1000
ani = animation.FuncAnimation(fig, animate, frames=nframes, repeat=True, interval=interval)

def plot_position_vs_time(i):
    ax2.clear()
    time_values = np.arange(0, i) * dt
    ax2.plot(time_values, bob_positions[:i], label='Bob')
    ax2.set_xlabel('Tempo (s)')                          # PAGINA 2
```

Fonte: Autoria própria

Figura 3.20 – Código da simulação de um pêndulo simples amortecido - página 3 de 3

simples-amortecido.py - C:\Users\ADM\Desktop\CODES\SIMPLES-AMORTECIDO\simples-amortecido.py (3.11.4)

File Edit Format Run Options Window Help

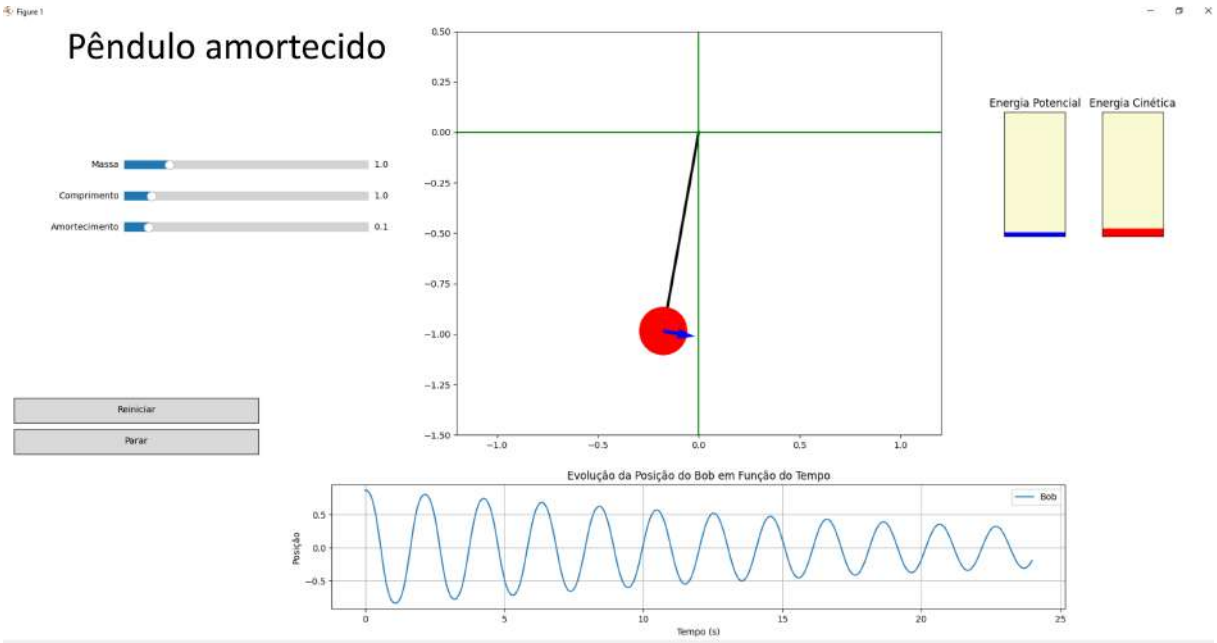
```
    return line1, circle1, potential_bar, kinetic_bar

nframes = len(theta)
interval = dt * 1000
ani = animation.FuncAnimation(fig, animate, frames=nframes, repeat=True, interval=interval)

def plot_position_vs_time(i):
    ax2.clear()
    time_values = np.arange(0, i) * dt
    ax2.plot(time_values, bob_positions[:i], label='Bob')
    ax2.set_xlabel('Tempo (s)')                                  # PAGINA 2
    ax2.set_ylabel('Posição')
    ax2.set_title('Evolução da Posição do Bob em Função do Tempo')
    ax2.legend()
    ax2.grid(True)

def stop_animation(event):
    ani.event_source.stop()
    plot_position_vs_time(len(theta))

def restart_animation(event):
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    bob_positions.clear()
    ax2.clear()
    ani.event_source.start()

stop_button_ax = plt.axes([0.01, 0.3, 0.2, 0.04])
stop_button = Button(stop_button_ax, 'Parar')
stop_button.on_clicked(stop_animation)

restart_button_ax = plt.axes([0.01, 0.35, 0.2, 0.04])
restart_button = Button(restart_button_ax, 'Reiniciar')
restart_button.on_clicked(restart_animation)

ax_mass = plt.axes([0.1, 0.75, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_length = plt.axes([0.1, 0.7, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_damping = plt.axes([0.1, 0.65, 0.2, 0.03], facecolor='lightgoldenrodyellow')

slider_mass = Slider(ax_mass, 'Massa', 0.1, 5.0, valinit=initial_m)
slider_length = Slider(ax_length, 'Comprimento', 0.5, 5.0, valinit=initial_L)
slider_damping = Slider(ax_damping, 'Amortecimento', 0.0, 1.0, valinit=initial_damping)

def update(val):
    global theta, v, initial_m, initial_L, initial_damping
    initial_m = slider_mass.val
    initial_L = slider_length.val
    initial_damping = slider_damping.val
    theta, v = run_simulation(initial_m, initial_L, initial_damping)
    ax1.set_xlim(-initial_L * 1.2, initial_L * 1.2)
    ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    ani.event_source.start()

slider_mass.on_changed(update)
slider_length.on_changed(update)
slider_damping.on_changed(update)

plt.show()
                                                             # PAGINA 3
```

Fonte: Autoria própria

A interface do programa (figura 3.21) apresenta uma simulação interativa da

dinâmica de um pêndulo simples amortecido, com visualização gráfica e animações. Na parte superior, a animação mostra o pêndulo em movimento, com uma linha preta representando o fio e um círculo vermelho representando o ponto de massa (bob), que oscila de acordo com os parâmetros definidos. À direita da animação, há barras gráficas que ilustram a evolução da energia potencial (barra azul) e da energia cinética (barra vermelha), com alturas variando conforme o movimento do pêndulo. À esquerda, controles deslizantes (sliders) permitem ao usuário ajustar a massa do pêndulo, o comprimento do fio e o coeficiente de amortecimento, com as alterações sendo refletidas em tempo real na simulação. Abaixo da animação, um gráfico exibe a evolução da posição do pêndulo ao longo do tempo. Há também botões para parar e reiniciar a animação, proporcionando uma interação dinâmica e flexível com o sistema. A interface oferece uma forma intuitiva de explorar a física do movimento oscilatório e observar os efeitos das variáveis no comportamento do pêndulo.

Figura 3.21 – Interface da simulação de um pêndulo simples amortecido



Fonte: Autoria própria

III - SIMULAÇÃO COMPUTACIONAL DO PÊNDULO SIMPLES FORÇADO

Introdução

Este documento descreve a implementação de um código desenvolvido para simular a dinâmica de um pêndulo simples submetido a uma força periódica externa. A simulação considera fatores como gravidade, amortecimento e movimento angular. O programa utiliza métodos de integração numérica e visualizações interativas para explorar a dinâmica do sistema.

Equação do Movimento

A equação diferencial que governa o pêndulo forçado é dada por:

$$\frac{d^2\theta}{dt^2} + \gamma\frac{d\theta}{dt} + \frac{g}{L}\sin\theta = F_0\cos(\omega t),$$

onde γ é o coeficiente de amortecimento, g é a aceleração gravitacional, L é o comprimento do pêndulo, F_0 é a amplitude da força externa, e ω é a frequência angular da força.

Descrição do Código

O programa define inicialmente as constantes físicas do sistema, como:

- Massa m,
- Comprimento do pêndulo L,
- Aceleração gravitacional $g = 9,81\,\mathrm{m/s^2}$,
- Amplitude da força externa F_0,
- Frequência angular da força ω,
- Coeficiente de amortecimento γ,
- Deslocamento angular inicial θ_0 (em radianos),
- Velocidade inicial v_0 (em m/s).

O intervalo de tempo $\Delta t = 0,01\,\mathrm{s}$ é usado para a integração numérica da equação de movimento utilizando o método de Runge-Kutta de quarta ordem.

Visualização Gráfica

A visualização é realizada com o `matplotlib`, dividida em duas partes principais:

- **Animação do pêndulo**: Mostra o movimento do pêndulo em tempo real. O pêndulo é representado por uma linha conectada a um círculo, com vetores indicando a velocidade.
- **Gráfico de posição**: Exibe a evolução da posição do pêndulo ao longo do tempo.

As barras de energia mostram as energias potencial, cinética e total, normalizadas pela energia máxima.

Interatividade

O programa inclui controles interativos:

- **Botões**: Permitem parar (`stop_animation`) ou reiniciar (`restart_animation`) a animação.
- **Sliders**: Ajustam parâmetros como massa m, comprimento L, amplitude F_0, frequência ω e amortecimento γ. As alterações atualizam a dinâmica em tempo real.

Conclusão

Este programa fornece uma ferramenta educativa para explorar a dinâmica de um pêndulo forçado. A combinação de simulação numérica e visualização gráfica interativa oferece uma experiência envolvente, permitindo aos usuários ajustar parâmetros e observar os efeitos em tempo real. O código pode ser expandido para incluir efeitos dissipativos complexos ou configurações adicionais.

Nas figuras abaixo, é apresentado o código desenvolvido para a simulação.

Figura 3.22 – Código da simulação de um pêndulo simples forçado - página 1 de 4

single-forcado.py - C:\Users\ADM\Desktop\CODES\SINGLE-FORCADO\single-forcado.py (3.11.4)

File Edit Format Run Options Window Help

```
# Este código, Versão Dezembro 2024, realiza a simulação numérica de um pêndulo simples forçado,
# considerando uma força periódica de excitação e outros fatores como gravidade, amortecimento e
# movimento angular. A implementação utiliza métodos de integração numérica e visualizações
# interativas para explorar a dinâmica do pêndulo.
# N. M. SOTOMAYOR; RAFAEL MEDEIROS DE FREITAS; L. Y. A. DAVILA
# EQUAÇÃO DO PÊNDULO:
# $\dfrac{d^2\theta}{dt^2} + \gamma \dfrac{d\theta}{dt} + \dfrac{g}{L} \sin\theta = F_0 \cos(\omega t)$

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
import matplotlib.animation as animation

# Constantes iniciais
initial_m = 1
initial_L = 1
g = 9.81  # aceleração devido à gravidade (m/s^2)

# Parâmetros da força periódica
F0 = 0.8  # Amplitude da força
omega_f = 3.3  # Frequência da força externa
gamma = 0.0  # Coeficiente de amortecimento (opcional)

# Deslocamento angular inicial (rad) e velocidade tangencial (m/s)
theta0, v0 = np.radians(60), 0

# Tempo de passo para integração numérica da equação de movimento (s).
dt = 0.01

# Número de períodos desejados
num_periods = 16  # Ajuste conforme necessário

# Tempo total de simulação
total_simulation_time = num_periods * (2 * np.pi / omega_f)

def derivs(state, t, m, L, F0, omega_f, gamma):
    theta, omega = state

    # Equação diferencial para pêndulo forçado
    # Inclui termos de:
    # 1. Gravidade (restituição)
    # 2. Amortecimento
    # 3. Força periódica externa
    dtheta_dt = omega
    domega_dt = -(g / L) * np.sin(theta) - gamma * omega + (F0 / (m * L)) * np.sin(omega_f * t)

    return np.array([dtheta_dt, domega_dt])

def run_simulation(m, L, F0, omega_f, gamma):
    theta, v = [theta0], [v0]
    state = np.array([theta0, v0])

    t = 0
    time_values = [t]

    while t < total_simulation_time:
        k1 = derivs(state, t, m, L, F0, omega_f, gamma)
        k2 = derivs(state + 0.5 * dt * k1, t + 0.5 * dt, m, L, F0, omega_f, gamma)
        k3 = derivs(state + 0.5 * dt * k2, t + 0.5 * dt, m, L, F0, omega_f, gamma)
        k4 = derivs(state + dt * k3, t + dt, m, L, F0, omega_f, gamma)
                                                                    # PAGINA 01
```

Fonte: Autoria própria

Figura 3.23 – Código da simulação de um pêndulo simples forçado - página 2 de 4

single-forcado.py - C:\Users\ADM\Desktop\CODES\SINGLE-FORCADO\single-forcado.py (3.11.4)

File Edit Format Run Options Window Help

```
                                                                    # PAGINA 01
        state += dt / 6.0 * (k1 + 2 * k2 + 2 * k3 + k4)
        theta.append(state[0])
        v.append(state[1])

        t += dt
        time_values.append(t)

    return theta, v, time_values

theta, v, time_values = run_simulation(initial_m, initial_L, F0, omega_f, gamma)

def get_coords(th, L):
    return L * np.sin(th), -L * np.cos(th)

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 8))

# Posicionamento dos eixos
ax1.set_position([0.27, 0.33, 0.6, 0.65])
ax2.set_position([0.27, 0.05, 0.6, 0.2])

x0, y0 = get_coords(theta0, initial_L)
line1, = ax1.plot([0, x0], [0, y0], lw=3, c='k')
bob_radius = 0.12
circle1 = ax1.add_patch(plt.Circle(get_coords(theta0, initial_L), bob_radius, fc='r', zorder=4))

# Eixos e barras de energia
ax1.axhline(0, color='green')
ax1.axvline(0, color='green')

# Modificação: Barras de energia mais largas
ax_potential_energy = plt.axes([0.85, 0.65, 0.05, 0.2], facecolor='lightgoldenrodyellow')
ax_kinetic_energy = plt.axes([0.93, 0.65, 0.05, 0.2], facecolor='lightgoldenrodyellow')
ax_total_energy = plt.axes([0.78, 0.65, 0.05, 0.2], facecolor='lightgoldenrodyellow')

potential_bar = ax_potential_energy.bar(0, 0, width=1, color='blue')
kinetic_bar = ax_kinetic_energy.bar(0, 0, width=1, color='red')
total_bar = ax_total_energy.bar(0, 0, width=1, color='green')

# Configurações das barras de energia
for ax_energy in [ax_potential_energy, ax_kinetic_energy, ax_total_energy]:
    ax_energy.set_xlim(-0.5, 0.5)
    ax_energy.set_ylim(0, 2)  # Aumentado para mostrar mais energia
    ax_energy.set_xticks([])
    ax_energy.set_yticks([])

ax_potential_energy.set_title('Energia Potencial')
ax_kinetic_energy.set_title('Energia Cinética')
ax_total_energy.set_title('Energia Total')

# Configurações do eixo
ax1.set_aspect('equal')
ax1.set_xlim(-initial_L * 1.2, initial_L * 1.2)
ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)

# Vetor velocidade
quiver = ax1.quiver(0, 0, 0, 0, angles='xy', scale_units='xy', scale=1, color='blue', zorder=5)

bob_positions = []
scale_factor = 0.2
                                                                    # PAGINA 02
```

Fonte: Autoria própria

Figura 3.24 – Código da simulação de um pêndulo simples forçado - página 3 de 4

single-forcado.py - C:\Users\ADM\Desktop\CODES\SINGLE-FORCADO\single-forcado.py (3.11.4)

File Edit Format Run Options Window Help

```
                                                                    # PAGINA 02
def animate(i):
    x, y = get_coords(theta[i], initial_L)
    line1.set_data([0, x], [0, y])
    circle1.set_center((x, y))

    # Vetor velocidade
    vx = scale_factor * initial_L * v[i] * np.cos(theta[i])
    vy = scale_factor * initial_L * v[i] * np.sin(theta[i])
    quiver.set_offsets([x, y])
    quiver.set_UVC(vx, vy)

    bob_positions.append(x)

    # Energias
    kinetic_energy = 0.5 * initial_m * (initial_L * v[i])**2
    potential_energy = initial_m * g * (initial_L * (1 - np.cos(theta[i])))
    total_energy = kinetic_energy + potential_energy

    max_energy = initial_m * g * initial_L
    normalized_kinetic = kinetic_energy / max_energy
    normalized_potential = potential_energy / max_energy
    normalized_total = total_energy / max_energy

    potential_bar[0].set_height(normalized_potential)
    kinetic_bar[0].set_height(normalized_kinetic)
    total_bar[0].set_height(normalized_total)

    plot_position_vs_time(i)

    return line1, circle1, potential_bar, kinetic_bar, total_bar

nframes = len(theta)
interval = dt * 1000
ani = animation.FuncAnimation(fig, animate, frames=nframes, repeat=True, interval=interval)

def plot_position_vs_time(i):
    ax2.clear()
    ax2.plot(time_values[:i], bob_positions[:i], label='Bob')
    ax2.set_xlabel('Tempo (s)')
    ax2.set_ylabel('Posição')
    ax2.set_title('Evolução da Posição do Bob')
    ax2.legend()
    ax2.grid(True)

def stop_animation(event):
    ani.event_source.stop()
    plot_position_vs_time(len(theta))

def restart_animation(event):
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    bob_positions.clear()
    ax2.clear()
    ani.event_source.start()

stop_button_ax = plt.axes([0.01, 0.3, 0.2, 0.04])
stop_button = Button(stop_button_ax, 'Parar')
stop_button.on_clicked(stop_animation)

restart_button_ax = plt.axes([0.01, 0.35, 0.2, 0.04])                    # PAGINA 03
```

Fonte: Autoria própria

Figura 3.25 – Código da simulação de um pêndulo simples forçado - página 4 de 4

single-forcado.py - C:\Users\ADM\Desktop\CODES\SINGLE-FORCADO\single-forcado.py (3.11.4)

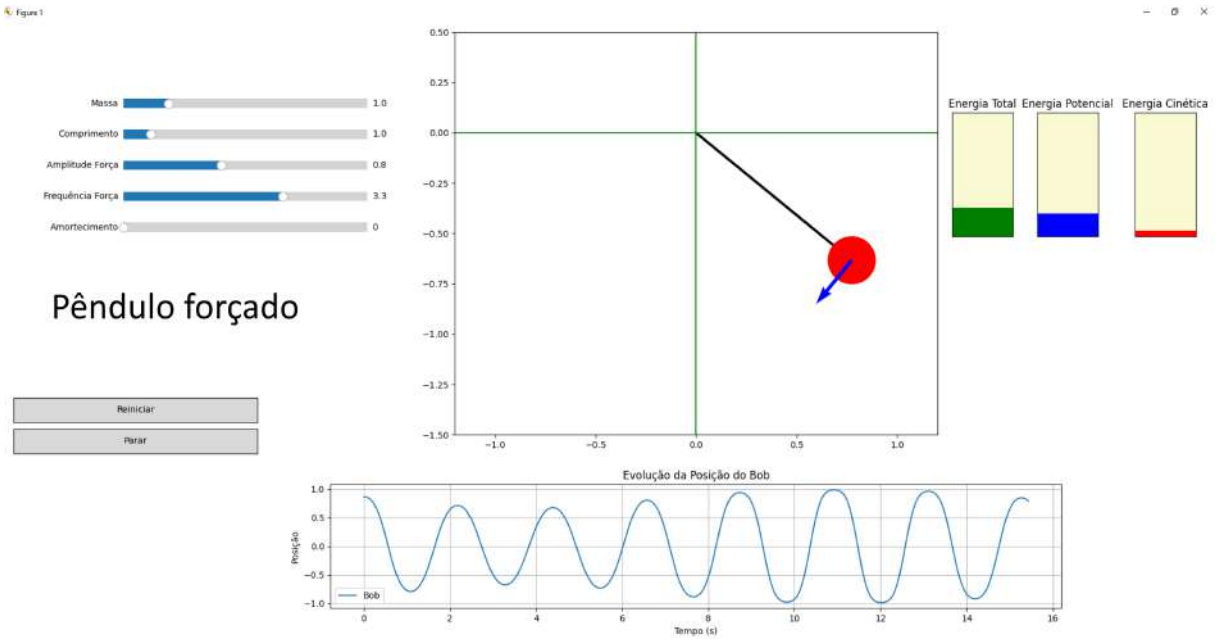
File Edit Format Run Options Window Help

```
    ax2.legend()
    ax2.grid(True)

def stop_animation(event):
    ani.event_source.stop()
    plot_position_vs_time(len(theta))

def restart_animation(event):
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    bob_positions.clear()
    ax2.clear()
    ani.event_source.start()

stop_button_ax = plt.axes([0.01, 0.3, 0.2, 0.04])
stop_button = Button(stop_button_ax, 'Parar')
stop_button.on_clicked(stop_animation)

restart_button_ax = plt.axes([0.01, 0.35, 0.2, 0.04])                    # PAGINA 03
restart_button = Button(restart_button_ax, 'Reiniciar')
restart_button.on_clicked(restart_animation)

# Sliders para parâmetros
ax_mass = plt.axes([0.1, 0.85, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_length = plt.axes([0.1, 0.8, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_force_amp = plt.axes([0.1, 0.75, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_force_freq = plt.axes([0.1, 0.7, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_damping = plt.axes([0.1, 0.65, 0.2, 0.03], facecolor='lightgoldenrodyellow')

slider_mass = Slider(ax_mass, 'Massa', 0.1, 5.0, valinit=initial_m)
slider_length = Slider(ax_length, 'Comprimento', 0.5, 5.0, valinit=initial_L)
slider_force_amp = Slider(ax_force_amp, 'Amplitude Força', 0, 2.0, valinit=F0)
slider_force_freq = Slider(ax_force_freq, 'Frequência Força', 0.1, 5.0, valinit=omega_f)
slider_damping = Slider(ax_damping, 'Amortecimento', 0, 2.0, valinit=gamma)

def update(val):
    global theta, v, time_values, initial_m, initial_L, F0, omega_f, gamma

    initial_m = slider_mass.val
    initial_L = slider_length.val
    F0 = slider_force_amp.val
    omega_f = slider_force_freq.val
    gamma = slider_damping.val

    theta, v, time_values = run_simulation(initial_m, initial_L, F0, omega_f, gamma)

    ax1.set_xlim(-initial_L * 1.2, initial_L * 1.2)
    ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)

    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    ani.event_source.start()

slider_mass.on_changed(update)
slider_length.on_changed(update)
slider_force_amp.on_changed(update)
slider_force_freq.on_changed(update)
slider_damping.on_changed(update)

plt.show()
                                                                         # PAGINA 04
```

Fonte: Autoria própria

A figura 3.26 apresenta a interface da simulação interativa de um pêndulo simples forçado, com a inclusão de barras de energia potencial, cinética e total. A animação exibe o movimento do pêndulo, com o bob (massa suspensa) sendo representado por um círculo vermelho, que oscila de acordo com a força periódica aplicada. O gráfico no eixo inferior mostra a evolução da posição do bob ao longo do tempo, enquanto, no gráfico superior direito, as barras de energia variam conforme o movimento do pêndulo, ilustrando as contribuições da energia cinética e potencial para a energia total. A visualização também permite a interação do usuário por meio de sliders, à esquerda, que ajustam parâmetros como a massa, o comprimento do fio, a amplitude e frequência da força aplicada, além do coeficiente de amortecimento, possibilitando explorar diferentes comportamentos do sistema. O gráfico de barras no canto superior direito exibe as energias de forma normalizada, oferecendo uma representação clara do comportamento dinâmico do pêndulo forçado.

Figura 3.26 – Interface da simulação de um pêndulo simples forçado



Fonte: Autoria própria

IV - SIMULAÇÃO COMPUTACIONAL DE DOIS PÊNDULOS SIMPLES ACOPLADOS POR MOLA ELÁSTICA

Introdução

Este documento apresenta a descrição detalhada de um código desenvolvido para simular a dinâmica de dois pêndulos simples acoplados por uma mola elástica. O objetivo principal do programa é integrar numericamente as equações de movimento utilizando o método de Runge-Kutta de quarta ordem (RK-4). A implementação também inclui uma visualização interativa das trajetórias dos pêndulos e uma animação que ilustra o comportamento do sistema em tempo real.

Descrição do Código

Inicialmente, são definidas as constantes físicas do sistema, como a aceleração da gravidade $g = 9,81\,\mathrm{m/s^2}$, e os parâmetros ajustáveis, incluindo a massa m, o comprimento dos pêndulos L e a constante elástica da mola k. Os deslocamentos angulares iniciais e velocidades tangenciais dos dois pêndulos são definidos em radianos e metros por segundo, respectivamente. O intervalo de tempo $\Delta t = 0,01\,\mathrm{s}$ determina a resolução temporal da integração numérica.

As equações diferenciais que governam o sistema consideram as forças gravitacionais e a força restauradora da mola. Estas equações são resolvidas iterativamente com o método RK-4. A função `derivs` calcula as derivadas das variáveis de estado, enquanto a função `run_simulation` executa a integração para um número de passos correspondente ao tempo total de simulação.

Visualização Gráfica

A visualização do sistema é implementada com o uso do `matplotlib`. Duas janelas gráficas principais são configuradas: a primeira exibe a animação dos pêndulos acoplados e a mola; a segunda mostra a evolução das posições dos bobs em função do tempo. Os pêndulos são representados por linhas conectadas a massas (bobs) desenhadas como círculos, enquanto a mola é modelada como uma linha ondulada. As posições angulares dos pêndulos são convertidas em coordenadas cartesianas para a animação.

A função `animate` atualiza a posição dos pêndulos e da mola a cada quadro da animação. A mola é gerada dinamicamente por meio de uma função que calcula uma curva ondulada conectando os dois bobs.

Interatividade

O programa inclui botões para controlar a execução da animação, como parar (`stop_animation`) ou reiniciar (`restart_animation`) a simulação. Além disso, sliders permitem ajustar os valores de m, L e k em tempo real. A função `update` é acionada sempre que os sliders são alterados, recalculando a dinâmica do sistema com os novos parâmetros.

Conclusão

Este programa é uma ferramenta educativa e interativa que permite explorar a dinâmica de sistemas acoplados de forma intuitiva. A visualização gráfica e os controles interativos tornam a experiência de simulação mais acessível, permitindo ajustar os parâmetros físicos e observar os efeitos em tempo real. Este código exemplifica a aplicação de métodos numéricos em problemas de física clássica e pode ser expandido para incluir efeitos dissipativos ou acoplamentos mais complexos.

As figuras apresentadas a seguir ilustram o código desenvolvido para esta simulação.

Figura 3.27 – Código da simulação de dois pêndulos simples acoplados por mola elástica - página 1 de 4

PENDULO-DUPLO6-gpt4-whole2.py - C:\Users\ADM\Desktop\CODES\DOUBLE\PENDULO-DUPLO6-gpt4-whole2.py (3.11.4)

File Edit Format Run Options Window Help

```
"""
# Código Python para simulação da dinâmica de Pêndulos Simples Acoplados com uma
Mola elástica

Este código implementa um simulador para a integração numérica da dinâmica de dois
pêndulos simples acoplados por uma mola elástica, utilizando o método de Runge-Kutta
de quarta ordem (RK-4). Ele permite visualizar as trajetórias dos dois pêndulos em
função do tempo, bem como ajustar interativamente os parâmetros de simulação.

## Funcionalidades Principais
1. **Integração numérica**:
   - As equações diferenciais que governam o sistema são resolvidas com o método RK-4.
   - As condições iniciais incluem deslocamento angular e velocidade inicial dos pêndulos.

2. **Gráficos interativos**:
   - Trajetórias dos pêndulos (posição angular) em função do tempo.
   - Representação gráfica dos pêndulos, mola e massas em movimento.

3. **Animação do sistema**:
   - Visualização da evolução do sistema em tempo real.
   - Representação gráfica da mola e dos bobs com movimento sincronizado.

4. **Interatividade**:
   - Ajuste de parâmetros como massa dos bobs, constante da mola e comprimento dos
     pêndulos via sliders.
   - Botões para parar e reiniciar a animação.

## Parâmetros Ajustáveis
- Massa dos pêndulos (\(m\)): Entre 0,1 e 5,0 kg.
- Comprimento dos pêndulos (\(L\)): Entre 0,5 e 5,0 metros.
- Constante elástica da mola (\(k\)): Entre 0,001 e 0,1 N/m.

## Estrutura do Código
1. **Constantes iniciais**:
   - Massa, comprimento do pêndulo, constante da mola, gravidade, etc.
2. **Integração numérica**:
   - Cálculo iterativo da dinâmica do sistema com RK-4.
3. **Visualização gráfica**:
   - Representação das posições dos pêndulos, mola e trajetória em tempo real.
4. **Interatividade**:
   - Sliders e botões para controle da simulação.

## Referências
- Este programa foi projetado e implementado com base na metodologia de simulação de
  sistemas dinâmicos.
- Autor: N. M. Sotomayor; L. Y. A. Davila; RAFAEL MEDEIROS DE FREITAS. V. (Maio 2024).
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
import matplotlib.animation as animation

# Constantes iniciais
initial_m = 1
initial_L = 1
initial_k = 0.01
g = 9.81  # aceleração devido à gravidade (m/s^2)

# Deslocamento angular inicial (rad) e velocidade tangencial (m/s)       # PAGINA 01
```

Fonte: Autoria própria

Figura 3.28 – Código da simulação de dois pêndulos simples acoplados por mola elástica - página 2 de 4

PENDULO-DUPLO6-gpt4-whole2.py - C:\Users\ADM\Desktop\CODES\DOUBLE\PENDULO-DUPLO6-gpt4-whole2.py (3.11.4)

File Edit Format Run Options Window Help

```
# Deslocamento angular inicial (rad) e velocidade tangencial (m/s)       # PAGINA 01
theta0, v0 = np.radians(60), 0
theta1, v1 = np.radians(30), 0

# Tempo de passo para integração numérica da equação de movimento (s).
dt = 0.01

# Inicializa as posições angulares e as velocidades tangenciais.
theta, v = [theta0], [v0]
theta2, v2 = [theta1], [v1]

# Número de períodos desejados
num_periods = 16  # Ajuste conforme necessário

# Tempo total de simulação
total_simulation_time = num_periods * (2 * np.pi * np.sqrt(initial_L / g))

def derivs(state, m, L, k):
    theta1, omega1, theta2, omega2 = state
    dtheta1_dt = omega1
    domega1_dt = -(g / L) * np.sin(theta1) - (k / m) * (theta1 - theta2)
    dtheta2_dt = omega2
    domega2_dt = -(g / L) * np.sin(theta2) + (k / m) * (theta1 - theta2)
    return np.array([dtheta1_dt, domega1_dt, dtheta2_dt, domega2_dt])

def run_simulation(m, L, k):
    theta, v = [theta0], [v0]
    theta2, v2 = [theta1], [v1]
    state = np.array([theta0, v0, theta1, v1])

    t = 0
    while t < total_simulation_time:
        k1 = derivs(state, m, L, k)
        k2 = derivs(state + 0.5 * dt * k1, m, L, k)
        k3 = derivs(state + 0.5 * dt * k2, m, L, k)
        k4 = derivs(state + dt * k3, m, L, k)
        state += dt / 6.0 * (k1 + 2 * k2 + 2 * k3 + k4)
        theta.append(state[0])
        v.append(state[1])
        theta2.append(state[2])
        v2.append(state[3])
        t += dt
    return theta, theta2

theta, theta2 = run_simulation(initial_m, initial_L, initial_k)

def get_coords(th, L):
    return L * np.sin(th), -L * np.cos(th)

def get_coords2(th2, L):
    return L * np.sin(th2) + 1, -L * np.cos(th2)

def plot_spring(x1, y1, x2, y2):
    num_coils = 20
    Ns = 1000
    L = np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
    spring_length = L - 2 * bob_radius

    w = np.linspace(0, spring_length, Ns)
    xs = np.linspace(x1, x2, Ns)                                    # PAGINA 02
```

Fonte: Autoria própria

Figura 3.29 – Código da simulação de dois pêndulos simples acoplados por mola elástica - página 3 de 4

PENDULO-DUPLO6-gpt4-whole2.py - C:\Users\ADM\Desktop\CODES\DOUBLE\PENDULO-DUPLO6-gpt4-whole2.py (3.11.4)

File Edit Format Run Options Window Help

```
    xs = np.linspace(x1, x2, Ns)                                    # PAGINA 02
    ys = np.linspace(y1, y2, Ns) + np.sin(w / spring_length * num_coils * 2 * np.pi) * 0.05

    return xs, ys

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 5))

# Ajustar a posição dos eixos da animação e do gráfico
ax1.set_position([0.3, 0.3, 0.6, 0.65])
ax2.set_position([0.3, 0.05, 0.6, 0.2])

x0, y0 = get_coords(theta0, initial_L)
x1, y1 = get_coords2(theta1, initial_L)
line1, = ax1.plot([0, x0], [0, y0], lw=3, c='k')
line2, = ax1.plot([1, x1], [0, y1], lw=3, c='k')
bob_radius = 0.12
bob_radius2 = 0.12
circle1 = ax1.add_patch(plt.Circle(get_coords(theta0, initial_L), bob_radius, fc='r', zorder=3))
circle2 = ax1.add_patch(plt.Circle(get_coords2(theta1, initial_L), bob_radius2, fc='b', zorder=3))

spring_xs, spring_ys = plot_spring(x0, y0, x1, y1)
spring_line, = ax1.plot(spring_xs, spring_ys, c='k', lw=2)

ax1.set_xlim(-initial_L * 1.2, initial_L * 2.2)
ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)

bob1_positions = []
bob2_positions = []

def animate(i):
    x, y = get_coords(theta[i], initial_L)
    line1.set_data([0, x], [0, y])
    circle1.set_center((x, y))
    x2, y2 = get_coords2(theta2[i], initial_L)
    line2.set_data([1, x2], [0, y2])
    circle2.set_center((x2, y2))

    spring_xs, spring_ys = plot_spring(x, y, x2, y2)
    spring_line.set_data(spring_xs, spring_ys)

    bob1_positions.append(x)
    bob2_positions.append(x2)

    plot_position_vs_time(i)

    return line1, line2, circle1, circle2, spring_line

nframes = len(theta)
interval = dt * 1000
ani = animation.FuncAnimation(fig, animate, frames=nframes, repeat=True, interval=interval)

def plot_position_vs_time(i):
    ax2.clear()
    time_values = np.arange(0, i) * dt
    ax2.plot(time_values, bob1_positions[:i], label='Bob 1')
    ax2.plot(time_values, bob2_positions[:i], label='Bob 2')
    ax2.set_xlabel('Tempo (s)')
    ax2.set_ylabel('Posição')
    ax2.set_title('Evolução da Posição dos Bobs em Função do Tempo')
    ax2.legend()
    ax2.grid(True)                                                  # PAGINA 03
```

Fonte: Autoria própria

Figura 3.30 – Código da simulação de dois pêndulos simples acoplados por mola elástica - página 4 de 4

PENDULO-DUPLO6-gpt4-whole2.py - C:\Users\ADM\Desktop\CODES\DOUBLE\PENDULO-DUPLO6-gpt4-whole2.py (3.11.4)

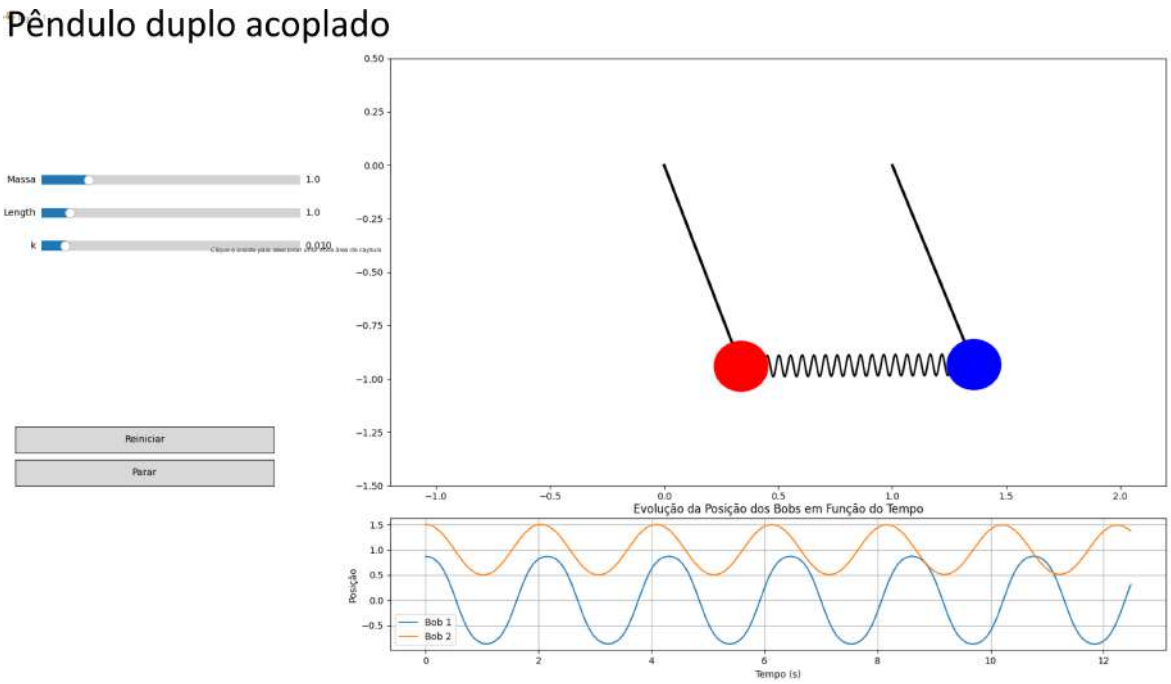
File Edit Format Run Options Window Help

```
ani = animation.FuncAnimation(fig, animate, frames=nframes, repeat=True, interval=interval)

def plot_position_vs_time(i):
    ax2.clear()
    time_values = np.arange(0, i) * dt
    ax2.plot(time_values, bob1_positions[:i], label='Bob 1')
    ax2.plot(time_values, bob2_positions[:i], label='Bob 2')
    ax2.set_xlabel('Tempo (s)')
    ax2.set_ylabel('Posição')
    ax2.set_title('Evolução da Posição dos Bobs em Função do Tempo')
    ax2.legend()
    ax2.grid(True)                                                    # PAGINA 03

def stop_animation(event):
    ani.event_source.stop()
    plot_position_vs_time(len(theta))

def restart_animation(event):
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    bob1_positions.clear()
    bob2_positions.clear()
    ax2.clear()
    ani.event_source.start()

# Criação dos botões interativos
stop_button_ax = plt.axes([0.01, 0.3, 0.2, 0.04])
stop_button = Button(stop_button_ax, 'Parar')
stop_button.on_clicked(stop_animation)

restart_button_ax = plt.axes([0.01, 0.35, 0.2, 0.04])
restart_button = Button(restart_button_ax, 'Reiniciar')
restart_button.on_clicked(restart_animation)

# Sliders para ajustar massa, constante da mola e comprimento do pêndulo
ax_mass = plt.axes([0.03, 0.75, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_length = plt.axes([0.03, 0.7, 0.2, 0.03], facecolor='lightgoldenrodyellow')
ax_k = plt.axes([0.03, 0.65, 0.2, 0.03], facecolor='lightgoldenrodyellow')

slider_mass = Slider(ax_mass, 'Massa', 0.1, 5.0, valinit=initial_m)
slider_length = Slider(ax_length, 'Length', 0.5, 5.0, valinit=initial_L)
slider_k = Slider(ax_k, 'k', 0.001, 0.1, valinit=initial_k)

def update(val):
    global theta, theta2, initial_m, initial_L, initial_k
    initial_m = slider_mass.val
    initial_L = slider_length.val
    initial_k = slider_k.val
    theta, theta2 = run_simulation(initial_m, initial_L, initial_k)
    ax1.set_xlim(-initial_L * 1.2, initial_L * 2.2)
    ax1.set_ylim(-initial_L * 1.5, initial_L * 0.5)
    ani.event_source.stop()
    ani.frame_seq = ani.new_frame_seq()
    ani.event_source.start()

slider_mass.on_changed(update)
slider_length.on_changed(update)
slider_k.on_changed(update)

plt.show()
                                                                      #PAGINA 04
```

Fonte: Autoria própria

A interface visualizada na imagem 3.31 exibe uma simulação interativa de dois pêndulos acoplados, com um gráfico abaixo mostrando a evolução das posições dos bobs ao longo do tempo. Na animação, dois pêndulos, representados por círculos vermelhos e azuis, estão conectados por uma mola, com os fios pendulares se movendo conforme o tempo passa. A mola ajusta-se à movimentação dos pêndulos, criando uma interação dinâmica entre eles. O gráfico abaixo da animação exibe a posição dos bobs em função do tempo, proporcionando uma visualização clara da evolução do sistema. A interface também contém controles interativos, como sliders para ajustar a massa dos bobs, o comprimento do fio e a constante da mola, permitindo ao usuário modificar as condições do sistema e observar as mudanças em tempo real. Além disso, há botões para parar ou reiniciar a animação, o que oferece maior flexibilidade na interação com a simulação.

Figura 3.31 – Interface da simulação de dois pêndulos simples acoplados por mola elástica



Fonte: Autoria própria

Os executáveis dos códigos desenvolvidos estão disponíveis no apêndice A deste trabalho, permitindo a execução das simulações de forma rápida e eficiente, sem a necessidade de ambiente de desenvolvimento ou compilação adicional. Esses arquivos oferecem uma interface acessível para rodar as simulações, tornando-as fáceis de utilizar e de distribuir, garantindo também um desempenho otimizado e a possibilidade de ajustar os parâmetros do sistema conforme necessário.

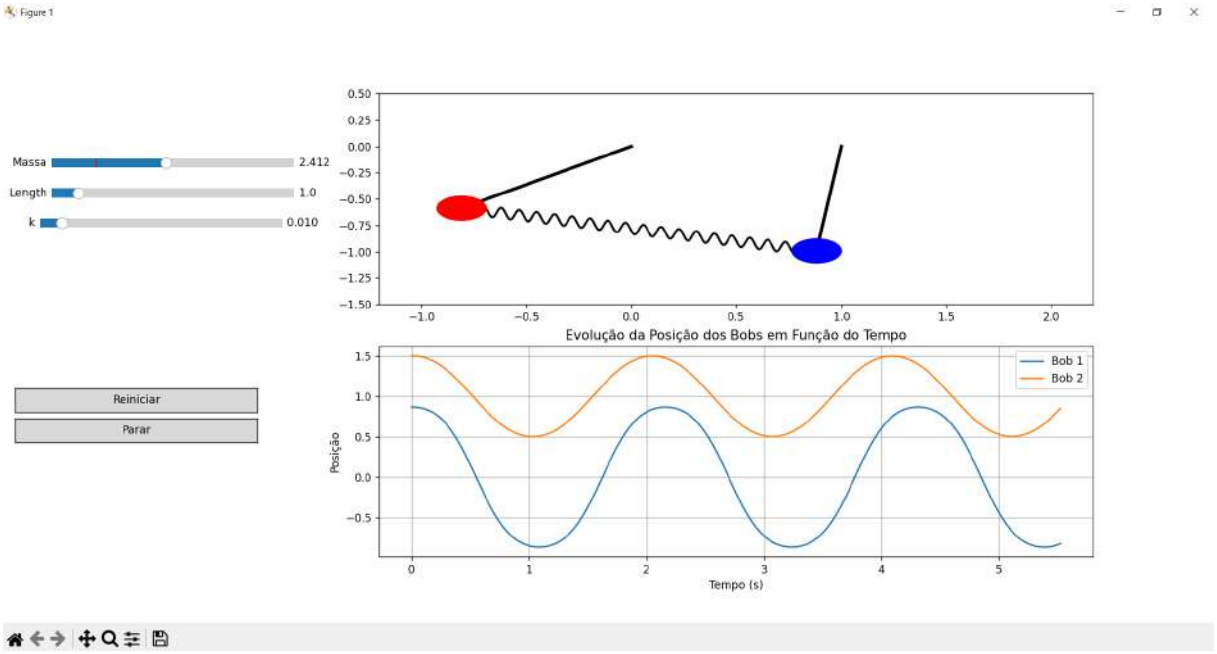
3.1.2.1.3 **Variação dos Parâmetros dos Pêndulos Acoplados**

Os códigos apresentados permitem a análise de diversas situações por meio da modificação dos parâmetros fundamentais do sistema. A seguir, são apresentadas três simulações do sistema de dois pêndulos simples acoplados por uma mola elástica, variando

parâmetros específicos como a massa pendular, o comprimento do fio e a constante elástica da mola. Essas variações têm como objetivo analisar o comportamento do sistema sob diferentes condições e observar como cada parâmetro influencia a dinâmica do movimento.

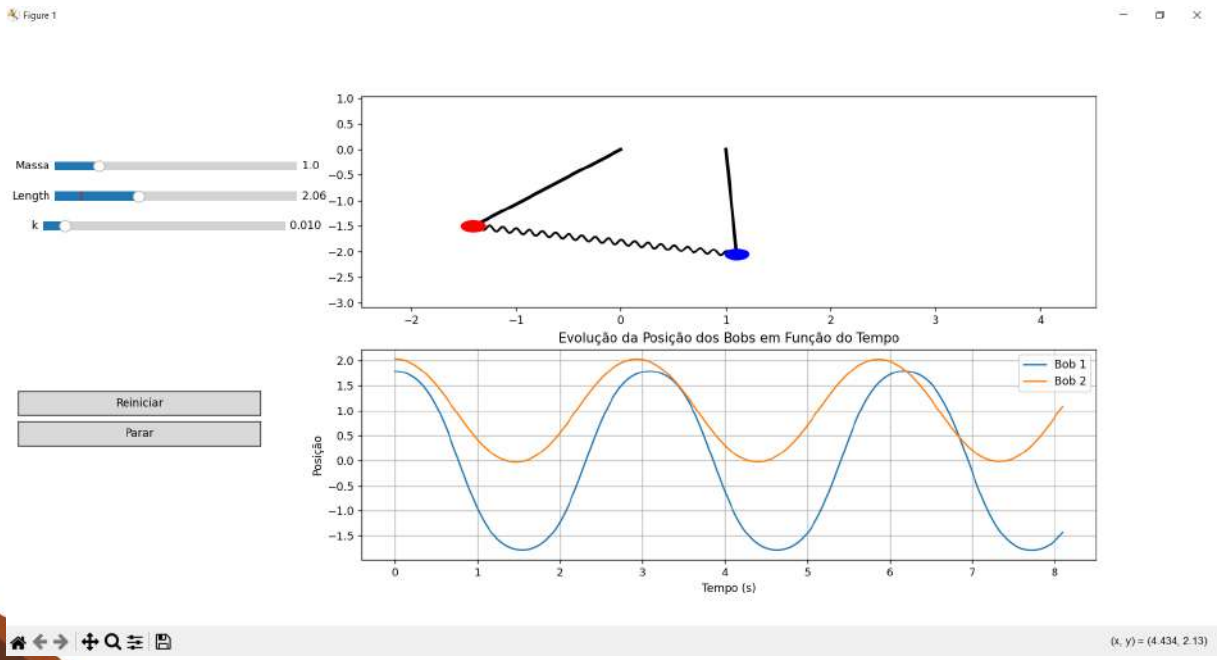
- **Simulação em Python com alteração da massa pendular**

Figura 3.32 – Alterando a massa pendular.



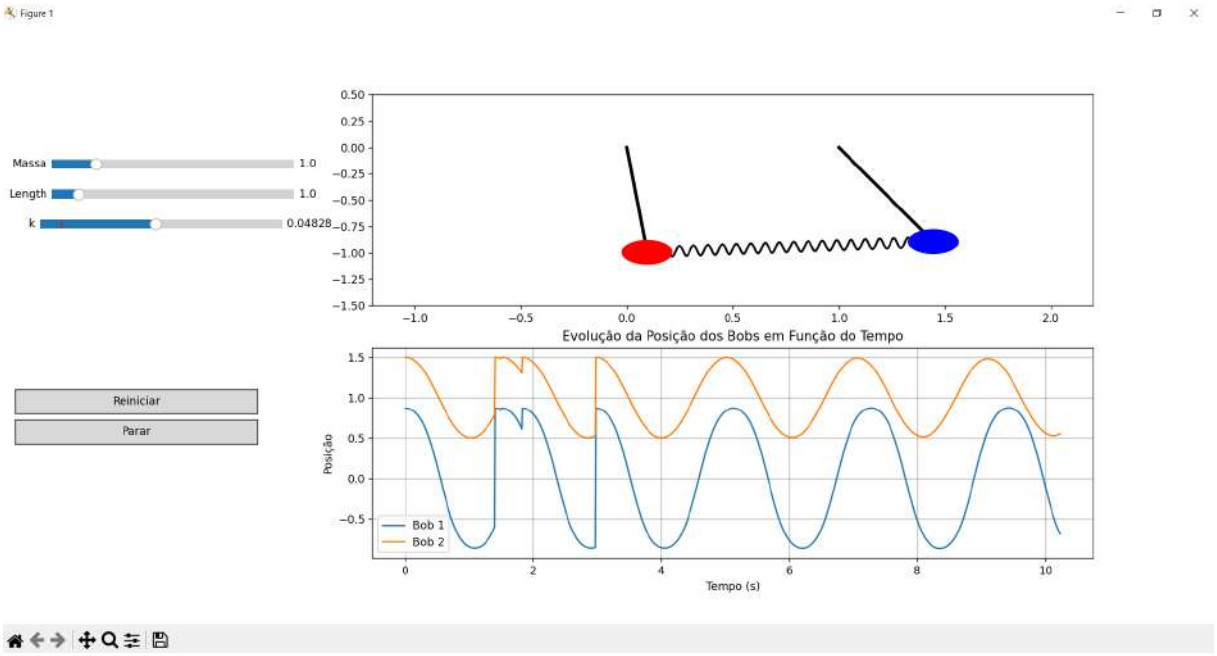
- **Alterando o comprimento dos fios**

Figura 3.33 – Simulação em Python com alteração do comprimento dos fios



- **Alterando a constante elástica da mola**

Figura 3.34 – Simulação em Python com alteração da constante elástica da mola



A interface mostrada na figura 3.31 apresenta o sistema de pêndulos acoplados por uma mola, representando o estado inicial da simulação com os valores padrão para os parâmetros ajustáveis: massa pendular, comprimento do fio e constante elástica da mola. O gráfico exibido na simulação apresenta um padrão básico de oscilação correspondente a esse estado inicial.

Na figura 3.32, visualiza-se que a massa pendular foi aumentada em relação à situação inicial. Como resultado dessa modificação, a frequência das oscilações diminuiu, e o sistema passou a apresentar movimentos mais lentos, com um período maior devido à maior inércia das massas pendulares. No gráfico, observa-se que as ondas se tornam mais espaçadas, indicando o aumento do período de oscilação. Quanto às energias, nota-se uma oscilação mais amortecida, com a troca entre energia cinética e potencial ocorrendo de maneira mais lenta.

Na figura 3.33, o comprimento do fio foi alterado. Essa modificação causou um aumento no período de oscilação, demonstrando a relação diretamente proporcional entre essas grandezas. No gráfico posição x tempo, as ondas aparecem mais espaçadas, refletindo o movimento mais lento. Além disso, é possível observar que, inicialmente, o sistema apresenta movimentos mais independentes, o que se traduz em padrões dessincronizados no início do gráfico.

Por fim, a figura 3.34 ilustra a modificação da constante elástica da mola. Quando aumentada, ela implica um incremento na força restauradora, permitindo uma sincronização mais rápida dos movimentos dos pêndulos. Nos gráficos, isso é evidenciado pelos dois pêndulos oscilando quase em fase. O aumento da constante elástica também resulta em maior energia potencial armazenada na mola, contribuindo para oscilações mais intensas e rápidas, além de uma frequência maior do movimento.

O código e o executável da simulação em Python de dois pêndulos simples acoplados por uma mola também estão disponíveis no apêndice deste trabalho (apêndice A1 e A2).

3.1.2.2 ***Ambiente de Desenvolvimento Integrado (IDE) do Arduíno***

O Ambiente de Desenvolvimento Integrado (IDE) é onde ocorre a escrita do código e as instruções que são enviadas à placa controladora, sendo, dessa forma, indispensáveis para a programação e controle do microcontrolador Arduíno. Trata-se de um software de código aberto onde é permitido ao usuário escrever, compilar e transferir programas, ora chamados de "sketches" para a placa programadora, permitindo o desenvolvimento de projetos interativos e a automação de experimentos.

Graças ao seu caráter acessível e flexível, a plataforma tem se popularizado e vem sendo utilizada com frequência em contextos educacionais e de prototipagem. A utilização é bastante intuitiva, o que possibilita o uso eficiente mesmo por pessoas que não possuem um vasto conhecimento sobre programação. A linguagem derivada de C/C++ caracteriza-se como mais uma vantagem, visto que são linguagens democratizadas, sobretudo, na última década.

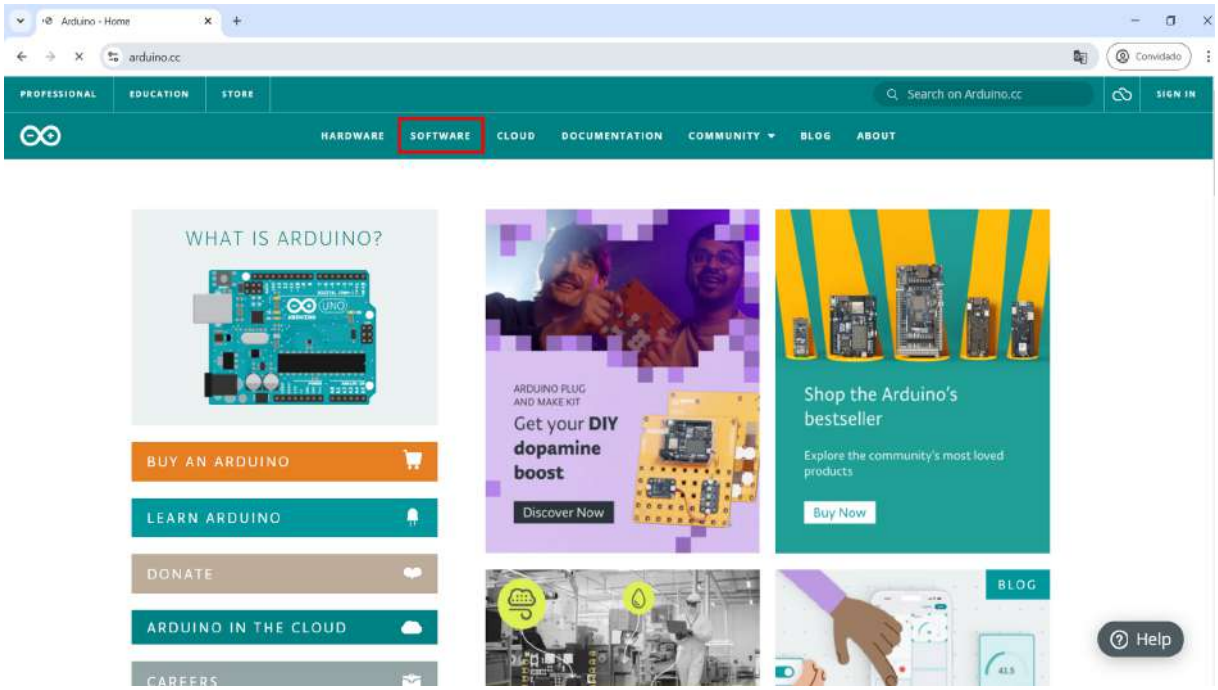
A IDE do Arduíno conta com a ferramenta integrada Serial Plotter, a qual permite visualizar graficamente os dados enviados pela porta serial. Ela é especialmente útil para monitorar em tempo real variáveis de sensores, resultados de cálculos, ou qualquer dado gerado pelo microcontrolador. A representação visual pode ajudar na análise e no entendimento do comportamento de sistemas dinâmicos.

3.1.2.2.1 **Instalação do Ambiente de Desenvolvimento Integrado (IDE)**

Para que o sistema experimental em que se esteja trabalhando funcione da maneira desejada, é indispensável a instalação e configuração correta da IDE. A instalação é bastante simples, como se pode observar nos passos a seguir.

Em um computador, com acesso à internet, é necessário acessar o endereço www.arduino.cc Uma vez no site, a tela mostrada na figura 3.35 estará disponível, então basta seguir os passos seguintes.

Figura 3.35 – Home page da plataforma Arduíno

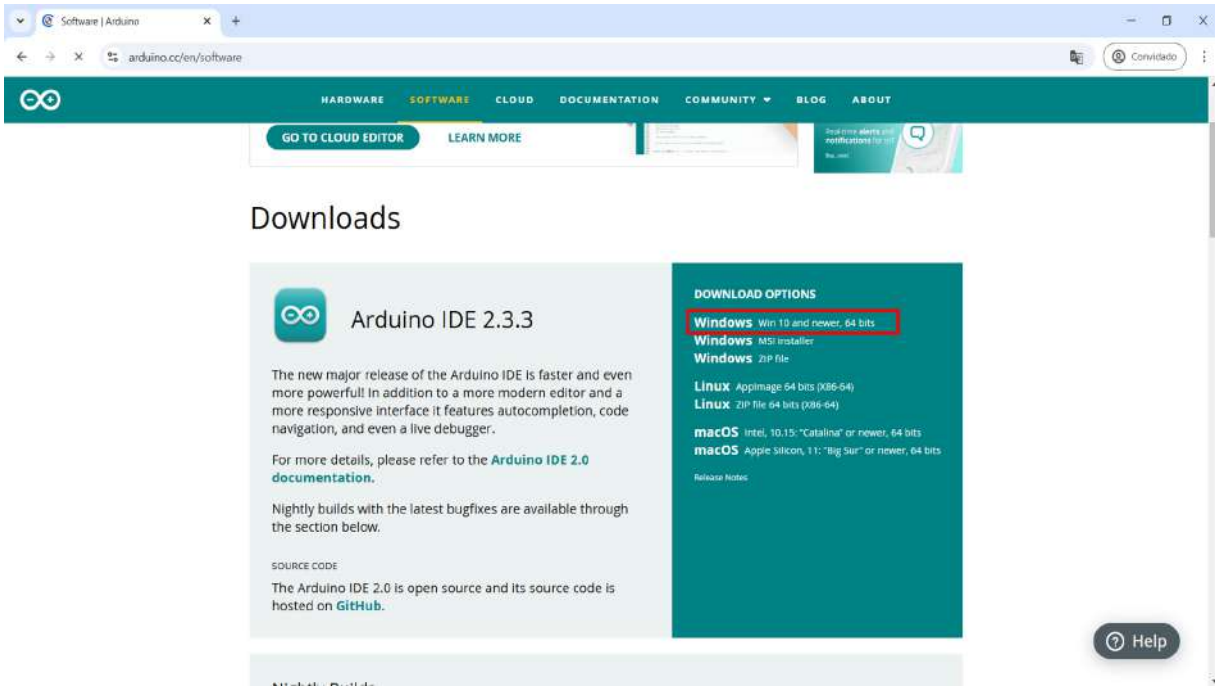


Fonte: Autoria própria

- 2º passo: Escolher o software de acordo com o sistema operacional do computados em que se deseja instalar a IDE.

Ao clicar na aba software, a tela seguinte (figura 3.36) mostrará as opções disponíveis para download. É necessário certificar-se do sistema operacional instalado na máquina a fim de baixar a versão compatível. Para essa exemplificação, será utilizado um computador com a versão Windows 10.

Figura 3.36 – Opções para download

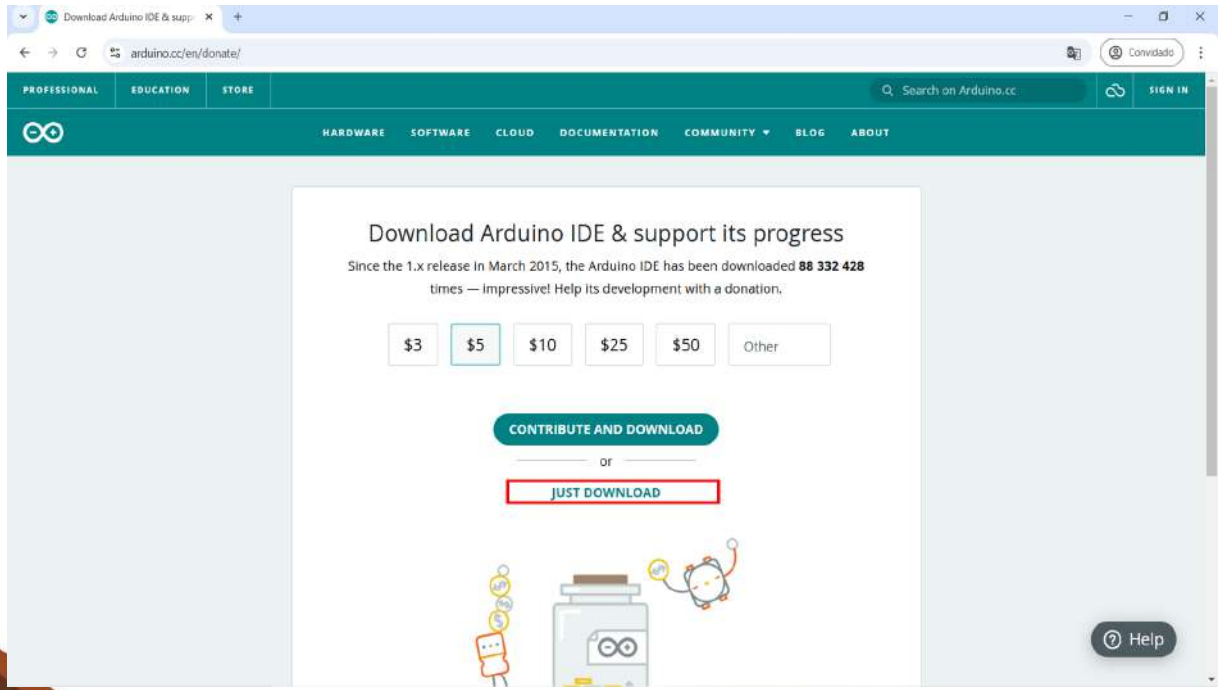


Fonte: Autoria própria

- 3º passo: Iniciando o download.

Ao clicar no sistema operacional, uma nova tela se abrirá (figura 3.37). Nessa tela há a opção de contribuir financeiramente com a plataforma. Entretanto, o acesso ao programa é gratuito e pode ser continuado clicando em "just download".

Figura 3.37 – Iniciando o download



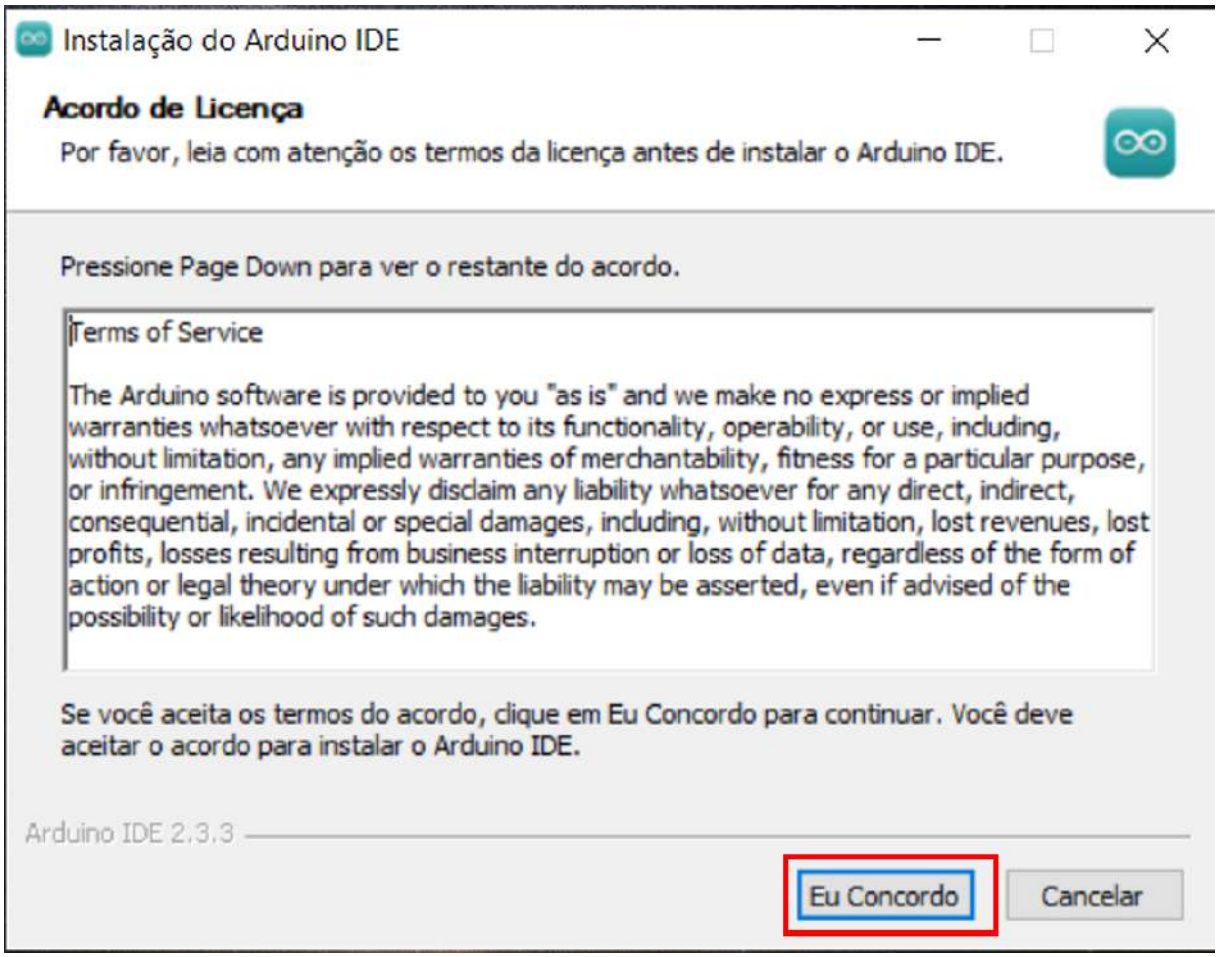
Fonte: Autoria própria

É possível que em alguns casos mais alguma tela abra após essa. Se ocorrer, basta clicar novamente em “just download” e o programa, então, iniciará o download.

- 4º passo: Instalação.

O passo seguinte é instalar o ambiente de desenvolvimento no computador. Para isso, é necessário localizar o arquivo na pasta de downloads e clicar duas vezes com o botão esquerdo do mouse. Ao fazer isso, uma tela de licença será mostrada (figura 3.38).

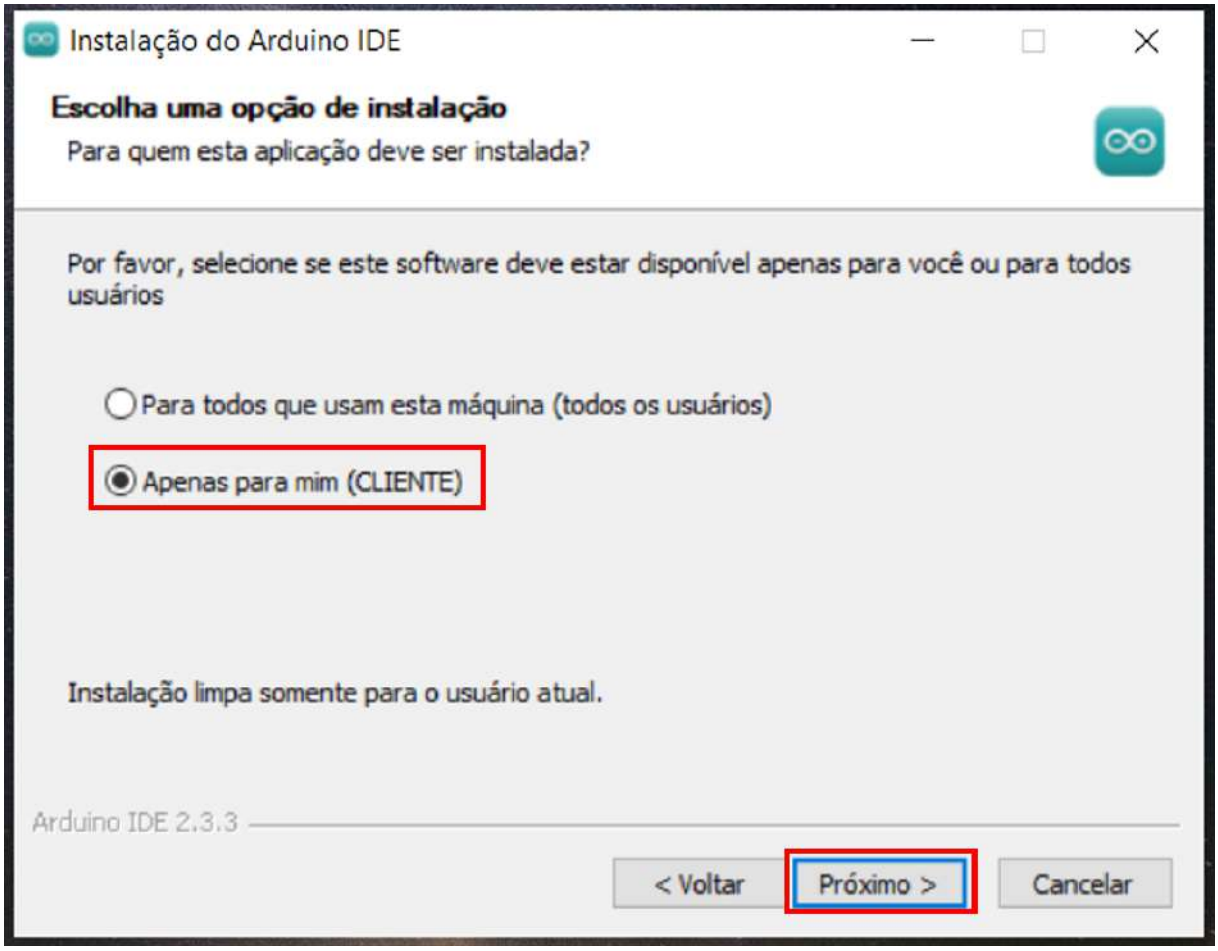
Figura 3.38 – Acordo de licença



Fonte: Autoria própria

Para prosseguir, basta clicar em “eu concordo” e uma nova tela será aberta (figura 3.39). Nesta, é possível optar entre instalar apenas para o usuário ou para todos os usuários da máquina, considerando que existam mais de um. Para essa demonstração, optou-se por instalar apenas para o usuário em execução.

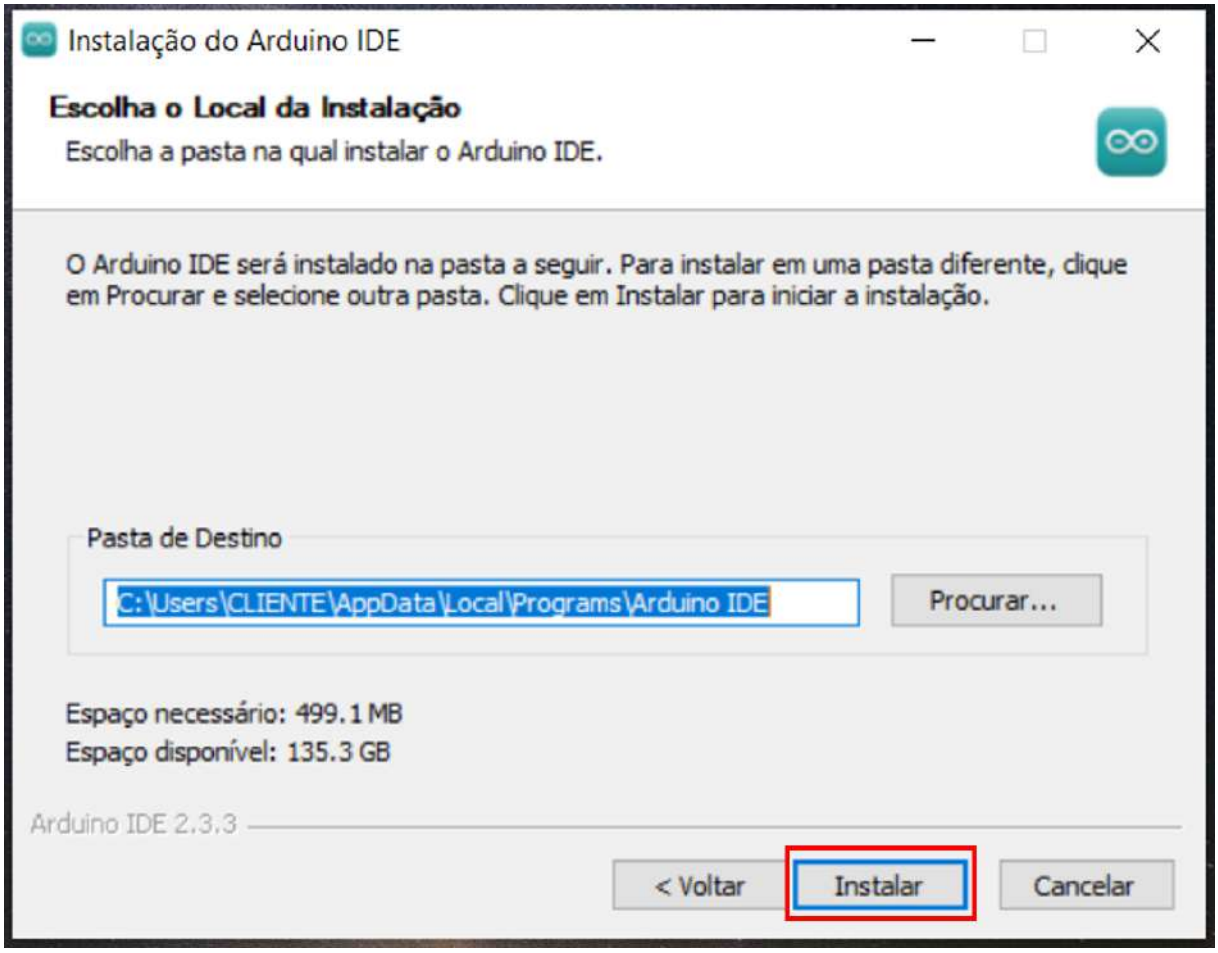
Figura 3.39 – Opções de instalação



Fonte: Autoria própria

Clicando em próximo, a tela seguinte será referente à escolha da pasta onde o usuário deseja instalar a IDE (figura 3.40).

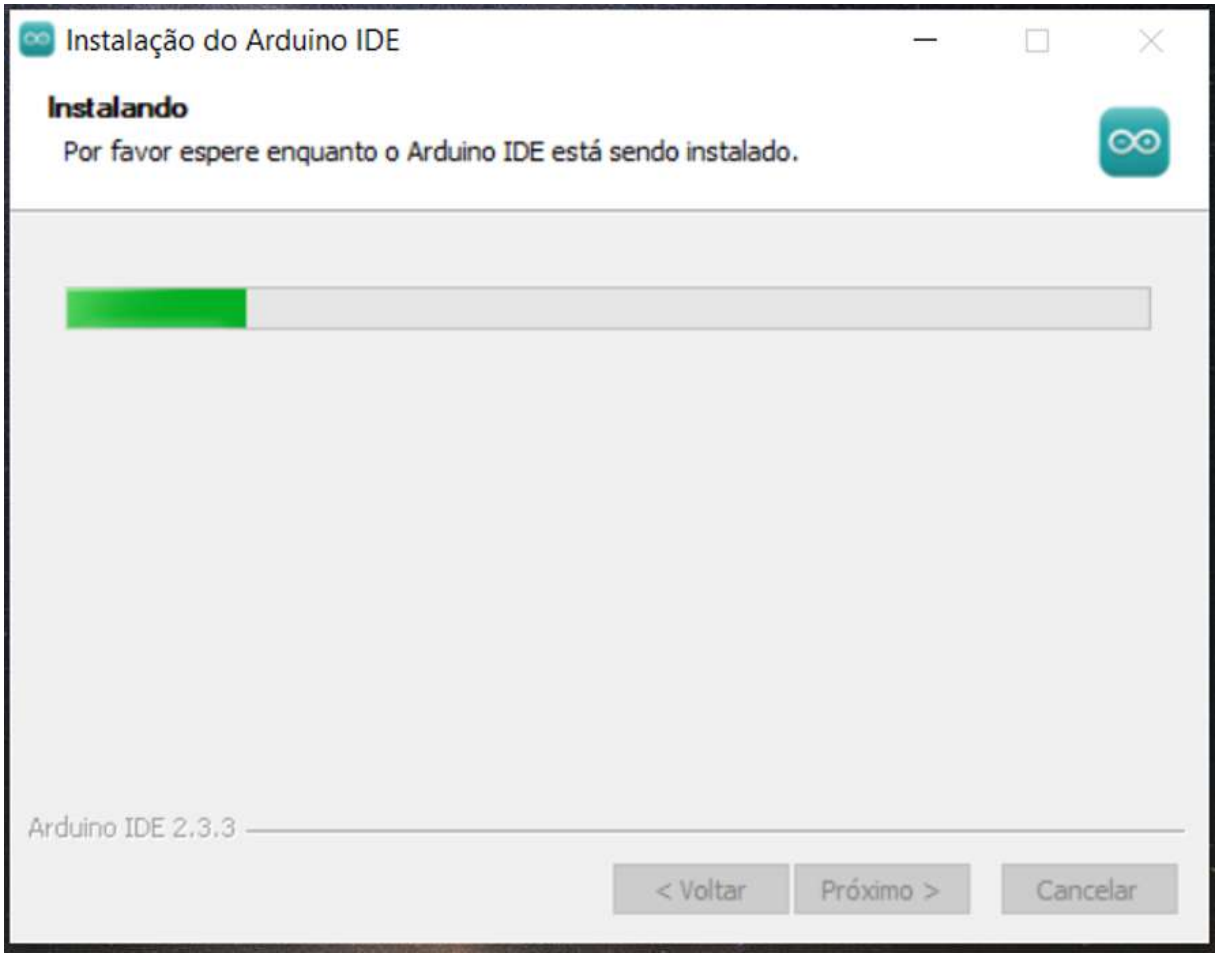
Figura 3.40 – Escolha da pasta de instalação



Fonte: Autoria própria

O programa já traz uma opção pré-definida, mas caso o usuário deseje uma opção diferente deve clicar em “procurar” e escolher a pasta pretendida. Caso contrário, basta clicar em “instalar” para iniciar a instalação. Durante o processo de instalação, a tela ficará carregando, como mostra a imagem 3.41. Esse processo geralmente é rápido, mas pode levar alguns minutos dependendo da máquina em uso.

Figura 3.41 – Progresso da instação



Fonte: Autoria própria

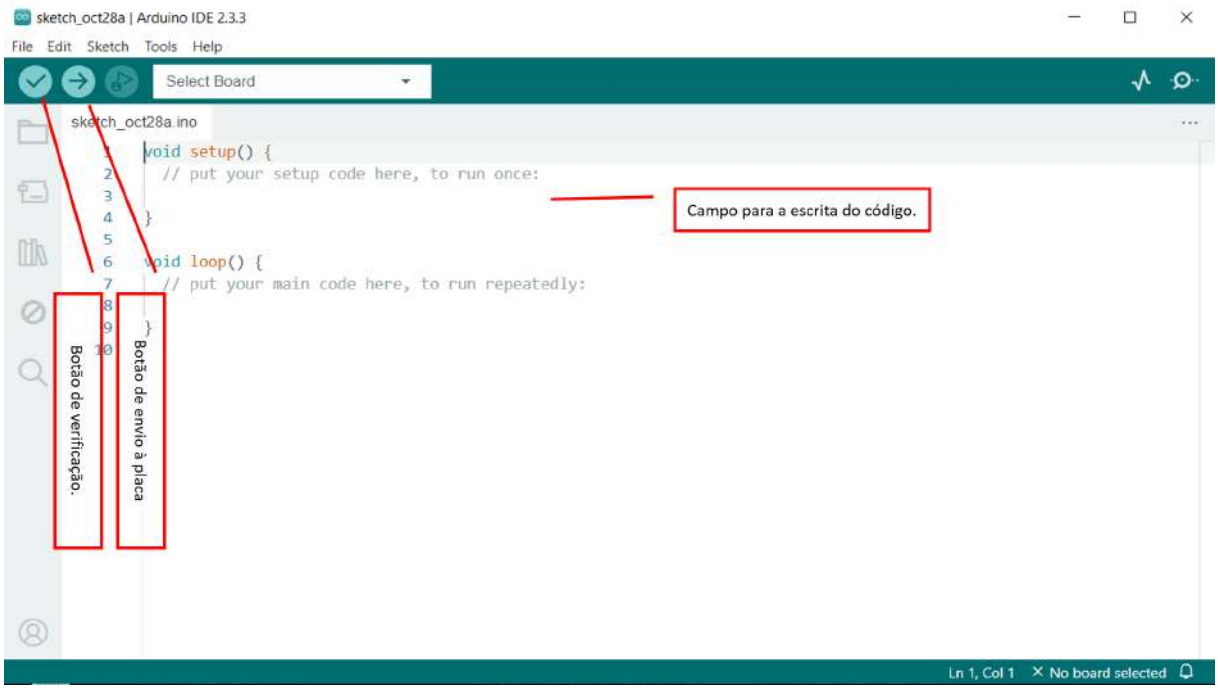
Uma vez carregado, a tela de conclusão será mostrada (imagem 3.42). Então, basta clicar em “concluir” para finalizar a instalação.

Figura 3.42 – Conclusão da instalação

Fonte: Autoria própria

Se a caixa de seleção “Executar o Arduíno IDE” estiver marcada, como na imagem acima, a tela inicial da IDE iniciará e estará pronta para uso (imagem 3.43).

Figura 3.43 – Tela inicial da IDE



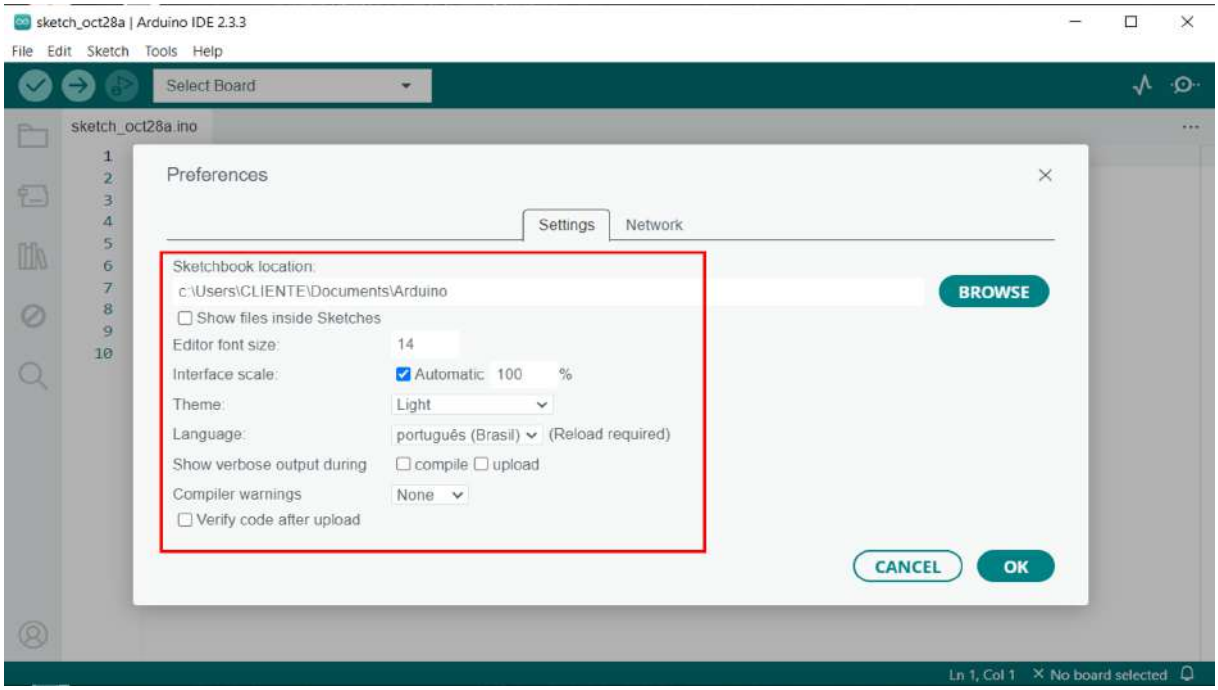
Fonte: Autoria própria

É nesse campo onde o código será escrito, verificado e enviado para a placa programável. Para a utilização da experiência proposta nesse trabalho, é necessário apenas copiar o código fornecido e colá-lo no campo mostrado na imagem acima e enviá-lo à placa.

- Passo 5: Ajustando as preferências.

Realizada a instalação, e uma vez com a tela inicial da IDE aberta, é possível ajustar algumas configurações de acordo com a necessidade ou predileção do utilizador. Clicando em “file” e depois em “preferences” abrirá a tela mostrada na imagem 3.44, onde é possível, entre outras coisas, alterar o tema, as dimensões e o idioma.

Figura 3.44 – Preferências



Fonte: Autoria própria

3.1.2.2.2 **Código Arduíno**

O código desenvolvido no Arduíno permite a medição precisa da posição angular dos encoders conectados ao sistema. Por meio de um processo automatizado, o Arduíno monitora continuamente os sinais provenientes do encoder e calcula o ângulo de rotação, que é exibido em intervalos regulares. A funcionalidade principal do código é identificar mudanças nos sinais gerados pelo encoder e traduzir essas variações em incrementos ou decrementos no valor do ângulo, dependendo da direção do movimento do pêndulo.

Esses dados são utilizados para analisar o comportamento dinâmico do sistema, incluindo aspectos como a interação entre os pêndulos, o acoplamento e as propriedades oscilatórias. O código desenvolvido está disponível no apêndice B deste trabalho.

A figura 3.45 ilustra a interface do Serial Plotter da IDE do Arduíno, ferramenta que permite a visualização gráfica dos dados transmitidos pelo microcontrolador em tempo real. A interface exibe as informações coletadas em forma de gráficos, permitindo identificar padrões de oscilação, amplitude e frequência.

Figura 3.45 – Teste do código Arduíno

Fonte: Autoria própria

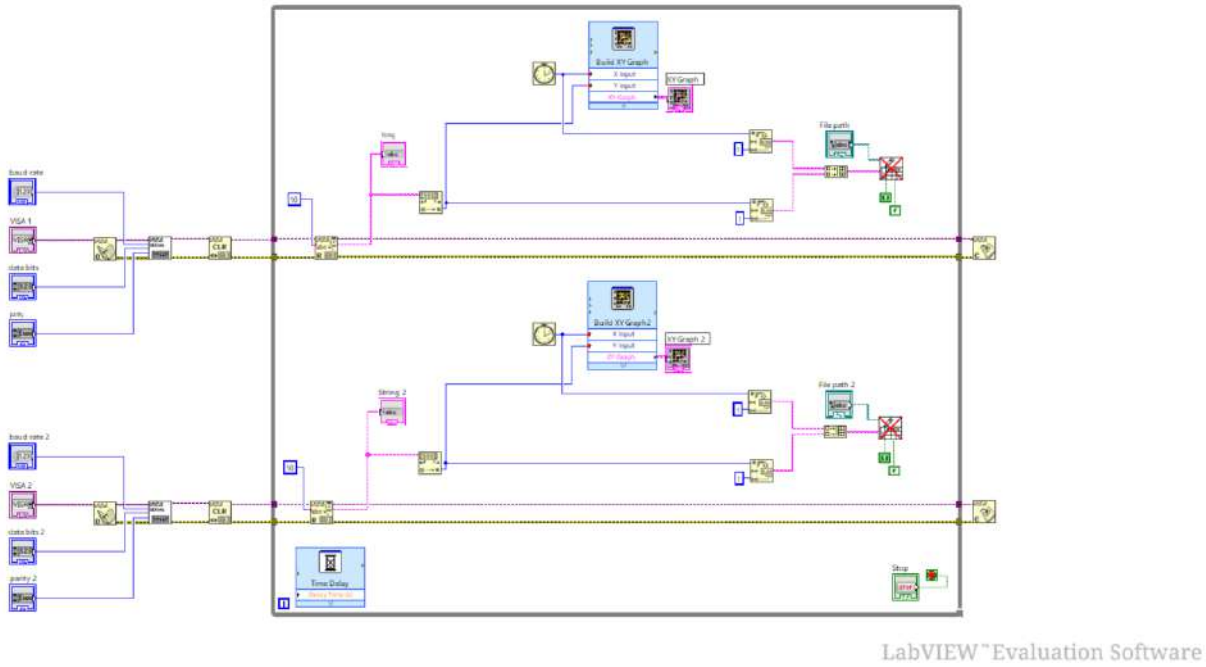
3.1.2.3 ***Ambiente gráfico de programação LabVIEW***

Apesar da possibilidade de observação dos gráficos na própria plataforma Arduíno, um código para este fim foi desenvolvido no ambiente de desenvolvimento e linguagem de programação gráfica Labview. O programa, assim como a ferramenta serial plotter do Arduíno, também permite a apresentação gráfica dos resultados em tempo real de funcionamento do mecanismo com a vantagem de ser uma ferramenta que permite criar interfaces customizáveis, incluindo botões, indicadores digitais, controles deslizantes, entre outros componentes interativos.

Além dessas vantagens, o Labview ainda permite a conexão de vários dispositivos simultaneamente via diferentes interfaces (USB, Ethernet, Serial, etc.); permite salvar dados em arquivos como .csv, .txt ou bancos de dados automaticamente para análises posteriores; consegue gerenciar grandes volumes de dados de forma eficiente, permitindo o processamento e a visualização em alta frequência; facilita a geração de relatórios automatizados e detalhados, incluindo gráficos, tabelas e descrições, em formatos como PDF ou Word, entre várias outras possibilidades.

A figura 3.46 mostra o diagrama de blocos do código desenvolvido no Labview para a visualização dos gráficos. Essa configuração organiza os elementos de um programa em blocos funcionais e conexões visuais que representam o fluxo de dados entre esses blocos.

Figura 3.46 – Diagrama de blocos em LabVIEW



Fonte: Autoria própria

A organização estrutural do diagrama de blocos acima conta com três entradas (Baud Rate, Data Bits e Parity). Essas entradas definem os parâmetros de comunicação serial (taxa de transmissão, número de bits de dados e verificação de paridade) para conectar o dispositivo externo ao sistema LabVIEW. São essenciais para garantir que o software se comunique corretamente com o hardware. Para gerenciar a comunicação serial com os dispositivos, é utilizada a biblioteca Virtual Instrument Software Architecture (VISA). Ela configura a porta serial para iniciar a troca de dados.

Os dados recebidos da comunicação serial são armazenados como strings, sequências de caracteres ASCII (American Standard Code for Information Interchange) que podem ser exibidas ou não. As strings são convertidas em números, permitindo que os dados sejam tratados como entradas para cálculos ou gráficos.

Após essa conversão, o controle gráfico XY (*Build XY Graph*) recebe os dados, agora numéricos, e os organiza para gerar gráficos XY em tempo real. A entrada *X Input* representa o eixo X (geralmente uma variável independente, como o tempo nesse caso), enquanto *Y Input* corresponde ao eixo Y (a variável dependente medida ou simulada, nesse caso, o deslocamento angular). O bloco *XY Graph* representa os gráficos finais que mostram a relação entre as variáveis monitoradas (tempo e deslocamento angular).

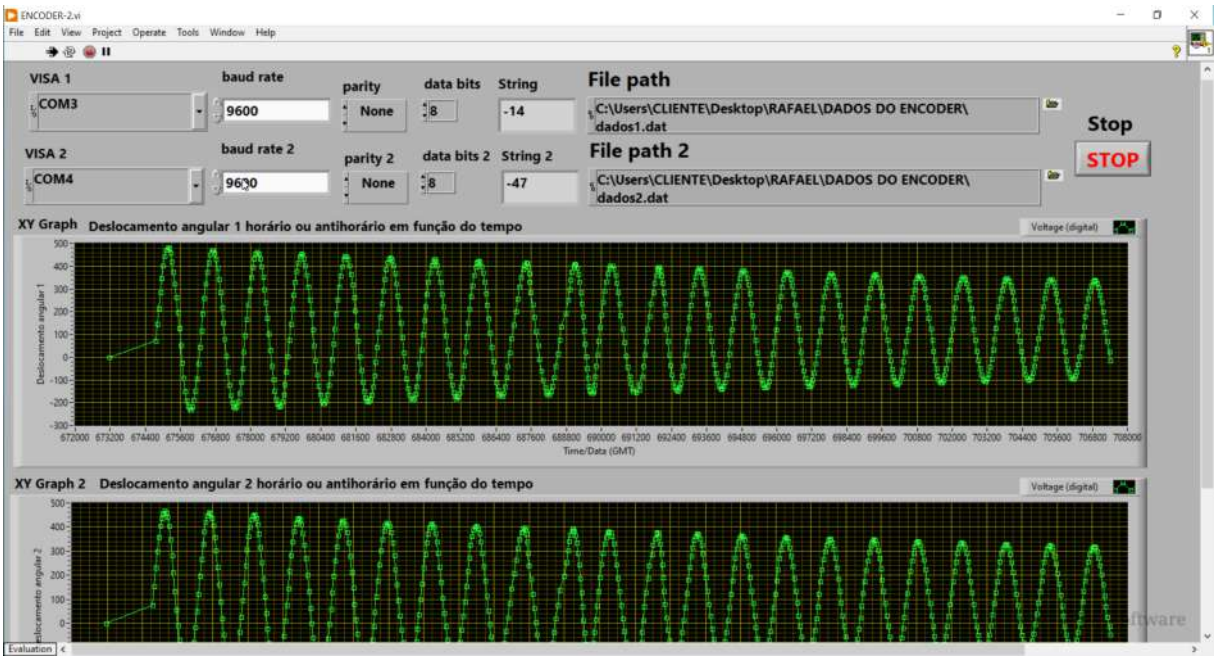
Quanto ao armazenamento de dados, o bloco *File Path* define o caminho do arquivo onde os dados recebidos serão armazenados. É utilizado para salvar os dados em formato texto ou planilha, permitindo análises posteriores. Já o *Write to Text File* salva os dados recebidos no formato de texto. Isso é útil para registrar os resultados da aquisição de dados em tempo real.

O *laço While* é usado para manter o programa em execução enquanto a condição de parada (botão *Stop*) não for acionada. Ele garante a continuidade da leitura de dados e atualização dos gráficos em tempo real. O *Time Delay* garante que a leitura dos dados

ocorra em intervalos regulares, evitando sobrecarga no sistema. O tempo de atraso entre cada ciclo do *laço While* é configurado para sincronizar a coleta de dados. Por fim, o botão de parada *Stop* permite ao usuário interromper a execução do programa de forma controlada. Ao pressionar este botão, o *laço While* termina e o programa finaliza sua execução.

A figura 3.47 mostra a interface do programa, onde é possível visualizar os elementos descritos acima, bem como os gráficos gerados do movimento oscilatório dos dois pêndulos simples acoplados pela mola de pequena constante elástica.

Figura 3.47 – Interface do Labview



Fonte: Autoria própria

4 APLICAÇÃO

Neste trabalho, utilizar-se-á a metodologia de sequências de ensino investigativo com demonstração experimental. Essa metodologia configura-se como uma abordagem pedagógica cujo objetivo é promover a aprendizagem ativa e o desenvolvimento de habilidades científicas nos estudantes, por meio da realização de investigações e experimentos práticos em sala de aula.

Essa sequência de ensino geralmente é dividida em três fases: a primeira é a fase exploratória, em que os estudantes são convidados a observar e descrever um fenômeno ou problema a ser investigado. Na segunda etapa, chamada de investigação propriamente dita, os estudantes são orientados a formular hipóteses, planejar e realizar experimentos para testar suas hipóteses. Na terceira etapa, a fase de comunicação, os estudantes são estimulados a compartilhar seus resultados com a classe, discutir suas descobertas e refletir sobre o processo investigativo (CARVALHO, A. M. P. de, 2018).

A demonstração, quando trabalhada de forma investigativa, ganha caráter representativo e ilustrativo de conceitos científicos de forma prática e visual. Através dela, o professor pode mostrar aos estudantes os efeitos e resultados esperados de um experimento, antes que eles o realizem de fato.

4.1 Etapas da Sequência de Ensino Investigativo (SEI)

A Sequência de Ensino Investigativo vem ganhando notoriedade dentro do contexto das metodologias ativas graças ao seu potencial em promover a aprendizagem significativa. No Brasil, uma das principais referências é a professora Anna Maria Pessoa de Carvalho, cuja dedicação à adaptação para diversos contextos, à formação de professores e à pesquisa contribuiu para tornar a SEI uma metodologia eficaz e amplamente reconhecida na educação brasileira.

Segundo Anna Maria Pessoa de Carvalho (2022), tal metodologia trata-se de um procedimento pedagógico que tem como objetivo promover a aprendizagem ativa e o desenvolvimento de habilidades científicas nos alunos por meio da realização de experimentos práticos em sala de aula.

A aplicação da Sequência de Ensino Investigativo, apesar das possíveis variações, pode ser dividida em três etapas: apresentação ou manipulação do material experimental e proposição do problema; identificação e exploração das hipóteses/previsões dos alunos; e elaboração de possíveis planos de ação e execução do planejado.

A Figura 4.1 mostra, de forma resumida, como normalmente a metodologia apresentada se estabelece.

Figura 4.1 – Sequência de Ensino Investigativo

Fonte: Autoria própria

Apesar das claras definições de fases que essa metodologia abrange, e que são mostradas a seguir, a SEI possui variações, o que a torna um instrumento dinâmico e não delimitado em etapas restritas. É comum perceber essas variações quando se tem aplicações de problemas experimentais complexos ou sensíveis que não permitem a manipulação pelo próprio aluno. Nesse caso, por exemplo, o aluno inicia como observador e a SEI configura-se com caráter demonstrativo. Em outros casos, com experimentos cuja manipulação descarta a possibilidade de dano ao aluno, ele pode participar ativamente desde a construção até a realização do experimento.

4.1.1 Introdução e Motivação

Na etapa inicial, o objetivo é despertar o interesse dos alunos pelo tópico a ser estudado e motivá-los para a investigação científica. Isso é fundamental para criar um ambiente propício à aprendizagem e à curiosidade. Algumas das atividades típicas nesta fase incluem:

- ***Ativação do conhecimento prévio***

  O mediador inicia com perguntas norteadoras a fim de perceber o que os alunos já sabem sobre o assunto em questão. Isso ajuda a identificar as ideias pré-concebidas dos alunos e a criar uma base para a construção do conhecimento.

- ***Apresentação do tópico***

  O professor apresenta o tópico de estudo de maneira instigante e relevante para a vida dos alunos, mostrando sua importância e aplicações práticas. Nessa etapa, é importante a adequação da linguagem, visto que, no primeiro momento, os termos técnicos não terão tanto significado para o aluno.

- ***Contextualização***

  É importante contextualizar a prática dentro do currículo escolar e da vida cotidiana dos alunos, explicando como ele se relaciona com outros conceitos e temas. A aplicação de uma SEI, quando não é rotineira, pode gerar no aluno a ideia de singularidade, podendo este entender como uma atividade extra e sem relação com o restante do currículo ou com assuntos anteriores.

- ***Questionamento***

  O professor estimula os alunos a fazerem perguntas sobre o tópico, promovendo a curiosidade e a investigação. Esse processo também serve como verificação do entendimento da proposta. Ao questionar, o aluno demonstra o nível de clareza quanto à situação. O mediador atento consegue discernir quando sua explicação não foi tão eficaz a partir das questões colocadas pelos estudantes.

4.1.2 Investigação e Exploração

Nesta segunda etapa, os alunos são envolvidos em atividades práticas de investigação, onde exploram, coletam dados, manipulam experimentos e investigam a proposta trabalhada na SEI em profundidade. Podemos destacar como características desta fase:

- ***Atividades práticas***

  Os estudantes participam de atividades práticas que envolvem a coleta de dados, experimentação e/ou observação. Eles podem trabalhar individualmente ou em grupos, dependendo da natureza do projeto. Apesar das interações defendidas por Vygotsky, a aplicação da SEI não está restrita à ocorrência em equipes.

- ***Mediação do professor***

  O professor atua como um mediador durante boa parte do processo, no entanto, nessa etapa, ele age de maneira a orientar, esclarecer dúvidas e fornecer recursos e suporte quando necessário, mas sem interferir na organização das ideias dos indivíduos, permitindo assim que os alunos conduzam suas próprias investigações..

- ***Registro de dados***

  Ainda nessa etapa da SEI, os alunos registram e analisam os dados coletados, fazem anotações e formulam hipóteses à medida que avançam em suas investigações. A colaboração entre eles é incentivada, promovendo discussões e compartilhamento de ideias.

4.1.3 Síntese e Compartilhamento

Na última etapa, os alunos consolidam suas descobertas e compartilham-nas com a turma. Isso proporciona reflexão, diálogo e aplicação do conhecimento adquirido. Algumas atividades que configuram essa etapa incluem:

- ***Apresentação dos resultados***

Os estudantes socializam suas descobertas e elucidações para a turma através de relatórios, apresentações orais, experimentos ou outras formas de comunicação.

- ***Discussão e reflexão***

  Os estudantes discutem os resultados obtidos pela equipe expositora, observam e analisam as implicações do exposto com os conceitos estabelecidos anteriormente ou com suas próprias descobertas.

- ***Aplicação do conhecimento***

  O mediador incentiva os estudantes a aplicar o conhecimento construído em situações mais abrangentes ou diferentes das trabalhadas inicialmente. A proposta deve encorajá-los em investigações mais complexas ou relacionadas com o mundo real.

4.2 Aplicação do produto

Tendo como base a Sequência de Ensino Investigativo (SEI), a metodologia utilizada na aplicação desse trabalho pode ser dividida em duas fases: discussões sobre os conteúdos que fundamentam o aparato experimental numa abordagem expositiva, fomentada por demonstrações simples e práticas, tendo como centro de interesse o aluno; e investigação prática do aparato, com manuseio, interpretação e formulação de hipóteses por parte dos estudantes, baseados no que foi visto na introdução teórica.

Reconhece-se que a introdução de práticas investigativas requer, em muitos casos, uma etapa inicial que assegure que os participantes dominem os conceitos e as ferramentas básicas necessárias para a realização de investigações mais profundas. Dada a complexidade do sistema experimental escolhido — dois pêndulos acoplados por uma mola — e os conteúdos a ele associados, optou-se por estruturar cinco encontros com abordagem expositiva e dialogada, com foco em apresentar os conceitos teóricos e as ferramentas fundamentais. Essa etapa foi planejada para promover uma base sólida de conhecimento, facilitando a transição dos estudantes para a etapa de investigação prática.

No sexto encontro, desenvolve-se uma atividade investigativa que contempla todas as etapas da metodologia SEI (problematização, construção de hipóteses, planejamento experimental, coleta e análise de dados e discussão de resultados). Essa abordagem permite que os estudantes apliquem os conhecimentos adquiridos nos encontros anteriores de forma autônoma e crítica, integrando os princípios da metodologia investigativa ao desenvolvimento prático.

Ao estruturar a aplicação dessa maneira, buscou-se equilibrar o aprofundamento conceitual e o desenvolvimento das habilidades investigativas, mantendo a coerência com os objetivos da metodologia SEI e maximizando o potencial de aprendizado dos estudantes.

A tabela a seguir (tabela 4.1) mostra a estruturação dos encontros, bem como as atividades a serem desenvolvidas em cada um deles. A organização dos conteúdos desenvolvidos em cada encontro e o roteiro de aplicação da SEI podem ser vistos no apêndice C deste produto.

Encontro	Objetivo	Atividades	Resultado Esperado
1º	Identificar os conhecimentos prévios dos estudantes sobre os conceitos básicos da Física.	- Aplicação de um **questionário diagnóstico**. - Discussão inicial sobre a percepção da Física no cotidiano.	Mapeamento dos conhecimentos prévios e principais dificuldades.
2º	Introduzir os fundamentos da Física e situar seu papel na compreensão de fenômenos naturais.	**- Apresentação teórica:** - Objeto de estudo da Física. - Conceitos de sistemas físicos. - Importância da Física Newtoniana.	Compreensão inicial dos fundamentos da Física como ciência.
3º	Explorar os conceitos de energia cinética, potencial e conservação da energia mecânica.	- Demonstrações práticas simples. - Vídeos curtos e simulações sobre transformação de energia. - Discussão sobre aplicações em sistemas reais.	Relacionar os conceitos de energia a fenômenos observáveis e compreendê-los.
4º	Conceituar força restauradora e explorar sua relação com movimentos oscilatórios.	- Introdução à Lei de Hooke e à força elástica com experimentos simples. - Discussão sobre força restauradora. - Representação gráfica.	Entendimento da força elástica e sua aplicação em sistemas físicos.
5º	Abordar o MHS e aplicar os conceitos ao pêndulo simples.	- Apresentação dos conceitos de MHS, amplitude, frequência e período. - Experimento prático com pêndulos. - Discussão sobre aplicações práticas.	Capacidade de relacionar os conceitos de MHS e pêndulo simples à prática experimental.
6º	Aplicar os conceitos de energia mecânica, forças restauradoras e movimentos oscilatórios.	- Problema investigativo: Conservação de energia em sistemas oscilatórios. - Planejamento e execução de experimentos pelos alunos. - Discussão.	Desenvolvimento do raciocínio científico, integração dos conceitos e apresentação de conclusões.

Tabela 4.1 – Sequência de aulas

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo principal a promoção de um ensino de Física inovador e motivador, por meio de um produto educacional integrado. Esse produto, composto por um sistema experimental com aquisição automática de dados e um e-book estruturado como manual de construção e aplicação, busca explorar os conceitos relacionados à Mecânica, com ênfase em sistemas não lineares, como os pêndulos acoplados. A proposta, fundamentada na Sequência de Ensino Investigativo (SEI), é difundida no Brasil por Anna Maria Pessoa de Carvalho, priorizando o aprendizado ativo, investigativo e significativo.

A experimentação, enquanto estratégia de ensino, sempre se mostrou uma ferramenta eficaz para superar os entraves relacionados à complexidade atribuída à abstração dos conceitos físicos. Quando integrada à tecnologia, demonstra que avanços tecnológicos, como sensores, softwares e simulações, podem enriquecer ainda mais o processo de ensino e aprendizagem. No presente trabalho, esses recursos ofereceram aos estudantes uma forma mais interativa e visual de explorar fenômenos físicos, conectando o conteúdo teórico à prática de maneira concreta e significativa.

Apesar das limitações frequentemente enfrentadas, como restrições de infraestrutura e a necessidade de capacitação dos professores no uso das ferramentas tecnológicas, o trabalho destaca a viabilidade de integrar práticas experimentais ao ensino de Física, mesmo em contextos desafiadores, como o das escolas públicas no Brasil. A utilização de recursos de baixo custo, associados a metodologias bem estruturadas, mostra-se uma alternativa viável para ampliar o acesso dos estudantes a uma aprendizagem mais dinâmica e prática. Isto, por sua vez, abre espaço para a implementação de práticas semelhantes em outras áreas do conhecimento, não se restringindo apenas ao ensino de Física.

Com base na experiência vivenciada desde a construção até a aplicação, conclui-se que o produto educacional desenvolvido possui grande potencial como ferramenta de ensino eficaz em laboratórios escolares ou mesmo na sala de aula, incentivando o aprendizado ativo e a formação de um pensamento crítico e reflexivo por parte dos estudantes. A experimentação, tanto científica quanto didática, continuará sendo um dos pilares para a construção do conhecimento em Ciências, e iniciativas como esta reforçam a importância de investir em abordagens pedagógicas inovadoras para melhorar a qualidade do ensino.

Dada a relevância do desenvolvimento de mecanismos como o apresentado neste trabalho, recomenda-se, como continuidade das pesquisas, a aplicação do produto em diferentes contextos educacionais e a realização de estudos complementares que avaliem sua efetividade em longo prazo. Além disso, sugere-se a criação de formações continuadas para professores, com o objetivo de capacitá-los no uso de tecnologias aplicadas ao ensino experimental. Espera-se que esta iniciativa inspire futuras pesquisas e contribuições voltadas para a melhoria do ensino de Física e demais disciplinas científicas.

REFERÊNCIAS

Artigo de periódicos

BISINOTTO, G. A.; CANCIAN, C. G.; DE OLIVEIRA, L. F. G. Memorial do Projeto De Um Dispositivo Mecânico Autocontrolado Por Encoder Óptico e Controlado Remotamente Via Bluetooth. **Mecatrone**, v. 1, n. 1, 2015. Citado 1 vez na página 43.

CARVALHO, A. M. P. de. Fundamentos teóricos e metodológicos do ensino por investigação. **Revista Brasileira de Pesquisa em Educação em Ciências**, p. 765–794, 2018. Citado 1 vez na página 96.

_____. Ensino de ciências por investigação: condições para implementação em sala de aula, 2022. Citado 1 vez na página 96.

CUNHA, F. C. M. et al. Revisitando modos normais de oscilações acopladas com a Teoria Espectral de Grafos. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 46, e20240095, 2024. Nenhuma citação no texto.

DOEBELIN, E. O.; MANIK, D. N. Measurement systems: application and design, 2007. Citado 1 vez na página 42.

FERRAZ, W.; OLIVEIRA JUNIOR, A. Sistema optico de aquisicao de sinais de velocidade e posicao de maquinas (encoder incremental com quadratura geometrica). **Anais**, 1992. Citado 1 vez na página 43.

INÁCIO, M. J. Sensores e Atuadores. **Faculdade de Ciências do Tocantins**, 2009. Citado 1 vez na página 44.

LEE, J.; BAGHERI, B.; KAO, H.-A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. **Manufacturing letters**, Elsevier, v. 3, p. 18–23, 2015. Citado 1 vez na página 42.

ROSSUM, G. van. Guido van Rossum. **Python (programming language) 1 CPython 13 Python Software Foundation 15**, Citeseer, p. 16. Citado 1 vez na página 49.

SILVA, J. L. d. S.; MELO, M. et al. Plataforma Arduino integrado ao PLX-DAQ: Análise e aprimoramento de sensores com ênfase no LM35. **XIV Escola Regional de Computação Bahia, Alagoas e Sergipe (ERBASE). Feira de Santana, BA**, 2014. Nenhuma citação no texto.

SILVA, R. O. da; ARAUJO, W. M.; CAVALCANTE, M. M. Visão geral sobre microcontroladores e prototipagem com Arduino. **Tecnologias Em Projeção**, v. 10, n. 1, p. 36–46, 2019. Citado 1 vez na página 46.

Artigo de anais de conferência

VAN ROSSUM, G. et al. Python programming language. In: SANTA CLARA, CA, 1. USENIX annual technical conference. [S.l.: s.n.], 2007. v. 41, p. 1–36. Citado 1 vez na página 49.

Livro

GROOVER, M. P. **Automation, production systems, and computer-integrated manufacturing**. [S.l.]: Pearson Education India, 2016. Citado 1 vez na página 42.

MARION, J. B. **Classical dynamics of particles and systems**. [S.l.]: Academic Press, 2013. Nenhuma citação no texto.

MCROBERTS, M. **Arduino Básico-2ª edição: Tudo sobre o popular microcontrolador Arduino.** [S.l.]: Novatec Editora, 2015. Citado 2 vezes nas páginas 47, 48.

THORNTON, S. T.; MARION, J. **Classical dynamics of particles and systems**. [S.l.]: MTM, 2019. Citado 1 vez na página 11.

VAN ROSSUM, G.; DRAKE JR, F. L. **Python tutorial**. [S.l.]: Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995. v. 620. Citado 1 vez na página 49.

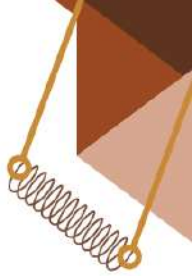
TCC

TIAGO, L. L. **Projeto de um simulador de atitude com três graus de liberdade**. 2013. Escola Politécnica da Universidade de São Paulo, São Paulo, SP. Graduação em Engenharia Mecatrônica. Citado 1 vez na página 46.

Referências online

RIFKIN, J. **The third industrial revolution**. [S.l.]: Palgrave macmillan, 2011. Citado 1 vez na página 42.

APÊNDICES



APÊNDICE A

EXECUTÁVEIS DAS SIMULAÇÕES EM PYTHON

PÊNDULO SIMPLES

Fonte: Autoria própria

PÊNDULO SIMPLES AMORTECIDO

Fonte: Autoria própria

PÊNDULO SIMPLES FORÇADO

Fonte: Autoria própria

PÊNDULO SIMPLES ACOPLADOS

Fonte: Autoria própria

APÊNDICE B

Código Arduíno

CÓDIGO ARDUÍNO

arduino__tesste_encoder_ino | Arduino IDE 2.3.3

Arquivo Editar Rascunho Ferramentas Ajuda

Selecionar Placa

arduino__tesste_encoder_ino.ino

```
//Declaração de variáveis globais

// Define os pinos digitais 2 e 3 como entradas para os sinais do encoder incremental.
//Esses pinos receberão os sinais dos canais A e B do encoder.
int encoderPin1 = 2;
int encoderPin2 = 3;
//interruptPin1 = 2;  alternativa
//interruptPin2 = 3;  alternativa
volatile int lastEncoded = 0;                          //Declara que a variável pode mudar inesperadamente (interrupções a alteram).
//lastEncoded: Armazena o último estado codificado dos sinais A e B.
volatile long encoderValue = 0;                        //Variável que armazena o valor atual do ângulo ou posição relativa calculada pelo encoder.
long lastencoderValue = 0;                             //Último valor conhecido de encoderValue, usado para comparação ou cálculos posteriores.
int lastMSB = 0;                                       //Armazena o último valor do sinal A do encoder.
int lastLSB = 0;                                       //Armazena o último valor do sinal B do encoder.

void setup() //Inicializa a comunicação serial com velocidade de 9600 bits por segundo para enviar dados ao monitor serial.
{
  Serial.begin (9600);
  pinMode(encoderPin1, INPUT_PULLUP);                  // define-se a porta digital como entrada com pull-up ativado (retorna nada)
  pinMode(encoderPin2, INPUT_PULLUP);                  // define-se a porta digital como entrada com pull-up ativado (retorna nada)
  digitalWrite(encoderPin1, HIGH);                     // Se nenhuma entrada esta presente coloca-se pino 2 como HIGH +5V
  digitalWrite(encoderPin2, HIGH);                     // Se nenhuma entrada esta presente coloca-se pino 3 como HIGH +5V
  attachInterrupt(0, updateEncoder, CHANGE);           // chamar a subrotina updateEncoder() quando qualquer mudança alto/baixo seja detectada na
  //porta e acciona a interrupção (no interrupt 0 (pin 2), ou no interrupt 1 (pin 3))
  attachInterrupt(1, updateEncoder, CHANGE);
//  attachInterrupt(digitalPinToInterrupt(2), updateEncoder, CHANGE);       alternativa
//  attachInterrupt(digitalPinToInterrupt(3), updateEncoder, CHANGE);       alternativa
}

                                                                                                                    //PAGINA 01
```

```
//Laço principal (loop)
//Envia o valor atual de encoderValue (posição calculada) para o monitor serial a cada 50 ms, facilitando a visualização da posição em tempo real.

void loop()
{
  Serial.println(encoderValue);     //imprime o valor do angulo
  delay(50);
}
//*****************************************************************************************************************************************
//Função de interrupção (updateEncoder)
void updateEncoder()
//Essa função é executada automaticamente quando ocorre uma mudança no sinal do encoder.
{
  int MSB = digitalRead(encoderPin1);                //MSB = bit mais significativo.  Lê a porta 2 e atribui o valor à variável MSB, poderia ser sinal A = MSB
  int LSB = digitalRead(encoderPin2);                //LSB = bit menos significativo. Lê a porta 3 e atribui o valor à variável LSB, poderia ser sinal B = LSB

  int encoded = (MSB << 1) | LSB;                    //é aumentado um zero à direita do binario MSB (A), emseguida é comparado com LSB (B) é formado um binario
  //com os digitos mais significativos dos dois A e B esse valor atribuido a encoded.
  int sum  = (lastEncoded << 2) | encoded;           //é aumentado dois zeros à direita do binario lastEncoded, emseguida é comparado com encoded e formado um
  //binario com os digitos mais significaticos dos dois esse valor e atribuido a soma.

  if(sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011) encoderValue ++;       //A partir de qualquer posição, se rotação horária aumentar angulo
  //em um passo
  if(sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000) encoderValue --;       //A partir de qualquer posição, se rotação antihorária diminuir
  //angulo em um passo

  lastEncoded = encoded;                                                                      //armazena encoded em lastencoded para a próxima vez.
}
                                                                                                                    //PAGINA 02
```

APÊNDICE C

Slides a serem utilizados na aplicação do produto e roteiro de aplicação da SEI

APÊNDICE C1 - SLIDES UTILIZADOS NA APLICAÇAO DO PRODUTO

Questionário diagnóstico

3

Encontro II

4

O que estuda a Física?

1. Estuda as propriedades fundamentais da natureza, bem como os fenômenos relacionados à matéria e à energia.

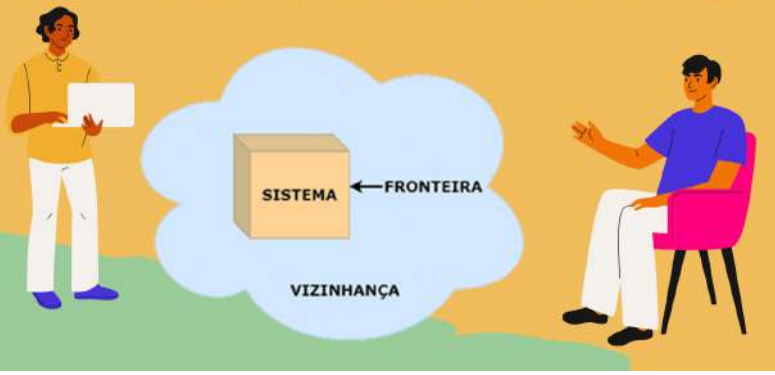
2. Busca entender os princípios que regem o universo e suas implicações.

3. Estuda desde minúsculas partículas subatômicas até grandes estruturas cósmicas, desde movimentos simples até fenômenos complexos como a relatividade e a mecânica quântica.

5

Sistemas Físicos

Um sistema físico é uma entidade ou conjunto de objetos que podem ser estudados e analisados com base em princípios e leis da física.



6

Sistema físico aberto

Um sistema físico aberto é aquele que pode trocar tanto matéria quanto energia com sua vizinhança.

Sistema físico fechado

Um sistema físico fechado é aquele que permite apenas a troca de energia com sua vizinhança, mas não troca matéria.

Sistema físico isolado

Um sistema físico isolado é aquele que não troca nem matéria nem energia com seu ambiente externo.

7

Mecânica

A Mecânica clássica, também conhecida como Mecânica Newtoniana, é a parte da Mecânica que descreve o movimento de objetos macroscópicos em velocidades muito menores que a velocidade da luz. Ela é baseada nas leis de Newton, que descrevem a relação entre a força aplicada em um objeto e sua interferência resultante.

8

Mecânica Newtoniana

Baseia-se nas leis de Newton, que descrevem a relação entre a força aplicada em um objeto e sua interferência resultante.

9

Leis de Newton

Força e movimento em ação

10

PRIMEIRA LEI DE NEWTON

I.

Lei da Inércia

Inercia

Força

Um corpo em repouso ou em MRU tende a permanecer em seu estado de repouso ou de MRU a menos que uma força resultante não nula seja aplicada sobre ele.

11

1.

Lei da Inércia

A bola de golfe permanecerá em repouso até que o jogador a acerte.

12

SEGUNDA LEI DE NEWTON

2.

Princípio Fundamental da Dinâmica

Afirma que a aceleração de um objeto com massa constante é proporcional à força resultante de todas as forças aplicadas sobre ele e tem o mesmo sentido que a força resultante.

13

2.

Princípio Fundamental da Dinâmica

O carro acelera para frente pois a força do seu motor é maior que o atrito na estrada.

14

TERCEIRA LEI DE NEWTON

3.

Lei da Ação e Reação

Toda força de ação tem uma força de reação de igual intensidade e direção mas em sentido oposto.

15

3.

Lei da Ação e Reação

Em um lançamento de foguete, a espaçonave exerce uma força para baixo, e a força de reação de mesma intensidade a empurra para cima.

16

POR HOJE É SÓ!

A força e o movimento estão relacionados porque o movimento é resultado da força.

O movimento de um objeto depende do equilíbrio das forças que atuam sobre ele.

As Leis de Newton são:

1. Lei da Inércia
2. Princípio Fundamental da Dinâmica
3. Lei da Ação e Reação

17

Encontro III

18

Energia

Na Física, um dos conceitos fundamentais é o que se refere a energia. Pode ser descrita como a capacidade de um sistema para realização de trabalho. Em outros termos, energia é a perícia de um sistema para produzir mudanças ou efeitos. Ela pode existir em diversas formas, como energia mecânica, energia térmica, energia elétrica, energia química, entre outras.

19

Energia Mecânica

A energia mecânica é uma forma de energia que está relacionada ao movimento e à posição dos objetos. Ela é a soma da energia cinética, que é a energia associada ao movimento dos objetos, e a energia potencial, que é a energia mantida nos objetos devido à sua posição ou configuração.

$$E_m = E_c + E_p$$

20

Energia Cinética

A energia cinética de um objeto é definida como a energia que um objeto possui devido ao seu movimento.

Ela depende da massa e da velocidade do objeto, podendo ser calculada pela equação

$$E_c = \frac{1}{2}mv^2$$

onde m é a massa do objeto e v é o módulo da sua velocidade.

A energia cinética pode ser transformada em outras formas de energia, como a energia térmica, quando um objeto em movimento é freado ou colide com outro objeto.

21

Energia Potencial

A energia potencial, por sua vez, é a energia que um objeto possui devido à sua posição ou configuração em um campo de forças. Ela pode ser gravitacional, elástica, elétrica, magnética, entre outras.

A potencial gravitacional, por exemplo, é a energia armazenada por um objeto devido à sua posição em relação à Terra. Ela é representada pela equação

$$E_{p_g} = mgh$$

onde m é a massa do objeto, g é a aproximação da gravidade e h é a altura do objeto em relação a um ponto de referência.

Já a potencial elástica é a energia armazenada em uma mola ou em um objeto deformável, quando ele é esticado ou comprimido.

$$E_{p_e} = \frac{1}{2}kx^2$$

onde é a constante elástica do objeto deformável e x é o deslocamento.

22

Energia Mecânica

$$E_m = E_c + E_p$$

$$E_c = \frac{1}{2}mv^2$$ Energia Cinética

$$E_{p_g} = mgh$$ Energia Potencial Gravitacional

$$E_{p_e} = \frac{1}{2}kx^2$$ Energia Potencial Elástica

23

Conservação da Energia Mecânica

A energia não se cria e não se destrói, apenas se transfere ou se transforma em outro tipo de energia, em quantidade iguais.

24

Conservação da Energia Mecânica

Desprezando as forças dissipativas, pode-se afirmar que a energia mecânica se conserva. Nesse caso o sistema é conservativo, isto é somente as forças conservativas atuam na realização de trabalho.

Em um sistema conservativo a energia mecânica total permanece constante qualquer que seja a transformação do sistema.

$$E_{M_i} = E_{M_f}$$

25

POR HOJE É SÓ!

Na ausência de forças dissipativas, a energia mecânica é conservada.

$$E_m = E_c + E_p$$

$$E_c = \frac{1}{2}mv^2$$ Energia Cinética

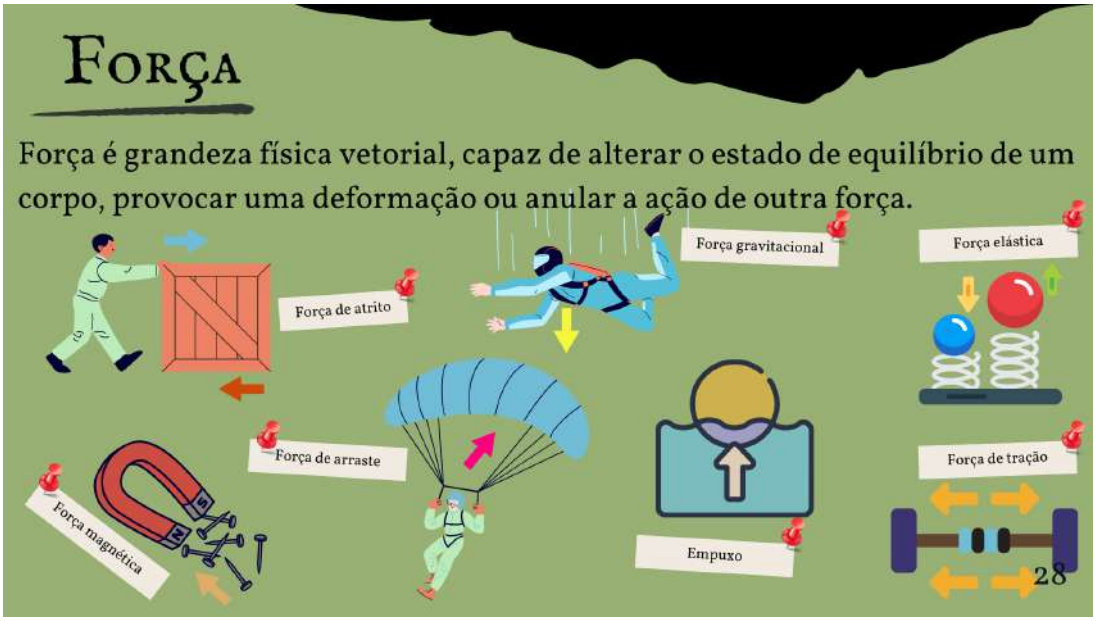
$$E_{p_g} = mgh$$ Energia Potencial Gravitacional

$$E_{p_e} = \frac{1}{2}kx^2$$ Energia Potencial Elástica

$$E_{M_i} = E_{M_f}$$ Conservação da Energia Mecânica

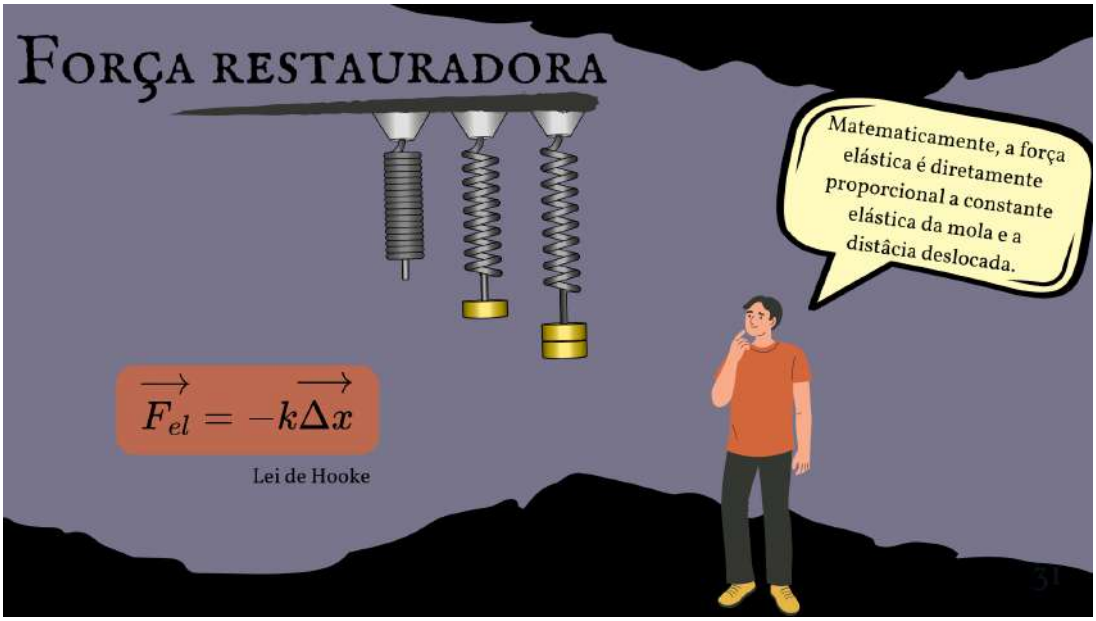
26

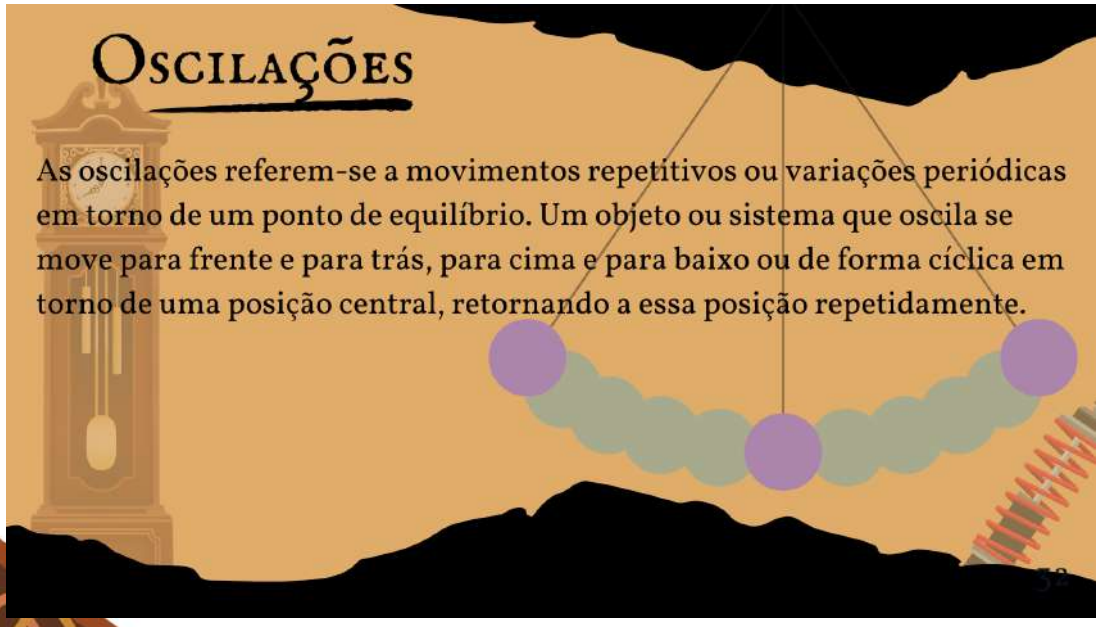

ENCONTRO IV
27


FORÇA
Força é grandeza física vetorial, capaz de alterar o estado de equilíbrio de um corpo, provocar uma deformação ou anular a ação de outra força.
Força de atrito
Força gravitacional
Força elástica
Força de arraste
Força magnética
Empuxo
Força de tração
28


FORÇA RESTAURADORA
A força restauradora surge sempre no sentido de restaurar a posição de equilíbrio do objeto.


Força restauradora
Nesse caso, por exemplo, a força restauradora corresponde a uma força elástica.


Força restauradora
Matematicamente, a força elástica é diretamente proporcional a constante elástica da mola e a distâcia deslocada.
$\overrightarrow{F_{el}} = -k\overrightarrow{\Delta x}$
Lei de Hooke


Oscilações
As oscilações referem-se a movimentos repetitivos ou variações periódicas em torno de um ponto de equilíbrio. Um objeto ou sistema que oscila se move para frente e para trás, para cima e para baixo ou de forma cíclica em torno de uma posição central, retornando a essa posição repetidamente.


CARACTERÍSTICAS
DAS OSCILAÇÕES
θ
AMPLITUDE
É a distância máxima que um objeto se afasta do ponto de equilíbrio.
33


CARACTERÍSTICAS
DAS OSCILAÇÕES
PERÍODO
O período de uma oscilação é o tempo necessário para que um ciclo completo seja concluído, ou seja, para que o objeto ou sistema volte à posição inicial.


CARACTERÍSTICAS
DAS OSCILAÇÕES
FREQUÊNCIA
A frequência é o número de ciclos completos que ocorrem em uma unidade de tempo.


CARACTERÍSTICAS
DAS OSCILAÇÕES
COMPRIMENTO DE ONDA
E o espaço que a onda percorre para completar uma oscilação (um ciclo).
λ


POR HOJE É SÓ!
A força restauradora age no intuito de restaurar a posição de equilíbrio do objeto.
$\overrightarrow{F_{el}} = -k\overrightarrow{\Delta x}$
As oscilações referem-se a movimentos repetitivos ou variações periódicas em torno de um ponto de equilíbrio.
PERÍODO
AMPLITUDE
FREQUÊNCIA
COMPRIMENTO DE ONDA
37


ENCONTRO V
38

Movimento harmônico simples

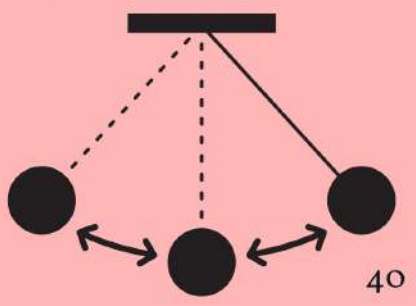
O movimento harmônico simples (MHS) é aquele em que um corpo oscila em torno de uma posição de equilíbrio devido à ação de uma força restauradora, cuja natureza pode ser elástica, gravitacional, elétrica, entre outras.

No MHS, não há forças dissipativas, como as forças de atrito e arraste, e, por isso, a energia mecânica total do sistema é conservada.

39

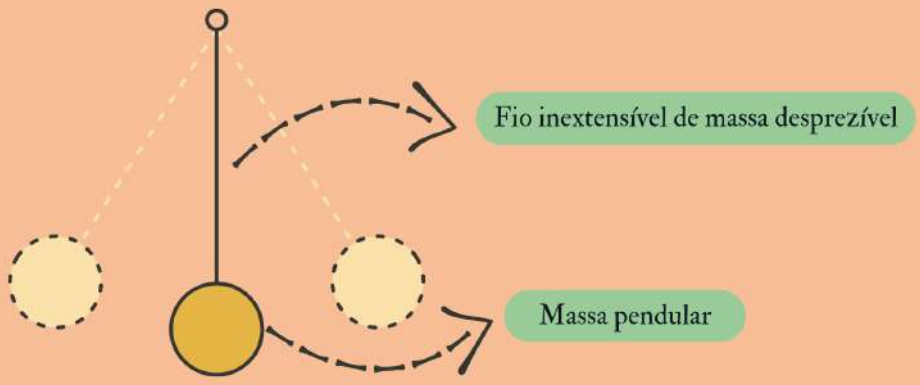
Pendulo simples

Um pêndulo simples é um dispositivo mecânico composto por uma massa pontual (ou quase pontual) chamada de "massa pendular" ou "massa suspensa" e um fio inextensível e leve, preso em um ponto fixo.



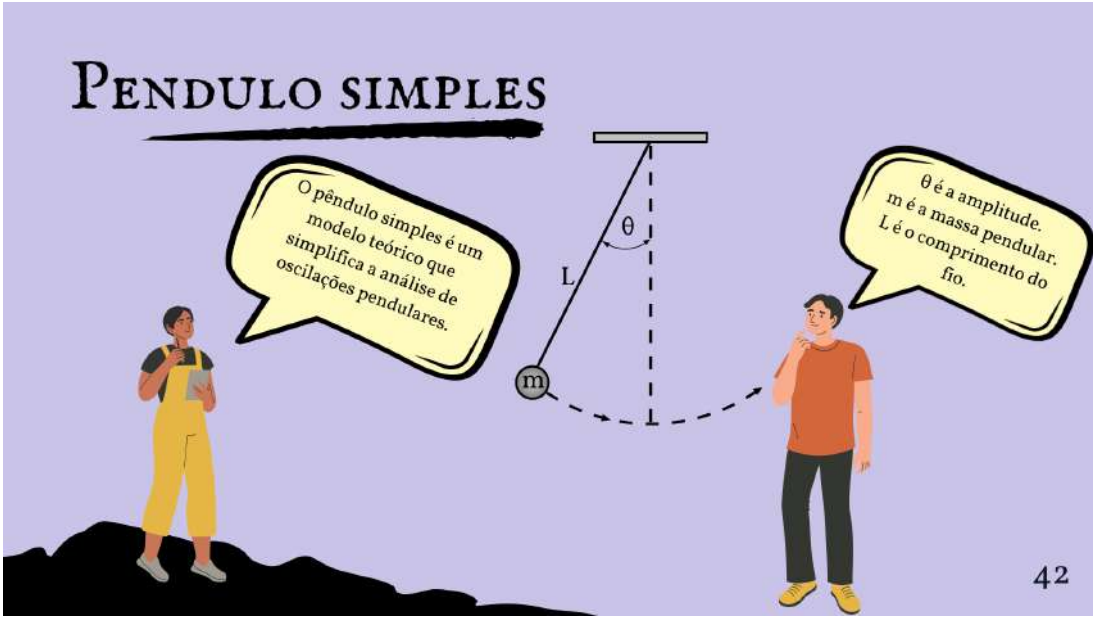
40

Pendulo simples



41

PENDULO SIMPLES



42

FORÇAS PRESENTES NO PENDULO SIMPLES

A força restauradora no pêndulo simples é a componente da força peso que tangencia a trajetória.

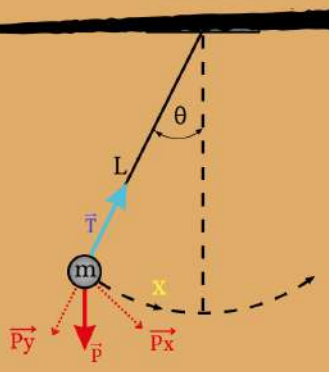
$$\overrightarrow{F_\theta} = \overrightarrow{P_x} = |\overrightarrow{P}| \sin\theta \qquad (1)$$

Sua intensidade pode ser escrita como

$$F_\theta = -mg\sin\theta \qquad (2)$$

Considerando que para pequenos ângulos $\sin\theta \approx \theta$, então:

$$F_\theta = -mg\theta \qquad (3)$$



43

FORÇAS PRESENTES NO PENDULO SIMPLES

Do comprimento do arco S tem-se que

$$x = L\theta$$

ou seja,

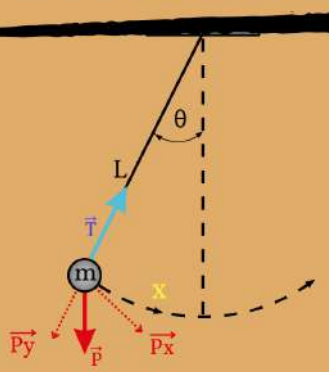
$$\theta = \frac{x}{L} \qquad (4)$$

Substituindo a equação 4 na equação 3, Pode-se notar que:

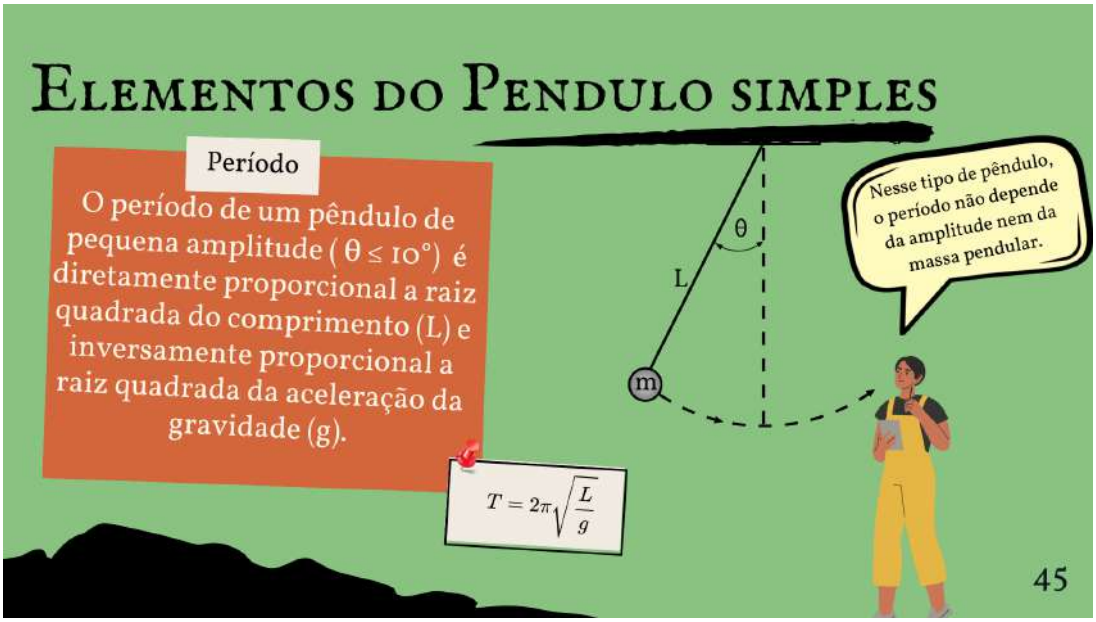
$$F_\theta = -mg\frac{x}{L} \qquad (5)$$

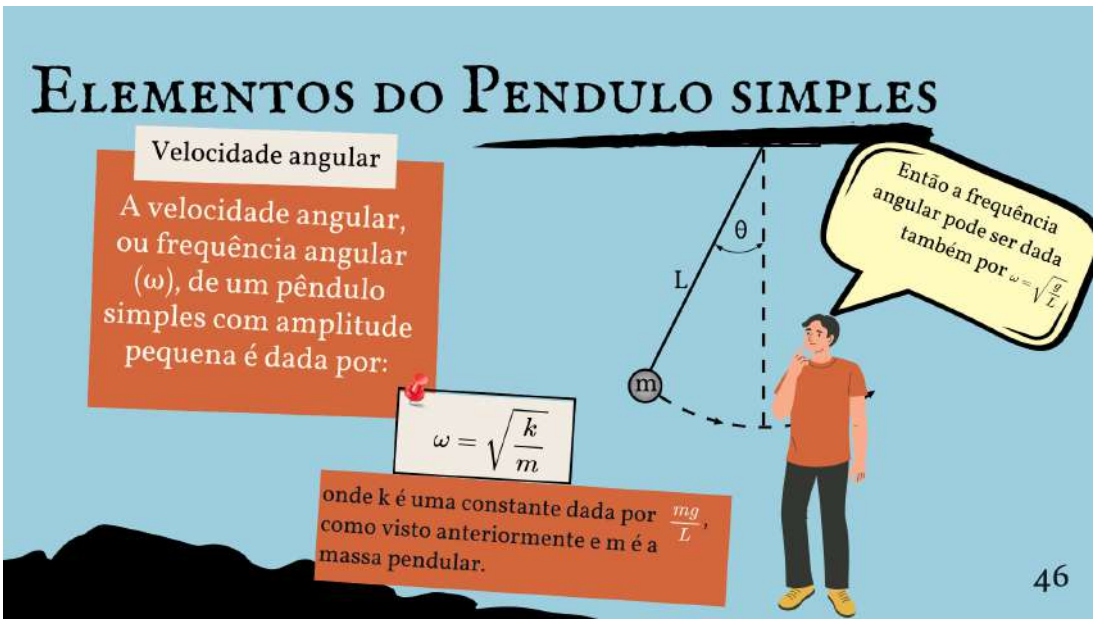
Fazendo , $k = \frac{mg}{L}$, chega-se à forma conhecida da intensidade da força restauradora dada pela lei de Hooke.

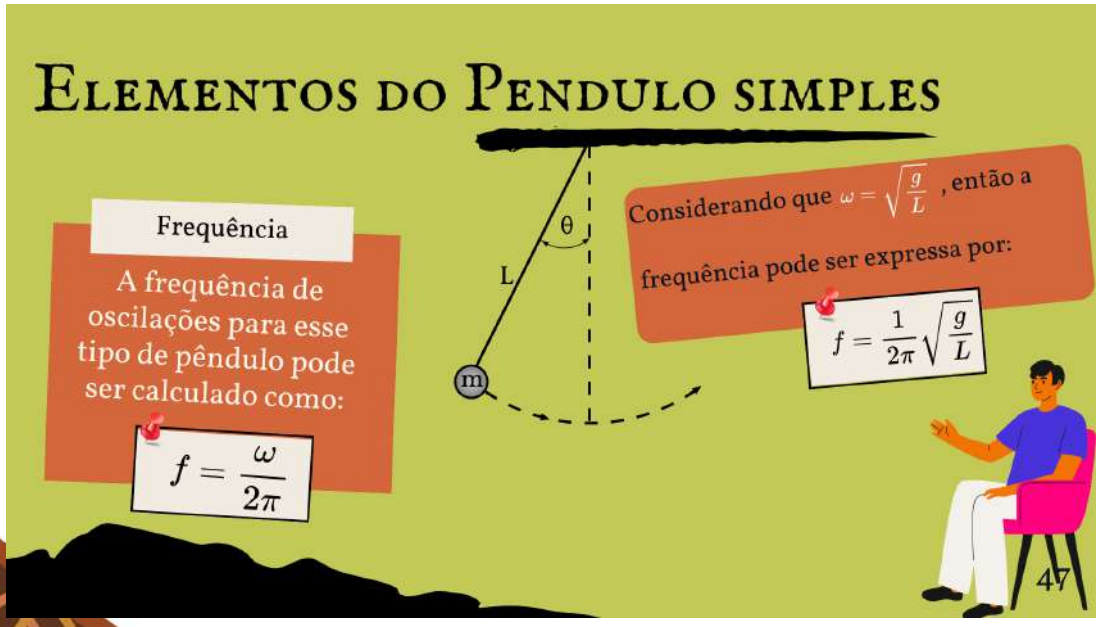
$$F = -kx \qquad (6)$$

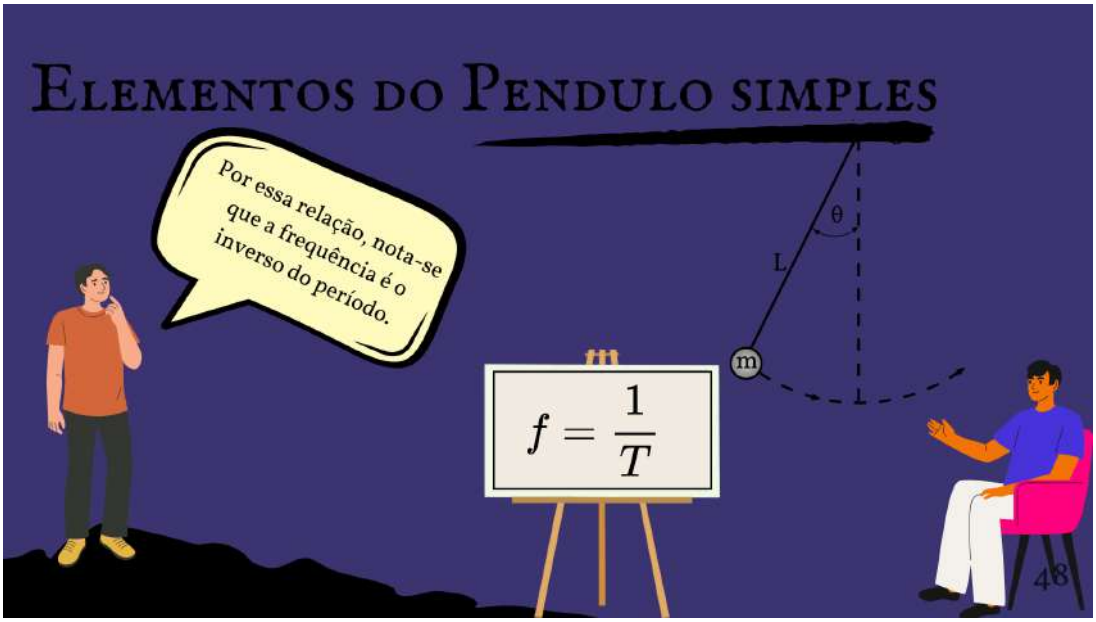


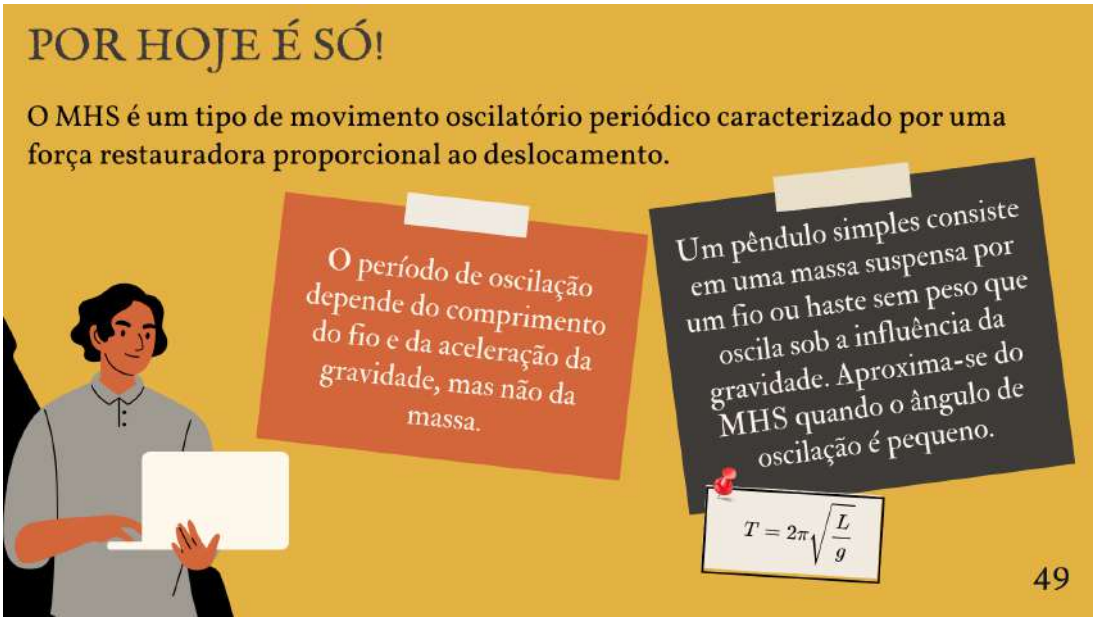
44


Elementos do Pendulo simples
Período
O período de um pêndulo de pequena amplitude (θ ≤ 10°) é diretamente proporcional a raiz quadrada do comprimento (L) e inversamente proporcional a raiz quadrada da aceleração da gravidade (g).
$T = 2\pi\sqrt{\frac{L}{g}}$
θ
L
m
Nesse tipo de pêndulo, o período não depende da amplitude nem da massa pendular.
45


Elementos do Pendulo simples
Velocidade angular
A velocidade angular, ou frequência angular (ω), de um pêndulo simples com amplitude pequena é dada por:
$\omega = \sqrt{\frac{k}{m}}$
onde k é uma constante dada por $\frac{mg}{L}$, como visto anteriormente e m é a massa pendular.
θ
L
m
Então a frequência angular pode ser dada também por $\omega = \sqrt{\frac{g}{L}}$
46


Elementos do Pendulo simples
Frequência
A frequência de oscilações para esse tipo de pêndulo pode ser calculado como:
$f = \frac{\omega}{2\pi}$
θ
L
m
Considerando que $\omega = \sqrt{\frac{g}{L}}$, então a frequência pode ser expressa por:
$f = \frac{1}{2\pi}\sqrt{\frac{g}{L}}$
47

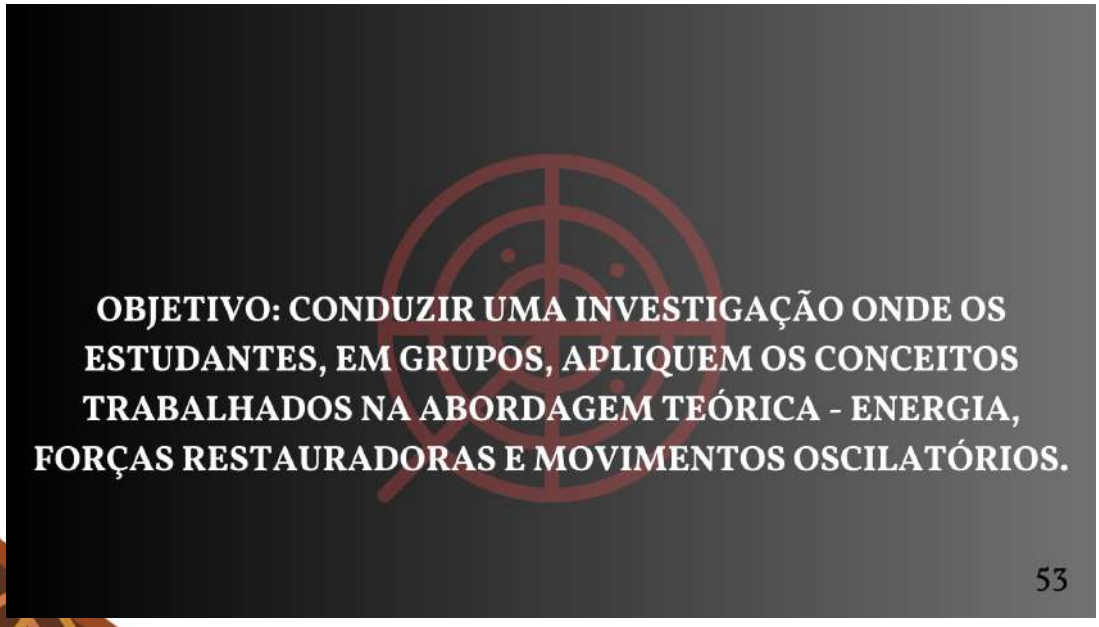

Elementos do Pendulo simples
Por essa relação, nota-se que a frequência é o inverso do período.
θ
L
m
$f = \frac{1}{T}$
48

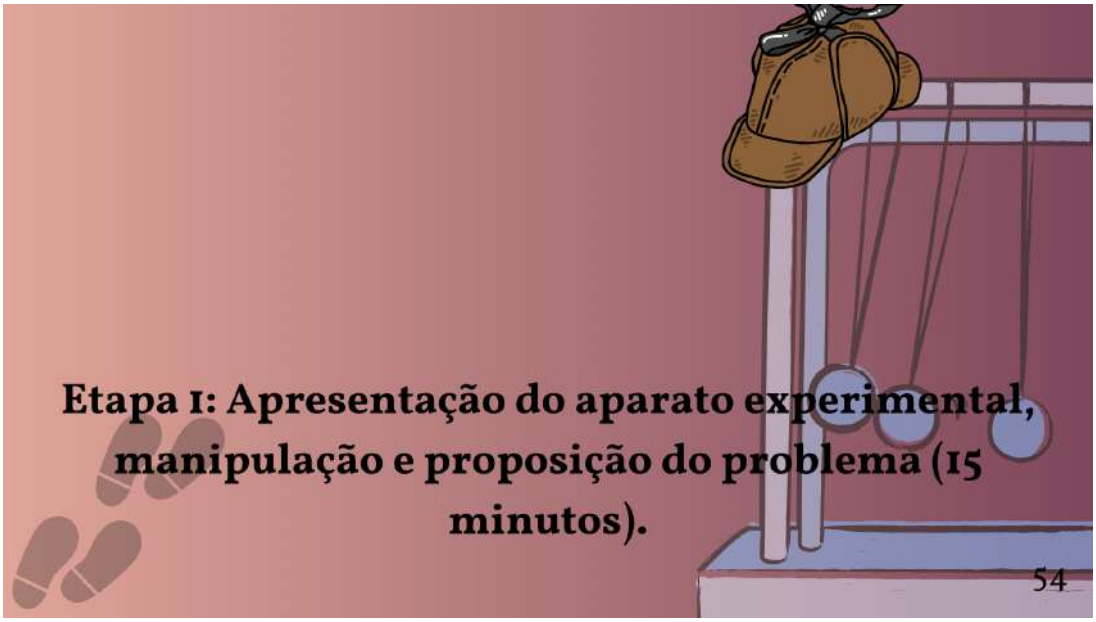

POR HOJE É SÓ!
O MHS é um tipo de movimento oscilatório periódico caracterizado por uma força restauradora proporcional ao deslocamento.
O período de oscilação depende do comprimento do fio e da aceleração da gravidade, mas não da massa.
Um pêndulo simples consiste em uma massa suspensa por um fio ou haste sem peso que oscila sob a influência da gravidade. Aproxima-se do MHS quando o ângulo de oscilação é pequeno.
$T = 2\pi\sqrt{\frac{L}{g}}$
49


Encontro VI
50


APLICAÇÃO DA SEI
Essa metodologia pedagógica busca estimular a aprendizagem ativa e o desenvolvimento de habilidades científicas nos alunos através de investigações e experimentos práticos realizados em sala de aula.
51


SEQUÊNCIA DE ENSINO INVESTIGATIVO
52


OBJETIVO: CONDUZIR UMA INVESTIGAÇÃO ONDE OS ESTUDANTES, EM GRUPOS, APLIQUEM OS CONCEITOS TRABALHADOS NA ABORDAGEM TEÓRICA - ENERGIA, FORÇAS RESTAURADORAS E MOVIMENTOS OSCILATÓRIOS.
53


Etapa 1: Apresentação do aparato experimental, manipulação e proposição do problema (15 minutos).
54


Etapa 2: Identificação e exploração das hipóteses e previsões dos estudantes (20 minutos).
55


Etapa 3: Elaboração de planos de ação e execução do planejado (35 minutos).
56


Etapa 4: Análise dos resultados e revisão das hipóteses (20 minutos).
57


Etapa 5: Discussão e conclusão (20 minutos).
58

APÊNDICE C2 - ROTEIRO DE ATIVIDADE PRÁTICA

ROTEIRO DE ATIVIDADE PRÁTICA DE MANUSEIO DO APARATO EXPERIMENTAL– INVESTIGAÇÃO GUIADA COM BASE NA SEI (SEQUÊNCIA DE ENSINO INVESTIGATIVO) - 1H50MIN.

Objetivo: Conduzir uma investigação onde os alunos, em grupos, apliquem os conceitos trabalhados na abordagem teórica - energia, forças restauradoras e movimentos oscilatórios.

ETAPAS DA ATIVIDADE:

Etapa 1: Apresentação do sistema de osciladores harmônicos acoplados, manipulação e proposição do problema (15 minutos).

- ✓ Proposição dos problemas investigativos:

  - Grupo 01: Como a energia mecânica total do sistema (cinética + potencial) se comporta durante o movimento oscilatório dos pêndulos acoplados?
  - Grupo 02: Como as energias cinética e potencial dos pêndulos variam durante o movimento oscilatório?
  - Grupo 03: Como a força elástica da mola varia em relação ao deslocamento causado pelo movimento dos pêndulos?
  - Grupo 04: De que forma o comprimento das hastes dos pêndulos influencia o período de oscilação do sistema?
  - Grupo 05: Como o movimento oscilatório de um dos pêndulos influencia o outro, devido ao acoplamento pela mola?
  - Grupo 06: Como a amplitude de oscilação dos pêndulos varia ao longo do tempo, e o que isso indica sobre a dissipação de energia no sistema?

Etapa 2: Identificação e exploração das hipóteses e previsões dos alunos (20 minutos)

- ✓ Nessa etapa será solicitado aos grupos que, com base na observação e manipulação inicial do material, façam previsões ou hipóteses sobre o comportamento das energias e das forças nos sistemas, de acordo com o problema recebido. O arquivo onde os estudantes descreverão suas hipóteses está disponível no apêndice A desse documento.

Etapa 3: Elaboração de planos de ação e execução do planejado (35 minutos)

- ✓ Após a etapa anterior, cada grupo escolhe uma ou mais hipóteses para investigar em detalhe. Eles devem refletir sobre como essas hipóteses podem ser testadas experimentalmente.
- ✓ Cada grupo elabora um plano detalhado de como vai testar suas hipóteses. O plano inclui: como medir as variáveis envolvidas (tempo de oscilação, deslocamento da mola, etc.); quais fatores serão mantidos constantes; como garantir a repetição do experimento para obter resultados mais precisos, entre outras possibilidades. Um modelo do plano a ser elaborado pelos grupos segue no apêndice 2 desse documento.
- ✓ Os grupos começam a realizar os experimentos planejados.

Etapa 4: Análise dos resultados e revisão das hipóteses (20 minutos)

- ✓ Após a coleta dos dados, os alunos organizam as informações e comparam os resultados experimentais com as suas previsões.

Etapa 5: Discussão e conclusão (20 minutos)

- ✓ Cada grupo apresenta os resultados de suas investigações explicando o problema recebido, a hipótese que investigaram, o procedimento experimental e as conclusões obtidas a partir dos dados.

APÊNDICE 1 – PROBLEMAS E HIPÓTESES

ETAPA 1: APRESENTAÇÃO DO APARATO EXPERIMENTAL, MANIPULAÇÃO E PROPOSIÇÃO DO PROBLEMA

Grupo: _____

Problema:

_____
_____.

ETAPA 2: IDENTIFICAÇÃO E EXPLORAÇÃO DAS HIPÓTESES E PREVISÕES DOS ALUNOS

Hipóteses do grupo:

_____
_____
_____
_____
_____
_____.

APÊNDICE 2 – MODELO DE PLANO DE AÇÃO

ETAPA 3: ELABORAÇÃO DE PLANO DE AÇÃO E EXECUÇÃO DO PLANEJADO

Plano de Ação do Grupo:

1- Identificação do grupo e do problema investigativo

Nome do grupo: _____

Problema investigativo: _____
_____

2- Hipóteses a serem testadas

Hipótese 1: _____

Hipótese 2 (*opcional*): _____

3- Variáveis envolvidas

Variável independente (*o que será alterado*): _____

Variável dependente (*o que será medido*): _____

Variáveis controladas (*o que será mantido constante*):_____

4- Métodos e instrumentos de medida

- Variável a medir:
- *Instrumento utilizado:* _____
- *Unidade de medida:* _____
- *Como será medida?* _____

5- Descrição do experimento

Passos para realizar o experimento (listados em sequência):
1- _____
2-_____
3-_____
4- _____
5- _____
Quantas vezes o experimento será repetido para garantir a precisão dos dados? _____

6- Registro dos dados (*Como os dados obtidos serão registrados? (ex.: tabela, gráficos, anotações*)
_____

7- Previsão dos resultados esperados (*baseada nas hipóteses*)
_____
_____

8- Observações adicionais
_____
_____

SOCIEDADE BRASILEIRA DE FÍSICA (SBF)
UNIVERSIDADE FEDERAL DO NORTE DO TOCANTINS (UFNT) - POLO 61
MESTRADO NACIONAL PROFISSIONAL EM ENSINO DE FÍSICA (MNPEF)
COORDENAÇÃO DE APERFEIÇOAMENTO DE PESSOAL DE NÍVEL SUPERIOR (CAPES)

2025