Universidade Federal Rural de Pernambuco
Pró-reitoria de Pesquisa e Pós-graduação
Departamento de física

Elaboração em Modelagem 3D de um Robô em Arduino: Aplicação Experimental de uma Partícula Ativa no Bilhar Limão

Ruan Victor Almeida Quirino
Dissertação de Mestrado

Recife - Pernambuco
9 de maio de 2025

Universidade Federal Rural de Pernambuco
Pró-reitoria de Pesquisa e Pós-graduação
Departamento de Física

Ruan Victor Almeida Quirino

Elaboração em Modelagem 3D de um Robô em Arduino: Aplicação Experimental de uma Partícula Ativa no Bilhar Limão

Trabalho apresentado ao Programa de Pós-Graduação em Física Aplicada do Departamento de Física da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do grau de Mestre em Física Aplicada.

Orientador: Prof. Dr. Tiago Araújo de Paula Lima
Coorientador: Dr. Francisco Carol Bonfim Leal

Recife - Pernambuco
9 de maio de 2025

Universidade Federal Rural de Pernambuco
Pró-reitoria de Pesquisa e Pós-graduação
Departamento de Física

Ruan Victor Almeida Quirino

Elaboração em Modelagem 3D de um Robô em Arduino: Aplicação Experimental de uma Partícula Ativa no Bilhar Limão

Dissertação de mestrado julgada adequada para obtenção do título de mestre em Física Aplicada e aprovada por unanimidade em 24 de Fevereiro de 2025 pela comissão examinadora.

Comissão Examinadora:

Dr. Tiago Araújo de Paula Lima
Orientador - UFRPE

Dr. Nathan Lima Pessoa
Examinador Interno - UFRPE

Dra. Leticia Ribeiro de Paiva
Examinador Externo - UFSJ

Recife - Pernambuco
9 de maio de 2025

Dados Internacionais de Catalogação na Publicação
Sistema Integrado de Bibliotecas da UFRPE
Bibliotecário(a): Auxiliadora Cunha – CRB-4 1134

Q8e Quirino, Ruan Victor Almeida.
Elaboração em modelagem 3D de um robô em Arduino: aplicação experimental de uma partícula ativa no Bilhar Limão / Ruan Victor Almeida Quirino. – Recife, 2025.
95 f.; il.

Orientador(a): Tiago Araújo de Paula Lima.
Co-orientador(a): Francisco Carol Bonfim Leal.

Dissertação (Mestrado) – Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Física Aplicada, Recife, BR-PE, 2025.

Inclui referências e apêndice(s).

1. Dinâmica Caótica. 2. Partículas Ativas. 3. Robô Arduino. 4. Bilhar Limão 5. Expoente de Lyapunov. I. Lima, Tiago Araújo de Paula, orient. II. Leal, Francisco Carol Bonfim, coorient. III. Título

CDD 621

“É a experiência que nos faz aprender todas as coisas.”
— L. Frank Baum.

Agradecimentos

Agradeço, antes de tudo, à minha companheira Amanda Santos. Nas horas mais difíceis, me ajudou a superar, meu anjo protetor, obrigado por estar ao meu lado, minha estrutura e meu guia é só você, agradeço pelo suporte emocional, pelo carinho e pelo incentivo constantes, sempre reconhecendo e valorizando meu trabalho. Sua presença foi essencial em cada etapa deste projeto, e sou imensamente grato pelo apoio incondicional durante todas as noites e madrugadas dedicadas a este trabalho.

Aos meus pais, Roseani e Ricardo, expresso minha profunda gratidão por me trazerem ao mundo, por todo o amor e dedicação na minha criação, e por sempre me incentivarem nos estudos, o que tornou possível a realização deste trabalho.

Aos meus gatos, Titinha, Simba e Baby, pelos miados de apoio nas madrugadas.

Agradeço à minha amiga Luana Hildever por quase uma década de amizade, pelas discussões e pelas lamentações compartilhadas. Hoje tenho a certeza de que posso dar minha opinião com a absoluta convicção de que serei julgado — e só tenho a agradecer por isso. Sou profundamente grato por sua recomendação sobre o Prof. Dr. Anderson Barbosa, cuja orientação ela elogiou e que, por sua influência, busquei como orientador. Graças a essa indicação, tive a oportunidade de ser introduzido ao caos, ao laboratório e ao experimento aqui apresentado.

Agradeço à Comitiva do LaSCoU (*Laboratório de Sistemas Complexos e Universalidade*). Assim como a Comitiva do Anel, de J.R.R. Tolkien, tem Gandalf, o mago, nós temos João, nosso mago particular, sempre nos guiando nas melhores aventuras pós-evento de física. Matheus, com a astúcia de um ladino e a versatilidade de um metamorfo, domina tanto as artes farmacológicas — do Imosec ao Gardenal — quanto as habilidades de um peixe Betta. José Dias, eterno questionador, cujas perguntas nos fazem refletir e buscar respostas além do óbvio... hãn? Marcos, do Piauí, mostrando-nos o que é coragem, sem precisar se justificar. Diego, o guardião dos segredos herbais. Mário, o homem dos teclados, nosso computeiro oficial. Ana Sofia, presença rara que apenas nosso mago tem o poder de convocar em momentos especiais. Luidje, nosso contador de historia, que até Sherlock Homes não conseguiria deduzir. Tiago, nosso vaqueiro b-boy nordestino favorito. Rodrigo, que transformou o calote em uma arte refinada. Pedro, que, como Wanderley, sempre traz ideias erradas. Antônio, eternamente em sua busca pela cerveja prometida, aguarda pacientemente o dia em que alguma alma a pagará. Laedson, o eloquente pai

de família. Matheus, vulgo Tanjiro, portador de uma arma secreta entalhada em papel, cujo poder permanece incontestável. Yago, cujo refinado interesse em vínculos matrimoniais estabelecidos não passa despercebido. Laíse e seu companheiro, sempre gerando entretenimento. Aos amigos do LaSCoU. Muito obrigado!

Agradeço aos professores do Departamento de Física que contribuíram significativamente para minha trajetória acadêmica. Em especial, ao Prof. Dr. Antonio Miranda, a quem tive a oportunidade de conhecer no final do ensino médio, juntamente com o projeto *Desvendando o Céu Austral*. Essa experiência despertou em mim o desejo de cursar Física, mesmo sem nunca ter tido uma aula da disciplina durante o ensino médio. Ao Prof. Dr. Jairo Rocha, por me mostrar, no início da graduação, a elegância da Física e me fazer ter a certeza de que escolhi o caminho certo. Ao Prof. Dr. Raul Montagne, pelos incentivos, pelas conversas inspiradoras e pela dedicação incansável no ensino. Ao Prof. Dr. Anderson Barbosa, por aceitar meu pedido de iniciação científica e me ensinar a importância dos gráficos. Ao Prof. Dr. Antonio Romaguera, por me ajudar a compreender os experimentos e mostrar o valor dos relatórios — um aprendizado que me salvou algumas vezes ao longo do mestrado. Ao meu Coorientador Dr. Francisco Carol, pelo suporte durante os experimentos.

E, especialmente, ao meu orientador, Prof. Dr. Tiago Araújo, cujas recomendações de livros sobre sistemas dinâmicos e caos me inspiraram a me dedicar e me aprofundar cada vez mais nessa área de estudo, pela qual me vi fascinado. Sou imensamente grato por sua paciência, dedicação e orientação exemplar ao longo desta trajetória acadêmica.

A todos, minha mais profunda gratidão!

Resumo

A construção de um sistema experimental versátil, integrando tecnologias como Arduino, modelagem e impressão 3D, além de processamento de imagens, mostrou-se eficaz para investigar a dinâmica de partículas ativas. O sistema permitiu explorar transições entre regimes dinâmicos regulares, mistos e caóticos, com análise quantitativa baseada em trajetórias e velocidades. Os resultados experimentais apresentaram boa concordância com previsões teóricas, validando a metodologia utilizada. Além disso, o sistema desenvolvido possui potencial de expansão para estudar novos cenários geométricos e os efeitos de perturbações externas, consolidando sua utilidade como ferramenta experimental para o estudo de sistemas dinâmicos complexos.

O formato das paredes do bilhar é determinado por um parâmetro $\gamma \in [0, 0{,}5]$. À medida que γ varia, a dinâmica do sistema transita de regular ($\gamma = 0$) para totalmente caótica ($\gamma = 0{,}5$), passando por regimes mistos para valores intermediários. O robô, em interação com o bilhar, fornece uma plataforma ideal para investigar essas transições dinâmicas.

A sensibilidade do robô às condições iniciais foi analisada por meio do expoente de Lyapunov λ, caracterizando a dinâmica observada na interação robô-ambiente. Inicialmente, explorou-se o expoente de Lyapunov no bilhar circular ($\gamma = 0$), onde o valor experimental obtido foi $\lambda_{\text{exp}} \simeq 0$, indicando a ausência de caos e servindo como calibração. Foram apresentadas medições de espaços de fases mistos ($0 < \gamma < 0{,}5$) em bilhares fora do escopo óptico, com abordagens numéricas complementando os resultados experimentais. Este estudo permitiu uma exploração detalhada da sensibilidade às condições iniciais em sistemas dinâmicos, destacando a característica marcante do caos em sistemas complexos.

Palavras-chave: Bilhar Experimental, Partículas Ativas, Dinâmica Caótica, Expoente de Lyapunov, Análise de Recorrência.

Abstract

The construction of a versatile experimental system, integrating technologies such as Arduino, modeling, 3D printing, and image processing, proved effective in investigating the dynamics of active particles. The system enabled the exploration of transitions between regular, mixed, and chaotic dynamic regimes, with quantitative analysis based on trajectories and velocities. Experimental results showed good agreement with theoretical predictions, validating the methodology used. Furthermore, the developed system has potential for expansion to study new geometric scenarios and the effects of external perturbations, consolidating its utility as an experimental tool for studying complex dynamic systems.

The shape of the billiard walls depends on a parameter $\gamma \in [0, 0.5]$. As γ varies, the system's dynamics transition from regular ($\gamma = 0$) to fully chaotic ($\gamma = 0.5$), passing through mixed regimes for intermediate values. The robot, interacting with the billiard, provides an ideal platform for investigating these dynamic transitions.

The robot's sensitivity to initial conditions was analyzed through the Lyapunov exponent λ, characterizing the observed dynamics in the robot-environment interaction. Initially, the Lyapunov exponent was explored in the circular billiard ($\gamma = 0$), where the experimental value obtained was $\lambda_{\text{exp}} \simeq 0$, indicating the absence of chaos and serving as a calibration. Measurements of mixed phase spaces ($0 < \gamma < 0.5$) were presented in billiards outside the optical scope, with numerical approaches complementing the experimental results. This study enabled a detailed exploration of sensitivity to initial conditions in dynamic systems, highlighting the remarkable feature of chaos in complex systems.

Keywords: Experimental Billiard, Active Particles, Chaos Dynamics, Lyapunov Exponent, Recurrence Analysis.

Sumário

Lista de Figuras

Lista de Tabelas

Acrônimos

Capítulo 1

Introdução

Este trabalho é uma continuação de uma pesquisa anterior, na qual foi desenvolvido um experimento para observar o comportamento caótico em sistemas de bilhar, utilizando uma montagem macroscópica. Nesse experimento, uma câmera digital registrou a evolução temporal da interação entre um robô e um bilhar tipo estádio de Bunimovich, com reflexão especular. A partir da série temporal obtida experimentalmente, foi calculado o Expoente de Lyapunov (EL) como função de um parâmetro geométrico, cujos resultados foram consistentes com as previsões teóricas. A superfície de seção de Poincaré foi determinada e sua sensibilidade às Condições Iniciais (CIs) foi analisada em função do tempo [1]. Partículas ativas continuamente injetam energia no sistema. Então, a área do conhecimento chamada de matéria ativa fala sobre entidades autopropelidas. O marco no estudo de sistemas ativos, é o estudo da interação das partículas com bordas. Essas interações têm aplicações práticas em sistemas biológicos e engenharia, tal como robôs. Tais resultados são motivação para exploração do papel do confinamento em novas configurações para partículas ativas, incluindo sistemas de bilhar [2, 3].

Na pesquisa atual, ampliamos o escopo da investigação ao explorar a interação entre um robô e o ambiente em um sistema de bilhar tipo Limão. Para isso, foi desenvolvido um robô em Arduino, que atua como uma partícula ativa no bilhar. As paredes do bilhar são ajustáveis conforme um parâmetro geométrico, $\gamma \in [0, 0{,}5]$, que permite observar mudanças dinâmicas desde um comportamento regular ($\gamma = 0$) até o regime totalmente caótico ($\gamma = 0{,}5$), passando por fases mistas em valores intermediários de γ. Usamos o EL para caracterizar a sensibilidade do robô às CIs.

1.1 Sistemas Dinâmicos

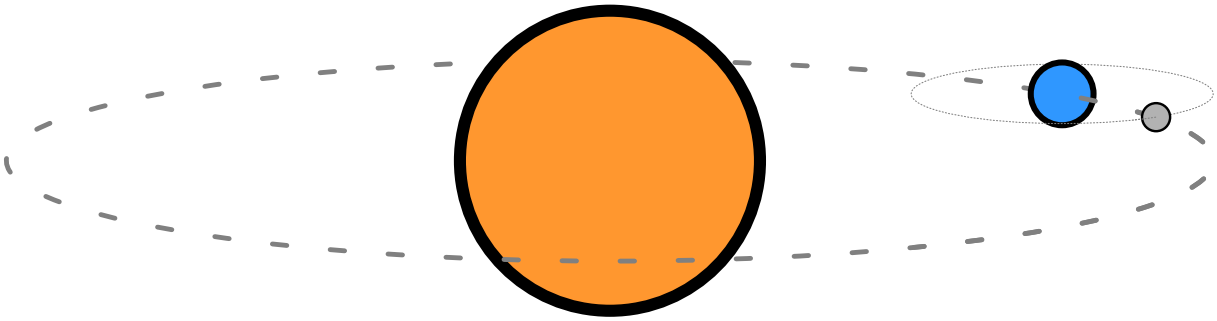

Figura 1.1: Representação qualitativa do sistema dinâmico *Sol-Terra-Lua*, conhecido historicamente como um exemplo do problema dos três corpos. Este sistema ilustra a complexidade intrínseca das interações gravitacionais entre três massas, onde a força mútua entre os corpos gera movimentos não lineares e acoplados.

Um sistema dinâmico é uma formulação matemática que descreve a evolução temporal de um conjunto de variáveis de estado. Para sistemas contínuos, a evolução é frequentemente descrita por equações diferenciais:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t), \tag{1.1}$$

onde $\mathbf{x}(t)$ é o vetor de estado e $\mathbf{f}$ é uma função vetorial que incorpora as leis físicas governantes. O estudo dos sistemas dinâmicos, iniciado por Isaac Newton no século XVII, constitui um pilar fundamental da física clássica. Newton, ao desenvolver o cálculo diferencial e integral, formulou as leis do movimento e da gravitação universal. Estas equações permitiram a solução analítica do problema dos dois corpos, descrevendo o movimento orbital elíptico previsto pelas leis de Kepler. Contudo, o problema dos três corpos revelou-se intratável analiticamente devido à sua natureza não-linear e acoplada:

$$\begin{cases} m_1 \dfrac{\mathrm{d}^2\mathbf{r}_1}{\mathrm{d}t^2} = G\dfrac{m_1 m_2(\mathbf{r}_2 - \mathbf{r}_1)}{|\mathbf{r}_2 - \mathbf{r}_1|^3} + G\dfrac{m_1 m_3(\mathbf{r}_3 - \mathbf{r}_1)}{|\mathbf{r}_3 - \mathbf{r}_1|^3}, \\ m_2 \dfrac{\mathrm{d}^2\mathbf{r}_2}{\mathrm{d}t^2} = G\dfrac{m_1 m_2(\mathbf{r}_1 - \mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|^3} + G\dfrac{m_2 m_3(\mathbf{r}_3 - \mathbf{r}_2)}{|\mathbf{r}_3 - \mathbf{r}_2|^3}, \\ m_3 \dfrac{\mathrm{d}^2\mathbf{r}_3}{\mathrm{d}t^2} = G\dfrac{m_1 m_3(\mathbf{r}_1 - \mathbf{r}_3)}{|\mathbf{r}_1 - \mathbf{r}_3|^3} + G\dfrac{m_2 m_3(\mathbf{r}_2 - \mathbf{r}_3)}{|\mathbf{r}_2 - \mathbf{r}_3|^3}. \end{cases} \tag{1.2}$$

Onde $\mathbf{r}_i$ são os vetores posição, m_i as massas dos corpos e G é a constante universal da gravitação. Newton foi o pioneiro na abordagem do problema e obteve resultados em ambas as frentes de investigação. Depois de analisar o movimento da Lua ao redor da Terra e derivar uma órbita elíptica, ele investigou o impacto do Sol sobre essa órbita,

considerando as variações induzidas (Fig. 1.1). Contudo, os cálculos se mostraram extremamente desafiadores, e sua estimativa para o movimento das apsides da Lua resultou em um valor cerca de metade do observado, algo que ele resumiu nas edições posteriores do *Principia* com a simples observação: "A ápse da Lua é cerca de duas vezes mais rápida". Os obstáculos que enfrentou foram tantos que ele comentou ao astrônomo John Machin: "... minha cabeça nunca doeu, exceto com meus estudos sobre a Lua" [4, 5].

Jean le Rond d'Alembert e Alexis Clairaut desenvolveram uma rivalidade de longa data, tentando analisar o problema dos três corpos com certo grau de generalidade. Eles submeteram suas primeiras análises concorrentes à *Academie Royale des Sciences* em 1747. Foi em conexão com suas pesquisas, em Paris durante a década de 1740, que o termo "problema dos três corpos" (em francês: *Problème des trois corps*) começou a ser comumente utilizado. Um relato publicado em 1761 por d'Alembert indica que o nome foi usado pela primeira vez em 1747 [6].

No final do século XIX, Henri Poincaré revolucionou o estudo dos sistemas dinâmicos ao introduzir métodos geométricos. Estes métodos focam nas propriedades estruturais das soluções, em vez de buscar soluções analíticas exatas. Os métodos geométricos envolvem a representação visual e análise das trajetórias nos Espaços de Fase (EFs), incluindo técnicas como seções de Poincaré. Estes permitem identificar estruturas como pontos fixos, ciclos limite e variedades estáveis/instáveis [7]. Poincaré demonstrou que pequenas perturbações nas CIs podem levar a divergências significativas nas trajetórias, um fenômeno hoje conhecido como caos determinístico. As equações de movimento podem ser lineares ou não-lineares. Para sistemas lineares, o princípio de superposição é válido:

$$f(\alpha x_1 + \beta x_2) = \alpha f(x_1) + \beta f(x_2). \tag{1.3}$$

Sistemas não-lineares, como o pêndulo simples (cuja a equação de movimento é dada por: $\ddot{\theta} + \frac{g}{l}\,\mathrm{sen}(\theta) = 0$), não obedecem a este princípio e podem exibir comportamentos complexos [8]. A abordagem de Poincaré estabeleceu as bases para a teoria moderna dos sistemas dinâmicos, possibilitando o estudo de sistemas onde soluções analíticas são impossíveis ou impraticáveis, revelando características globais do comportamento do sistema e fornecendo informação sobre estabilidade e comportamento a longo prazo.

O Espaço de Fase (EF) é definido pelo conjunto de todas as possíveis configurações do sistema, pode exibir três tipos principais de comportamento:

1. **Periódico**: $\mathbf{x}(t+T) = \mathbf{x}(t)$ para algum período T;
2. **Quase-periódico**: $\mathbf{x}(t) = \mathbf{f}(\omega_1 t, \omega_2 t, \ldots, \omega_n t)$, onde $\mathbf{f}$ é uma função periódica em cada argumento e as frequências ω_i são incomensuráveis;
3. **Caótico**: caracterizado por sensibilidade exponencial às CIs, quantificada pelo EL $\lambda > 0$.

	Periódico	**Quase-Periódico**	**Misto**	**Caótico**
Comportamento	Repetitivo e previsível	Regular, mas sem repetição exata	Combinação de regiões periódicas e caóticas	Irregular, imprevisível
Órbitas no espaço de fase	Fechadas	Em torno de toros	Variável (dependendo da condição inicial)	Preenchem áreas
Previsibilidade	Alta	Alta	Variável (dependendo da condição inicial)	Baixa (sensível a condições iniciais)
Exemplo	Pêndulo simples, órbita planetária (2 corpos)	Planetas com perturbações, osciladores acoplados	Bilhar cogumelo, Mapa Padrão, Bilhar Limão	Pêndulo duplo, sistema de três corpos, bilhar de Bunimovich

Tabela 1.1: Tabela comparativa destacando as principais características de sistemas dinâmicos classificados como periódicos, quase-periódicos, mistos e caóticos. São apresentados aspectos qualitativos como comportamento, tipos de órbitas no espaço de fase, previsibilidade e exemplos típicos, ilustrando a diversidade de dinâmicas possíveis em sistemas físicos.

1.2 Dinâmica Caótica e Expoente de Lyapunov

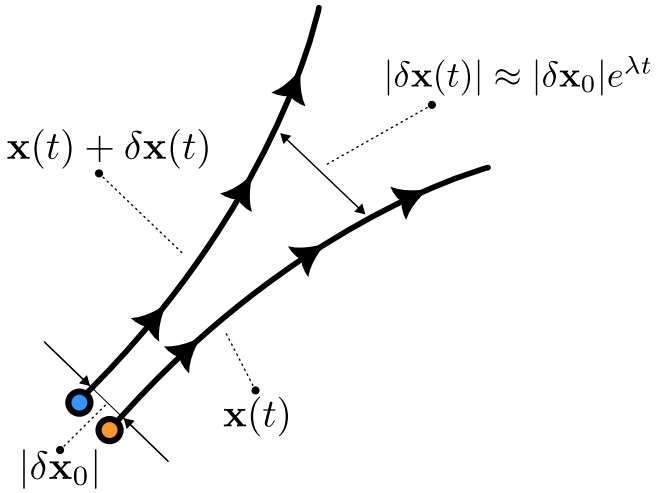

Figura 1.2: Representação da separação exponencial de trajetórias $|\delta\mathbf{x}(t)| \approx |\delta\mathbf{x}_0|e^{\lambda t}$, que caracteriza sistemas dinâmicos caóticos por meio do EL.

A separação exponencial de órbitas com Condições Iniciais (CIs) próximas no Espaço de Fase (EF) leva ao conceito de EL. Este expoente, quando positivo, identifica a dinâmica como caótica, um valor positivo de λ indica que as trajetórias se separam exponencialmente como ilustrado na Fig. 1.2, caracterizando o caos [9, 10]. Formalmente, dado um pequeno desvio inicial $\delta\mathbf{x}_0$, o desvio $\delta\mathbf{x}(t)$ após um tempo t evolui aproximadamente como:

$$|\delta\mathbf{x}(t)| \approx |\delta\mathbf{x}_0|e^{\lambda t}, \tag{1.4}$$

onde o λ é o maior EL. Os indicadores de sensibilidade exponencial aparecem na literatura com diversos nomes: número de Lyapunov, número característico de Lyapunov, expoente característico de Lyapunov ou Expoente de Lyapunov (EL). Estes indicadores são conceitualmente fundamentados em uma série de teoremas demonstrados por matemáticos como Valery Oseledec e Yakov Pesin, e foram bastante divulgados por Giancarlo Benettin, Luigi Galgani e Jean-Marie Strelcyn [11]. Como o caos se manifesta como divergência assintoticamente exponencial de duas trajetórias vizinhas, para estudá-lo podemos construir soluções das equações diferenciais do movimento que comecem em dois pontos próximos, $\mathbf{x}$ e $\mathbf{x}'$, e acompanhá-las quando estamos em uma região caótica. A tendência dessas soluções é se afastarem exponencialmente (Fig. 1.2).

Para medir a divergência exponencial, Benettin *et al.* [11] indicam a necessidade de proceder normalizações da distância para evitar que as duas soluções usadas se afastem demasiadamente. Vamos chamar de d_0 a distância (no EF) entre $\mathbf{x}(0)$ e $\mathbf{x}'(0)$. Após um tempo τ, interrompe-se o cálculo. Neste momento, a distância entre os dois pontos é d_1. Efetua-se uma renormalização que consiste em tomar como ponto de partida para o cálculo seguinte, ao invés de $\mathbf{x}'(\tau)$, um ponto $\mathbf{x}''(\tau)$ que se situa sobre uma linha unindo $\mathbf{x}(\tau)$ ao ponto $\mathbf{x}'(\tau)$, cuja distância ao ponto $\mathbf{x}(\tau)$ é a distância inicial d_0. E assim procede-se um número de vezes n suficientemente grande para que se possa estimar a divergência exponencial, que será dada pela fórmula:

$$\lambda(\tau, \mathbf{x}, d_0) = \frac{1}{n\tau} \ln\left(\frac{d_n}{d_0}\right). \tag{1.5}$$

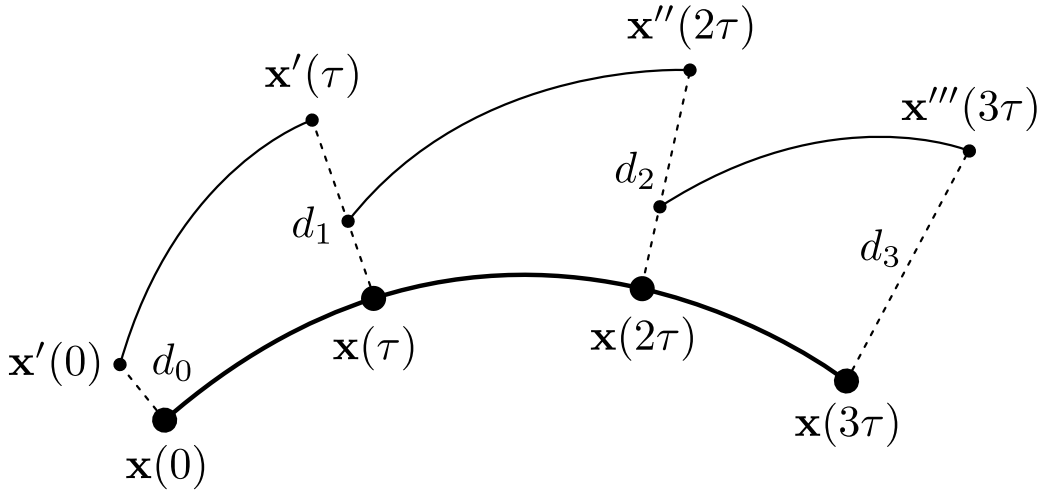

Figura 1.3: Processo de renormalização.

A não linearidade é necessária para a existência do caos; entretanto, nem todos os sistemas não lineares apresentam dinâmica caótica. Exemplos de sistemas caóticos incluem o pêndulo duplo [12], o atrator de Lorenz [13].

O conhecimento das leis que descrevem a dinâmica dos sistemas permite estabelecer uma distinção clara entre determinismo e não determinismo. O determinismo está associado às leis que descrevem a dinâmica do sistema, as quais são conhecidas explicitamente e não contêm termos associados à estocasticidade. Em um sistema caótico, as leis que descrevem o movimento são conhecidas e não envolvem termos estocásticos. No entanto, a natureza não linear das equações que descrevem o movimento e a característica de afastamento exponencial no tempo a partir de duas CIs próximas impedem o conhecimento preciso de estados futuros, mesmo que o sistema seja determinístico [14].

1.2.1 Sistemas Caóticos de Tempo Discreto

Sistemas caóticos de tempo discreto são gerados pela iteração de uma função não linear adequada, com a característica de que o estado atual do sistema é influenciado pelos estados anteriores. Esses sistemas podem ser aplicados diretamente em ambientes digitais, sem a necessidade de algoritmos de discretização, ao contrário dos sistemas de tempo contínuo. A literatura apresenta diversos exemplos de sistemas caóticos de tempo discreto, tanto unidimensionais quanto bidimensionais. Entre os sistemas unidimensionais, destacam-se o Mapa Logístico [15] e o Mapa Cúbico [16]. Já para os sistemas bidimensionais, são comumente citados o Mapa de Henon [17], Mapa Padrão e os Bilhares, que exploram a complexidade das interações em duas dimensões.

1.2.2 Mapa Padrão

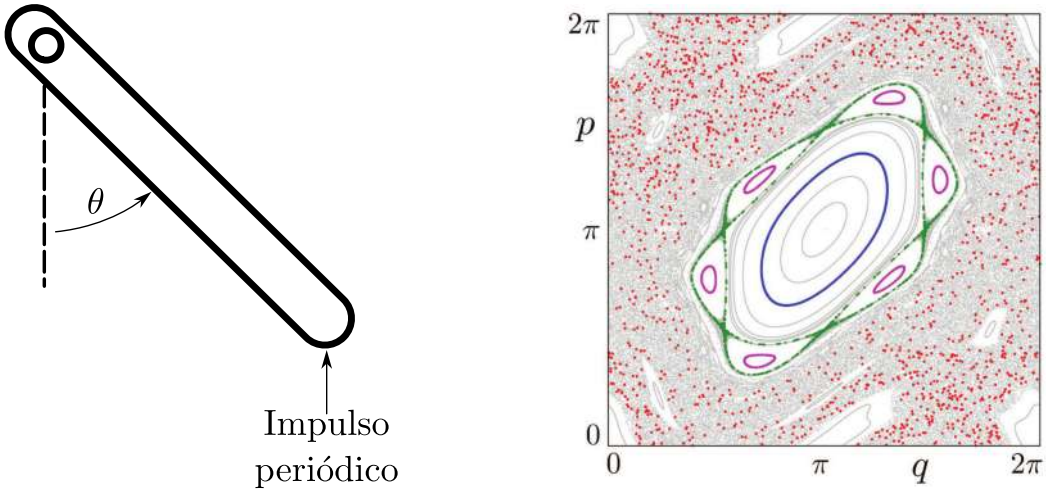

Figura 1.4: **Esquerda:** Representação esquemática do *rotor chutado*, adaptada de [18]. **Direita:** EF para $K = 0{,}001$, ilustrando diferentes trajetórias em cores, conforme apresentado em [19].

O mapa padrão, também conhecido como mapa de Chirikov-Taylor, é um exemplo de sistema dinâmico discreto. Introduzido por Boris Chirikov em 1969 [20], este mapa serve como um modelo simples, porém essencial, para explorar a transição da regularidade para o caos em sistemas não lineares conservativos. Modelando o comportamento de um *rotor chutado* — um sistema mecânico onde uma haste gira livremente em torno de um eixo e recebe impulsos periódicos em sua extremidade — o mapa padrão captura a essência da dinâmica complexa em sistemas físicos reais. As equações que definem o mapa são:

$$\theta_{n+1} = (\theta_n + p_n) \pmod{2\pi}, \tag{1.6}$$

$$p_{n+1} = p_n + K \operatorname{sen}(\theta_{n+1}) \pmod{2\pi}, \tag{1.7}$$

onde θ_n representa a posição angular no instante discreto n, e p_n é o momento angular correspondente, ambos considerados módulo 2π devido à periodicidade angular. O parâmetro K é uma constante real que controla a intensidade da não linearidade e dos *chutes* aplicados ao sistema. Uma característica do mapa padrão é que ele preserva a área, ou seja, possui jacobiano unitário. Isso reflete a natureza conservativa do sistema hamiltoniano original e implica na conservação do volume no EF sob a evolução dinâmica, uma propriedade fundamental em sistemas sem dissipação. Para valores pequenos do parâmetro K (isto é, $K \ll 1$), o sistema está próximo do comportamento integrável, e as trajetórias no EF são confinadas a curvas invariantes toroidais, conforme previsto pelo Teorema Kolmogorov-Arnold-Moser (KAM) [21–23]. Nesse regime, as órbitas são periódicas ou quase periódicas, e o movimento é regular, conforme mostram as curvas azul e roxas na Fig. 1.4. À medida que K aumenta, ressonâncias não lineares causam a quebra dessas curvas invariantes, levando ao surgimento de ilhas de estabilidades e regiões caóticas. O valor crítico $K_c \simeq 0{,}9716$ marca a transição para o caos global, onde ocorre a sobreposição de ressonâncias, e a dinâmica torna-se caótica em quase todo oEF [18, 20].

1.3 Mapa de Poincaré

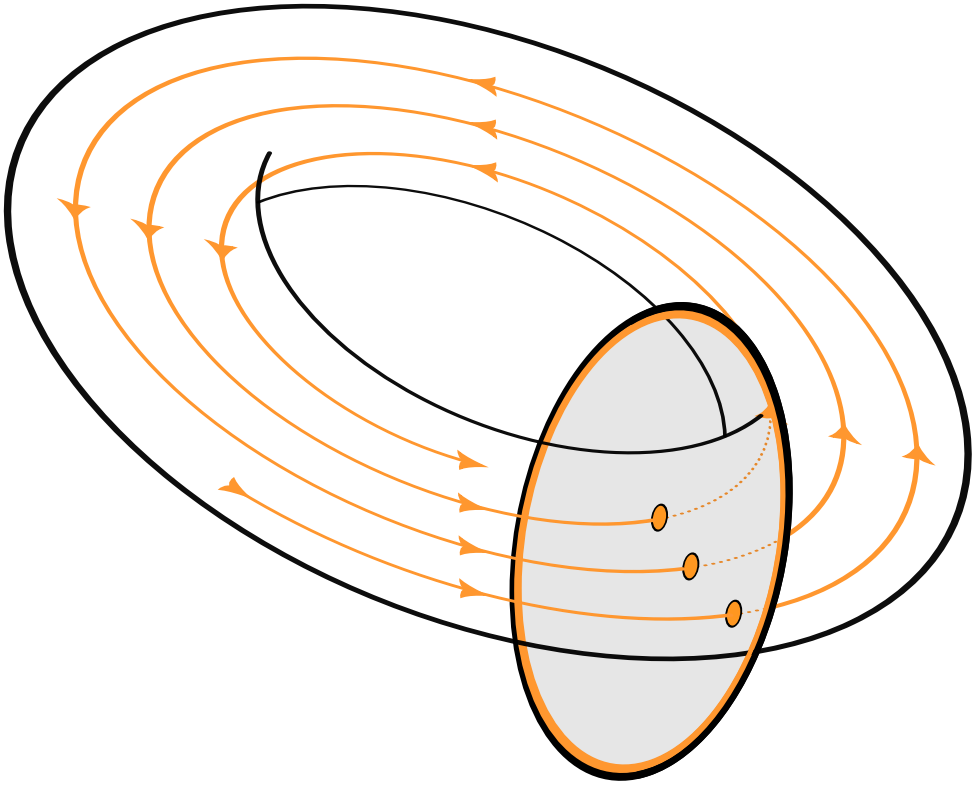

Figura 1.5: Representação da seção de Poincaré, mostrando os pontos de interseção do fluxo de soluções com a superfície no EF.

O fluxo de soluções é um conceito fundamental na análise de sistemas dinâmicos descritos por equações diferenciais. Ao resolver essas equações, sejam lineares ou não lineares, obtemos um conjunto contínuo de soluções que representam a evolução temporal do sistema. Essas soluções são visualizadas no EF, um espaço multidimensional onde cada ponto representa um estado completo do sistema, incluindo todas as suas variáveis dinâmicas, como posições e velocidades. A dimensão N do fluxo de soluções corresponde ao número de variáveis necessárias para descrever completamente o estado do sistema. Por exemplo, no problema de três corpos, o sistema teria uma dimensão de $N = 18$ (três coordenadas de posição e três coordenadas de velocidade para cada corpo) [24]. Um sistema com uma única partícula em movimento bidimensional teria uma dimensão $N = 4$ (duas coordenadas de posição e duas de velocidade). Para simplificar a análise de sistemas dinâmicos complexos, é comum utilizar a técnica do Mapa de Poincaré (MP). Essa técnica envolve a interseção do fluxo de soluções com uma superfície específica no EF, chamada seção de Poincaré. Cada vez que a trajetória do sistema cruza essa superfície, um ponto é marcado (Fig. 1.5). A coleção desses pontos forma o chamado MP. O uso deste mapa

é vantajoso porque reduz a dimensão do problema em pelo menos uma unidade. Isso transforma o problema contínuo, que pode ser extremamente complexo, em um problema discreto mais fácil de analisar. Por meio desse mapa, é possível identificar padrões e estruturas, como pontos fixos, ciclos limites e comportamento caótico [9].

1.4 Bilhares

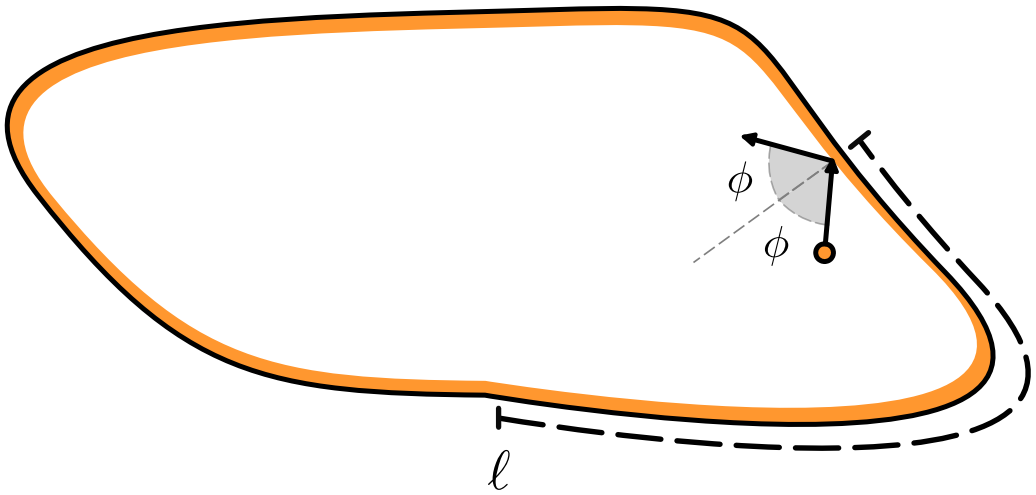

Figura 1.6: Representação esquemática de um bilhar. A partícula é indicada pelo círculo laranja com contorno preto, enquanto a fronteira do bilhar é destacada pela borda em laranja e preto. O ângulo ϕ corresponde ao ângulo de reflexão ou incidência da partícula na parede e ℓ ao comprimento do arco do bilhar.

O estudo de sistemas de bilhar é um exemplo tradicional de análise em dinâmica não linear. Nesses sistemas, uma partícula se move livremente dentro de uma fronteira fechada, interagindo com essa fronteira por meio de colisões elásticas. O sistema, que é bidimensional, é descrito por duas variáveis de posição e duas de velocidade. Esta dinâmica é principalmente analisada nos momentos em que a partícula colide com a fronteira (Fig. 1.6). A partir das posições e ângulos dessas colisões, podemos construir um mapa discreto, conhecido como *mapa de colisões*, que descreve a evolução do sistema ao longo do tempo discreto.

A geometria da fronteira desempenha um papel fundamental na dinâmica do bilhar. Dependendo da forma da fronteira, o comportamento do sistema pode variar de completamente integrável — onde o movimento é regular e previsível — até caótico, onde pequenas mudanças nas CIs podem causar grandes diferenças na evolução do sistema [14].

Coordenadas de Birkhoff

As Coordenadas de Birkhoff (CB) são um sistema de coordenadas usado para descrever o comportamento de partículas em bilhares. Em vez de usar coordenadas tradicionais, que são quadrimensionais para o EF completo, as CB são projetadas para descrever o comportamento da partícula apenas nos pontos de colisão ao longo da borda do bilhar. Isso reduz a complexidade do problema ao representar o sistema como um mapeamento dinâmico em uma **dimensão reduzida** [25]. Em sistemas dinâmicos como os bilhares, existem dois tipos principais de coordenadas:

1. **Coordenadas do EF**: Estas coordenadas são quadridimensionais e descrevem a posição e o momento da partícula em todo o EF. Para o bilhar, essas coordenadas são x, y, v_x e v_y, onde x e y representam a posição da partícula, e v_x e v_y representam as componentes da velocidade da partícula.

2. **CB**: Estas coordenadas são uma redução do sistema original e são usadas para descrever apenas o comportamento da partícula na borda do bilhar onde ocorre a colisão da partícula com as paredes. São coordenadas bidimensionais que substituem a descrição contínua do movimento da partícula por um mapeamento de colisões discretas. As CB geralmente são compostas por:

    - ℓ, que é uma parametrização do comprimento do arco na borda, ou seja, a posição da colisão ao longo da borda como demonstra a Fig. 1.6.
    - $\text{sen}(\phi)$, que representa o seno do ângulo da direção da partícula na colisão, medido em relação ao vetor normal da parede de colisão.

Uma das principais utilidades das CB é a análise do EF em sistemas com comportamentos caóticos. No caso do Bilhar Cogumelo [26] ou do Bilhar Limão [27], que possuem um EF misto, as CB podem ser usadas para visualizar e calcular a proporção do EF caótico em relação ao total. O uso das CB transforma o problema de estudar o bilhar contínuo em um problema de mapeamento discreto, o que facilita a análise, especialmente em sistemas dinâmicos que podem exibir comportamento caótico.

1.4.1 Bilhar Circular

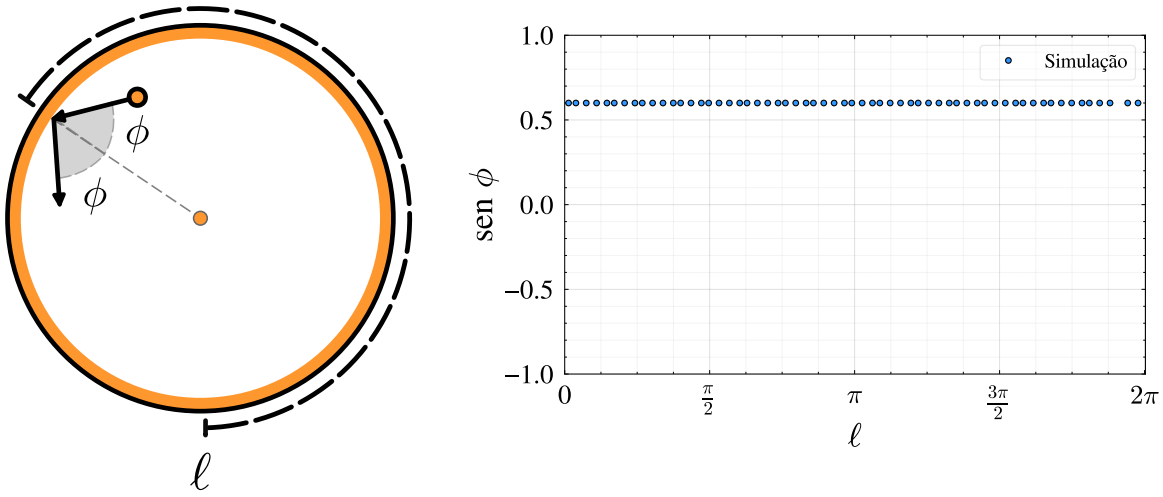

Figura 1.7: **Esquerda:** Ilustração da fronteira de um bilhar circular, com as CB **Direita:** MP nas CB do Bilhar Circular (BC), feito em simulação, mostrando uma órbita quase-periódica.

Quando a fronteira de um bilhar é circular (Fig 1.7), duas quantidades conservam-se: a energia mecânica e o momento angular. Sistemas com dois graus de liberdade com duas quantidades conservadas são ditos integráveis, onde o comportamento caótico não é observado. No EF de um Bilhar Circular (BC), observa-se apenas pontos ou linhas contínuas, refletindo órbitas periódicas ou quase-periódicas respectivamente.

De fato, o BC é caracterizado por sua insensibilidade às CI, o que significa que pequenas variações na posição ou no ângulo de lançamento da partícula não resultam em mudanças significativas no comportamento das trajetórias [28]. Essas propriedades fazem do bilhar circular um modelo fundamental para a compreensão de sistemas integráveis, servindo como contraponto aos sistemas não integráveis, onde o caos e a sensibilidade às CI são predominantes.

1.4.2 Bilhar de Sinai

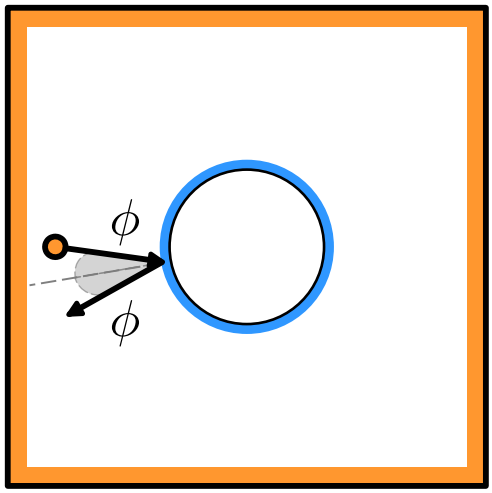

Figura 1.8: Representação esquemática do bilhar de Sinai, quadrado com um obstáculo circular central, em azul.

No bilhar proposto por Yakov G. Sinai, a fronteira é delimitada por uma borda quadrada, com um círculo central inacessível como representado na Fig. 1.8, onde todas as trajetórias possíveis apresentam ELs positivos, evidenciando o comportamento caótico [29]. O principal feito de Sinai com esse modelo foi demonstrar que as moléculas de gás seguem trajetórias descritas no sistema dinâmico de Hadamard, cuja análise do caos matemático foi pioneiramente abordada por Hadamard em 1898, onde o bilhar não apresenta o obstáculo central [30].

Em um bilhar de borda retangular e sem o obstáculo central, o comportamento caótico não é observado. Pequenas alterações nas CI podem gerar desvios ao longo do tempo, mas o desvio será uma função sub-exponencial do tempo. Durante muito tempo, acreditou-se que o desvio exponencial das trajetórias próximas era causado pela forma côncava da borda interna, sendo essa forma considerada essencial para produzir caos, tal como uma lente divergente dispersa a luz, por isto, bilhares do tipo Sinai são chamados de dispersivos. No entanto, Leonid Bunimovich mostrou que bilhar com formato de estádio, que possui lados opostos curvos, pode apresentar comportamento caótico, mesmo sendo totalmente convexa [31].

1.4.3 Bilhar Estádio de Bunimovich

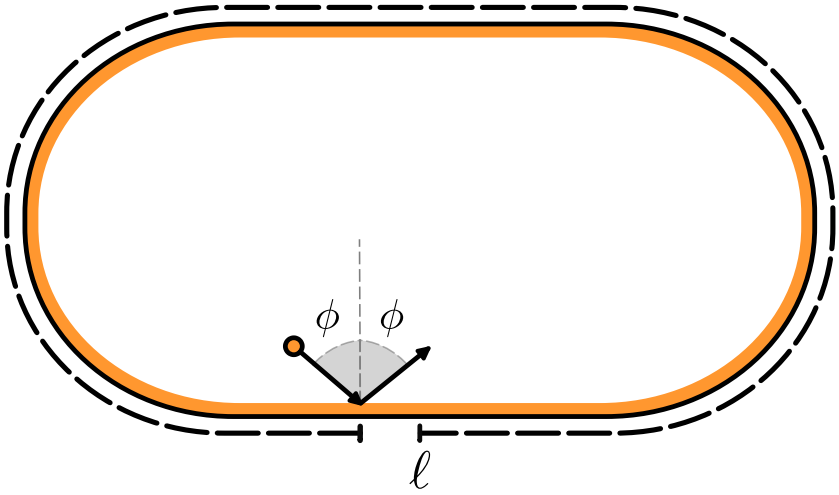

Figura 1.9: Representação gráfica do Bilhar Estádio com bordas semi circulares e segmentos retos.

A principal contribuição de Bunimovich com esse modelo foi demonstrar que o caos pode surgir mesmo em sistemas com fronteiras convexas, estendendo o resultado anterior de que formas côncavas, como no bilhar de Sinai, eram essenciais para gerar comportamento caótico. Ao contrário das bordas côncavas, que atuam dispersando as trajetórias, as fronteiras convexas no bilhar de Bunimovich ainda permitem o desvio exponencial das trajetórias próximas, resultando em ELs positivos [22, 32–34]. Este efeito é possível devida a característica desfocalizadora do bilhar. As colisões com os semicírculos conservam o momento angular. Esta conservação é quebrada nas colisões com as regiões retangulares.

Bunimovich também estendeu essa ideia para uma classe mais ampla de bilhares, permitindo a substituição dos semicírculos representada na Fig. 1.9 por curvas estritamente convexas gerais e relaxando a exigência de que os segmentos retos sejam paralelos. Esses modelos ampliados ainda preservam as propriedades caóticas originais, comprovando a robustez do caos em uma variedade de condições geométricas [23, 35].

1.4.4 Bilhar Limão

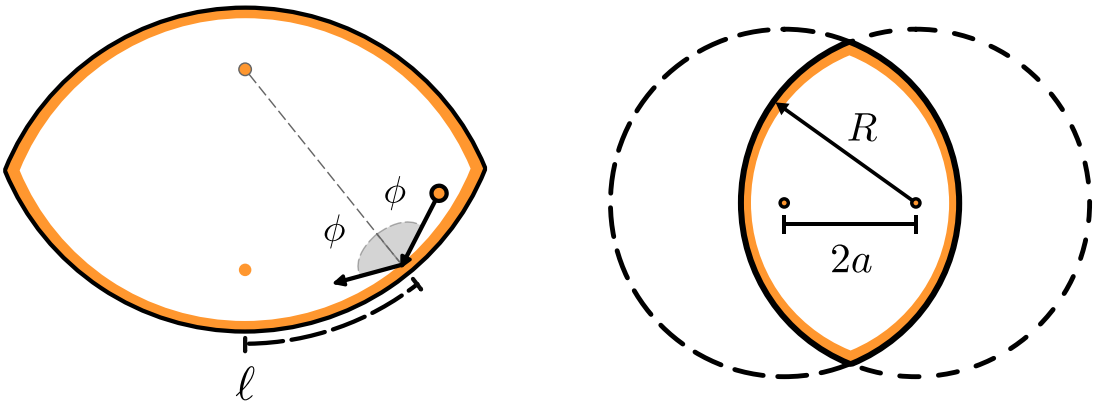

Figura 1.10: **Esquerda:** Bilhar Limão. **Direita:** Formação do Bilhar Limão a partir da interseção de dois círculos de raio R e separação nos centros de $2a$.

O Bilhar Limão (BL) representa uma generalização do estádio de Bunimovich, incorporando uma perturbação geométrica que quebra a integrabilidade inerente ao BC, ao aproximar suas metades. Especificamente, o BL é definido pela interseção de dois círculos de raio R, separados pela distância $2a$ entre seus centros, onde $a \in [0, R)$. A variação do parâmetro a introduz diferentes regimes dinâmicos, permitindo a exploração de comportamentos que variam desde a integrabilidade até o caos total.

A dinâmica do BL é caracterizada por um EF misto, contendo regiões de estabilidade regular conhecidas como *ilhas de estabilidade* e áreas caóticas, denominadas *mares de caos*. As ilhas de estabilidade correspondem a trajetórias periódicas ou quase periódicas, onde as partículas seguem órbitas previsíveis e confinadas a subespaços específicos do EF. Essas regiões são influenciadas pelas ressonâncias do sistema, que criam barreiras dinâmicas limitando o movimento das partículas. Em contraste, os mares de caos são regiões onde as trajetórias exibem sensibilidade extrema às CIs, característica fundamental do comportamento caótico. Nesses domínios, pequenas variações nas CIs resultam em divergências exponenciais nas trajetórias subsequentes, levando a uma exploração abrangente e aparentemente aleatória do EF. A coexistência de ilhas de estabilidade e mares de caos no BL apresenta uma transição contínua entre ordem e desordem [27]. Na Fig. 1.11 observam-se vários exemplos de casos mistos. As regiões brancas são as ilhas de estabilidade, enquanto as regiões escuras são os mares de caos. Tais resultados foram extraídos de [36].

O BL serve como um modelo protótipo para investigar a transição entre comportamento integrável e caótico em sistemas dinâmicos hamiltonianos. A análise das trajetórias no EF permite a aplicação de diversas ferramentas teóricas e computacionais, como o cálculo de ELs, análise de estruturas de fase e mapeamento de regiões de estabilidade e caos.

Chen *et al.* expandiram os estudos sobre o BL ao apresentarem uma generalização do modelo em [37]. Essa generalização incorpora modificações geométricas e dinâmicas que permitem explorar um conjunto ainda maior de comportamentos, incluindo novas transições entre regimes dinâmicos. Tais avanços demonstram a versatilidade do BL como modelo para compreender fenômenos não lineares em sistemas físicos.

Recentemente, o BL tem recebido crescente atenção no contexto da mecânica quântica [38–40]. Nesse domínio, o interesse pelo modelo está relacionado à sua capacidade de elucidar a correspondência entre sistemas clássicos caóticos e suas contrapartes quânticas. Em particular, aspectos como a distribuição espectral de níveis de energia e o comportamento de funções de onda em sistemas quânticos caóticos têm sido investigados no BL. Grande parte dos estudos recentes está relacionada ao efeito de *stickiness* [41, 42]. Esse fenômeno descreve o aprisionamento temporário de trajetórias caóticas nas proximidades de regiões regulares, como as ilhas de KAM, que aparecem em EFs mistos. De acordo com Zaslavsky [43], "o tempo que uma trajetória pode permanecer na camada de fronteira de uma ilha depende da profundidade de sua penetração na camada, e esse tempo pode ser extremamente longo. Esse fenômeno de aprisionamento é, às vezes, chamado de *stickiness*".

O efeito de *stickiness* possui implicações significativas para a dinâmica global do sistema, pois ele pode alterar as estatísticas de transporte caótico, introduzindo uma escala de tempo característica muito longa. Por exemplo, trajetórias que exibem *stickiness* podem contribuir para a formação de padrões temporários de ordem dentro de um regime majoritariamente caótico. Esse comportamento tem relevância tanto para sistemas clássicos quanto quânticos, com aplicações que vão desde o estudo de plasmas até a dinâmica de partículas em campos eletromagnéticos. Assim, o BL, com sua rica variedade de comportamentos dinâmicos e forte relevância teórica, continua sendo um modelo essencial para explorar fenômenos não lineares, transições de regime dinâmico e a interface entre a dinâmica clássica e quântica.

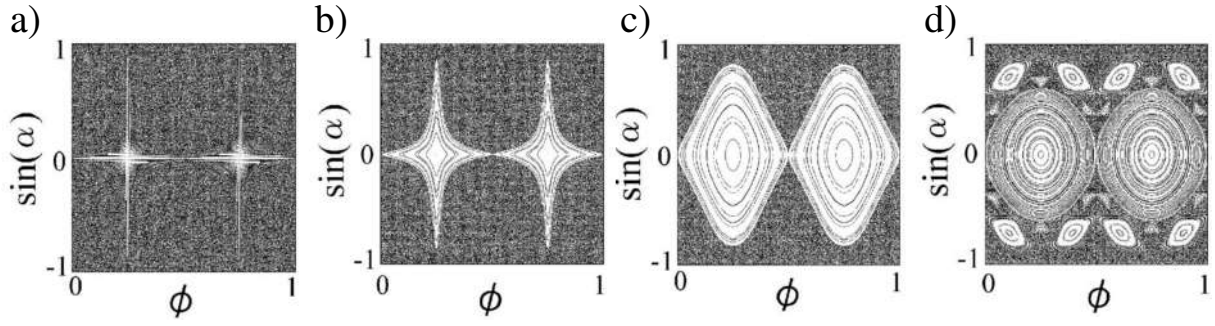

Figura 1.11: Espaços de Fase, para o bilhar Limão para diferentes valores de a. Figuras extraídas de [36].

Capítulo 2

Montagem Experimental

Para investigar a sensibilidade às CIs no BL e compreender como pequenas variações nessas condições podem desencadear o seu comportamento, desenvolvemos um experimento de baixo custo e facilmente reproduzível. Tradicionalmente, sistemas ópticos são amplamente utilizados para o estudo do caos, dada a precisão com que replicam dinâmicas complexas [44–47]. No entanto, experimentos ópticos exigem equipamentos de alta tecnologia, como lasers, câmeras de alta velocidade e cavidades específicas, o que os torna dispendiosos.

Como alternativa, propomos a utilização de um robô programável, controlado por Arduino, para simular o comportamento de uma partícula em um sistema de bilhar. Este robô pode ser equipado com sensores e algoritmos de controle que permitem simular com precisão as leis de colisão elástica. Partindo de uma CI, ele pode seguir trajetórias bem definidas e, ao colidir com as bordas do bilhar, reagir conforme as leis físicas que regem essas interações [1]. O uso do robô permite a introdução de variações controladas nas CIs, como posição e velocidade, facilitando o estudo experimental da sensibilidade a essas CIs. O parâmetro do bilhar também pode ser configurado para ajustar parâmetros específicos, permitindo que os experimentadores observem o impacto das perturbações.

2.1 Materiais e Componentes Utilizados

Nesta seção serão apresentados os principais elementos empregados na construção do robô, detalhando as características de cada componente e suas contribuições para o funcionamento do sistema. Ao abordar aspectos técnicos de cada material, essa seção visa esclarecer a funcionalidade específica de cada elemento, destacando o papel dos microcontroladores, motores, sensores, reguladores de tensão e baterias. Esses componentes foram escolhidos para garantir precisão, eficiência e confiabilidade nas operações do robô, considerando suas especificações técnicas e adequação às necessidades do projeto.

2.1.1 Arduino

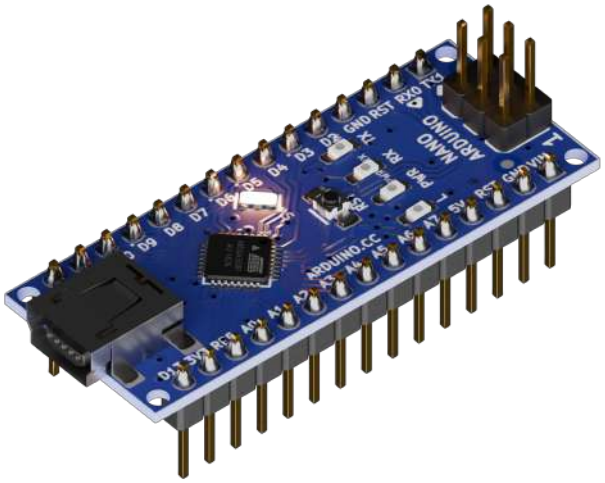

Figura 2.1: Arduino Nano (ARN), microcontrolador renderizado em 3D.

O Arduino é uma plataforma de prototipagem eletrônica aberta que combina hardware e software de fácil utilização, projetada para tornar a programação e a criação de circuitos acessíveis a iniciantes e avançados. Ele é baseado em uma placa de microcontrolador, em um ambiente de desenvolvimento integrado (IDE), os usuários podem escrever e carregar códigos na placa [48]. Existem várias versões do Arduino, como o Arduino Uno, Nano, Mega, entre outros, cada uma com características específicas para diferentes tipos de projetos. Essas placas permitem conectar componentes variados, como sensores, LEDs, motores e displays, que podem ser controlados por meio de um código escrito na IDE Arduino, que utiliza uma linguagem de programação baseada em C/C++ [49].

No nosso robô foi utilizado o Arduino Nano (ARN), uma placa de desenvolvimento compacta, equipada com um microcontrolador ATmega328 de 8 bits e projetada para facilitar a criação de protótipos rápidos e de baixo consumo energético. Com um tamanho reduzido, é especialmente útil em projetos que exigem uma interface amigável com a breadboard e suportam uma série de comunicações seriais, incluindo UART (*Universal Asynchronous Receiver-Transmitter*), SPI (*Serial Peripheral Interface*) e I^2C (*Inter-Integrated Circuit*). Essa placa conta com 20 pinos de entrada/saída digitais, 8 pinos analógicos e oferece suporte a diversos modos de operação de baixa energia. Ela pode ser alimentada por meio de uma porta Mini-B USB, uma entrada de alimentação externa ($7 - 15$ V) ou diretamente por um pino de 5 V. Os pinos digitais e analógicos permitem a conexão de sensores, LEDs, motores e outros dispositivos, tornando-a ideal para aplicações em áreas como robótica, controle ambiental e segurança [50].

2.1.2 Motores

Servomotores (SVMs) são componentes eletrônicos amplamente utilizados em diversas aplicações devido à sua capacidade de controlar com precisão posição angular ou linear, velocidade e aceleração. Embora sejam de pequeno porte, eles fornecem alto torque e são energeticamente eficientes. Esses motores são empregados em brinquedos controlados remotamente, robôs, aeronaves, e possuem também vasta aplicação industrial, incluindo áreas como robótica, manufatura, farmacêutica e serviços alimentícios [51].

No funcionamento interno, um SVM conta com um motor de corrente contínua (DC) ou alternada (AC), uma roda de controle, engrenagens e um circuito de controle, além de um sensor de feedback, geralmente um potenciômetro. O eixo do motor é acoplado a essa roda de controle, permitindo que o circuito de controle monitore a posição do eixo continuamente. Conforme o motor gira, a resistência do potenciômetro varia, o que permite ao circuito de controle ajustar o movimento do eixo com precisão e determinar a direção e o ângulo exatamente. Quando o eixo atinge a posição desejada, a alimentação do motor é interrompida, economizando energia e estabilizando o eixo na posição. Caso o eixo não esteja na posição desejada, o motor ajusta sua rotação de acordo com a diferença entre a posição atual e a desejada. Este tipo de controle, chamado controle proporcional, faz com que o motor ajuste sua velocidade proporcionalmente ao erro de posição, de forma que quanto mais próximo o eixo estiver da posição final, mais lento ele se moverá. O controle de SVMs é geralmente realizado por meio de um sinal de largura de pulso variável (*PWM - Pulse Width Modulation*), onde o comprimento do pulso determina a posição do eixo. Em SVMs típicos, um pulso de 1,5 ms posiciona o eixo a 90°, enquanto pulsos mais curtos ou mais longos alteram o ângulo para direções específicas, de 0° a 180°. Para manter o motor na posição desejada, é necessário que o pulso PWM seja repetido a cada 20 ms. Caso uma força externa atue contra o servomotor, ele resistirá ao deslocamento até o limite de torque, que é a medida de força máxima que o servomotor consegue exercer.

Existem diferentes tipos de SVMs, sendo que os modelos AC suportam maiores picos de corrente e são mais comuns em aplicações industriais pesadas. Já os modelos DC são mais utilizados em pequenas aplicações e são mais acessíveis. Alguns SVMs DC foram projetados para rotação contínua, possuindo rolamentos de esferas no eixo de saída para reduzir o atrito e permitir acesso fácil ao potenciômetro. No presente trabalho, utilizamos o FS5103R, um servomotor DC analógico contínuo de 6 V [52].

2.1.3 Sensores de Distância

O VL53L0X é um sensor de medição de distância baseado na tecnologia *Time-of-Flight* (ToF), desenvolvido pela STMicroelectronics, que utiliza um laser infravermelho de 940 nm, invisível ao olho humano, para realizar medições precisas de distância de até

2 metros. Equipado com um laser VCSEL (*Vertical Cavity Surface-Emitting Laser*) e um array de diodos de avalanche de fóton único (SPAD), o VL53L0X opera com alta precisão, mas o alcance e a exatidão podem variar dependendo da refletividade do alvo e das condições ambientais. Este sensor é projetado para aplicações onde o cálculo preciso de distâncias curtas é essencial, como em sistemas de controle de acesso, robótica para detecção de obstáculos e automação residencial, incluindo funções de monitoramento de inventário e controle de nível de líquidos. O sensor possui recursos avançados de compensação de *crosstalk* óptico, que facilitam seu uso em ambientes com vidro ou superfícies translúcidas, minimizando interferências e garantindo maior precisão. Além disso, ele oferece fácil integração via interface I^2C, que permite a comunicação direta com microcontroladores, como Arduino, e outros dispositivos. O controle do sensor é feito por meio de uma *Application Programming Interface* (API), que disponibiliza funções de calibração, inicialização, escolha de modos de operação e perfis de medição específicos para diferentes exigências de precisão e velocidade. O VL53L0X conta com três modos principais de medição:

1. **Medição Única (Single Ranging)**: realiza apenas uma medição e retorna ao modo standby.
2. **Modo Contínuo (Continuous Ranging)**: realiza medições de forma ininterrupta até ser interrompido pelo usuário.
3. **Modo Cronometrado (Timed Ranging)**: permite medições contínuas com intervalos definidos entre cada leitura.

Essa versatilidade e eficiência tornam o VL53L0X uma escolha robusta e confiável para aplicações de medição de distância, especialmente em sistemas compactos onde a precisão e a rapidez de resposta são essenciais. De acordo com a fabricante, o erro na medição do VL53L0X é caracterizado pela variação padrão (desvio padrão) e depende de fatores como a refletividade do alvo, a distância e as condições ambientais. Em ambientes internos sem luz infravermelha, a precisão para um alvo branco (88% de refletância) a 120 cm de distância tem um desvio padrão de 4% com um tempo de medição de 33 ms, e 3% com 66 ms. Para um alvo cinza (17% de refletância) a 70 cm, o desvio padrão é de 7% a 33 ms e 6% a 66 ms [53].

2.1.4 Controlador de Voltagem

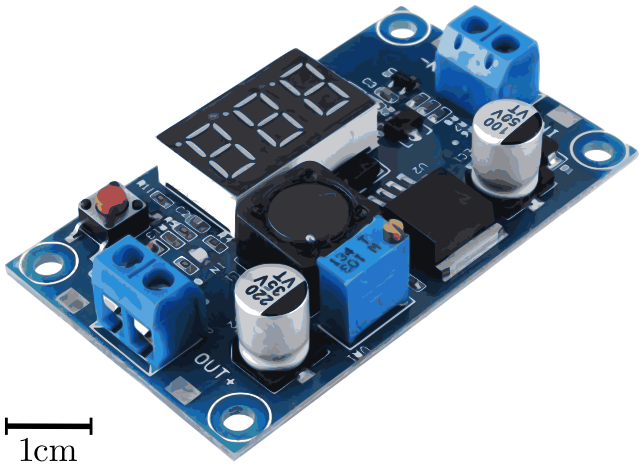

Figura 2.2: Controlador de Voltagem, LM2596, imagem modificada de[54].

O LM2596 é um regulador integrado monolítico que desempenha a função de conversor DC-DC do tipo step-down, ou seja, reduz a tensão de entrada para níveis inferiores na saída. Este regulador opera em uma frequência de comutação fixa de 150 kHz, permitindo um design simplificado e exigindo poucos componentes externos, incluindo um capacitor de entrada e saída, um diodo de recuperação rápida e um indutor, que é selecionado de acordo com a corrente e a tensão de saída desejadas. Ele é capaz de fornecer uma corrente de saída de até 3 A, sendo ideal para aplicações que requerem regulação de linha e carga. Com uma faixa de tensão de entrada de até 40 V e opções de saída fixa em 3,3 V, 5 V e 12 V, além de uma versão ajustável que permite uma saída entre 1,2 V e 37 V, o LM2596 é adequado para uma ampla gama de aplicações. O dispositivo utiliza um sistema de feedback que ajusta o ciclo de trabalho da comutação interna para manter a saída estável, compensando variações na carga e na entrada. Adicionalmente, o LM2596 incorpora funções de proteção contra sobretemperatura e limitações de corrente, o que aumenta sua robustez e confiabilidade. O dispositivo tem a capacidade de operar em modo descontínuo, adequada para cargas leves ou quando uma redução no consumo energético é desejada. A eficiência energética do LM2596 é alta, sendo que em condições ideais, ele atinge até 90%, dependendo da tensão de entrada e saída, o que reduz o aquecimento e melhora a durabilidade do circuito [55].

2.1.5 Controlador de Velocidade

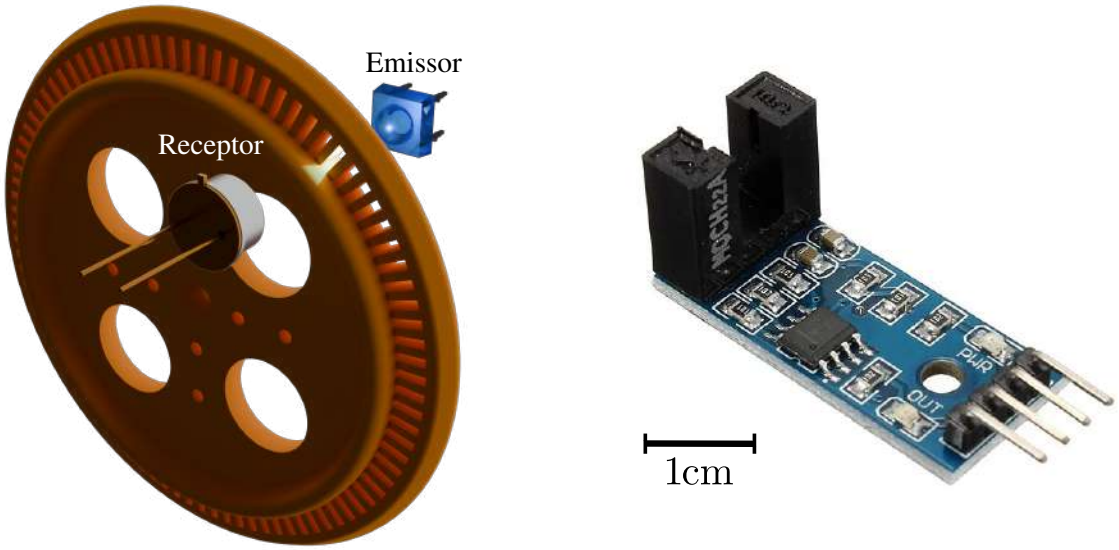

Figura 2.3: **Esquerda:** Esquema de funcionamento de um encoder, mostrando o disco com marcações, emissor e receptor [56]. **Direita:** Módulo LM393 utilizado para controle de velocidade, incluindo suas conexões e indicadores LED, Imagem retirada de [57].

Para controlar a velocidade do dispositivo, utilizamos um encoder, que é um sensor eletromecânico responsável por converter a posição em sinais elétricos digitais. Os encoders permitem quantificar distâncias, controlar velocidades, medir ângulos e contar rotações, entre outras aplicações [56, 58]. Basicamente, um encoder é composto por um disco com marcações, um emissor (LED) e um receptor (*Fotodiodo*), conforme ilustrado na Figura 2.3. O fotodiodo é um componente semicondutor essencial no funcionamento do encoder, atuando como receptor de luz emitida pelo LED. Ele opera com base no efeito fotoelétrico, onde a luz incidente transfere energia suficiente para excitar elétrons na junção semicondutora do dispositivo. Esse processo gera uma corrente elétrica proporcional à intensidade luminosa [58, 59]. No encoder, o fotodiodo detecta as interrupções da luz causadas pelas marcações do disco, convertendo-as em sinais elétricos que correspondem à posição ou velocidade do movimento. O dispositivo utilizado para o controle de velocidade é o LM393, um comparador de tensão de uso geral amplamente empregado em módulos de sensores de velocidade para microcontroladores, como o Arduino. Este módulo é ideal para medições de velocidade de motores, contagem de pulsos e detecção de posição, funcionando como um sensor óptico que utiliza um acoplamento com fenda de 5 mm para detecção precisa. O LM393 opera em uma faixa de tensão de 3 V a 5 V e possui duas

saídas: uma digital (D0) e uma analógica (A0), sendo que esta última geralmente não é utilizada neste módulo específico. Quando algo interrompe a fenda do sensor, o módulo gera um pulso digital no pino D0, alternando entre 0 V e 5 V. Essa saída digital, de nível TTL, facilita a conexão direta com o pino de entrada de um microcontrolador, permitindo que este registre a passagem e, consequentemente, calcule a velocidade do motor [60].

Na prática, o LM393 pode ser conectado a dispositivos como relés, buzzers ou interruptores de limite para diversas aplicações, incluindo alarmes e controle de posição. Além disso, o módulo possui LEDs indicadores de estado tanto para a alimentação quanto para o sinal de saída, facilitando a visualização do funcionamento durante a operação. O módulo LM393 é compacto, medindo aproximadamente 3,2 cm $\times$ 1,4 cm, e possui um orifício para parafuso, o que simplifica sua instalação em projetos.

2.1.6 Baterias

As baterias 18650 são células de íon-lítio amplamente utilizadas em dispositivos eletrônicos, veículos elétricos e sistemas de armazenamento de energia devido à sua alta densidade energética e eficiência. Com dimensões cilíndricas de aproximadamente 18,4 mm de diâmetro e 65,2 mm de altura e peso de 46,5 g, essas baterias são projetadas para fornecer um bom desempenho em um formato compacto, adequado para uma grande variedade de aplicações. As especificações técnicas dessas baterias são essenciais para garantir sua segurança e durabilidade. A capacidade nominal das baterias 18650 é de 2600 mAh, medida com uma descarga de 0,52 A até uma tensão de 2,75 V a uma temperatura de 25 °C $\pm$ 5 °C. Essa capacidade representa a quantidade de carga que a bateria pode armazenar e fornecer ao longo do tempo, atendendo à maioria das necessidades de carga em dispositivos de consumo e industriais. A tensão nominal da bateria é de 3,7 V, enquanto a tensão de carga máxima é de 4,20 V $\pm$ 0,05 V e a tensão de corte de descarga é de 3,0 V. Operar a bateria dentro desses limites é fundamental para prolongar sua vida útil e evitar danos. A corrente de carga padrão é de 0,52 A, enquanto a corrente de carga rápida é de 1,3 A. Em relação à temperatura, as baterias 18650 são projetadas para operar de 0 °C a 45 °C durante a carga e de -20 °C a 60 °C durante a descarga, o que as torna adaptáveis a diferentes ambientes. Para armazenamento prolongado, recomenda-se uma temperatura entre -5 °C e 35 °C por até um mês ou entre -20 °C e 25 °C por até seis meses, mantendo a célula com 50% de carga e uma tensão de 3,75 a 3,80 V para preservar a saúde da célula [61].

2.2 Circuito Eletrônico e Conexões

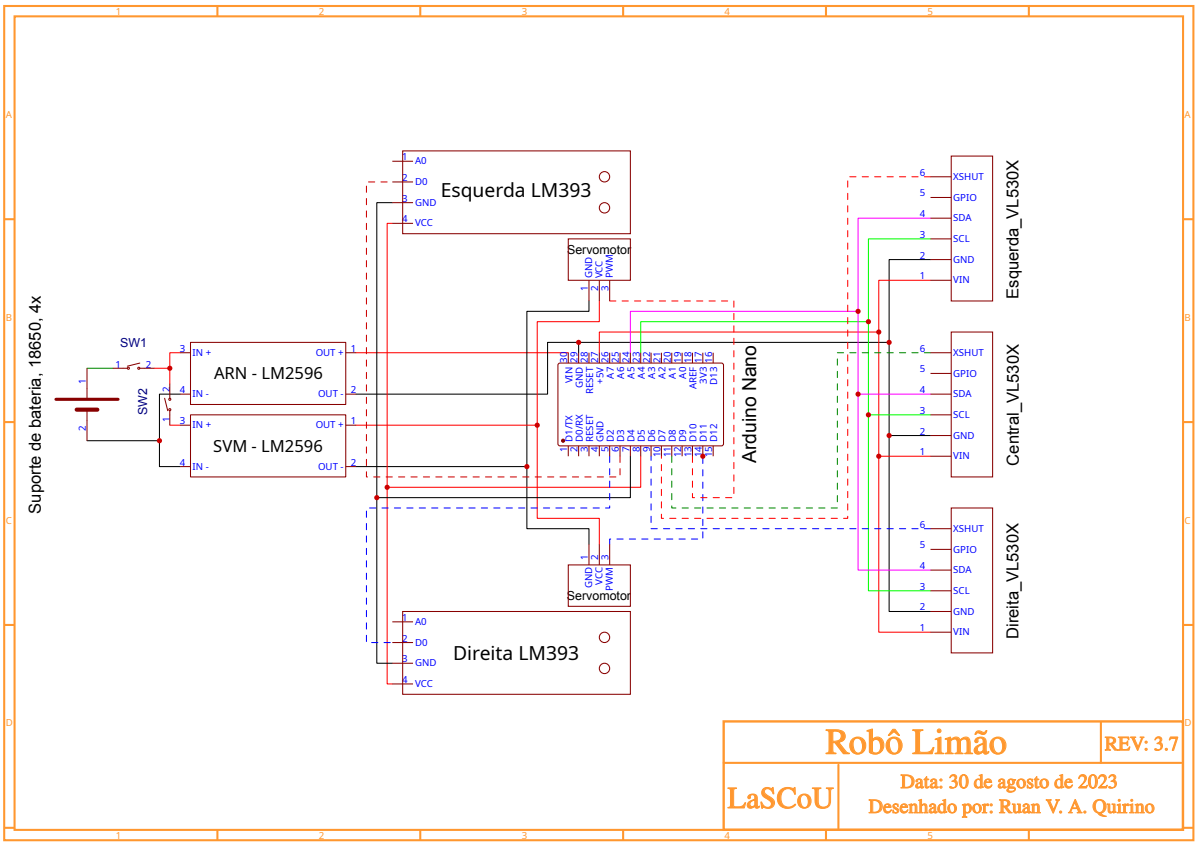

Figura 2.4: Esquema elétrico do robô, com todos os componentes.

Identificação	Descrição
———	**VCC**: Pino de entrada de tensão positiva que alimenta os componentes.
———	**GND**: Pino de referência de tensão (terra), permitindo o retorno da corrente no circuito.
———	**SCL**: Linha de clock do protocolo I^2C, que sincroniza a comunicação entre dispositivos.
———	**SDA**: Linha de dados do protocolo I^2C, que transporta as informações entre dispositivos conectados.
- - - - - - - - -	Conexão dos componentes no lado direito do robô, como D0 do LM393 e XSHUT dos VL530X.
- - - - - - - - -	Conexão de comunicação com o sensor de distância no centro do robô.
- - - - - - - - -	Conexão dos componentes no lado esquerdo do robô, como D0 do LM393 e XSHUT dos VL530X.

Tabela 2.1: Tabela de identificações e descrições dos pinos e conexões

O esquema elétrico na Fig. 2.4 descreve as conexões de um circuito que integra os componentes. O ARN é o microcontrolador central do circuito, responsável pelo controle e leitura dos sensores e atuadores utilizando os pinos digitais e analógicos para a comunicação com os demais componentes. No diagrama, estão identificados os pinos de alimentação (+5 V, GND) e os pinos digitais e analógicos conectados aos sensores e atuadores.

Para que o robô se comporte como uma partícula, é necessário identificar os obstáculos, que, no nosso caso, correspondem à fronteira do bilhar. Para isso, utilizam-se sensores de distância VL530X: três módulos VL530X estão conectados ao ARN. Esses sensores de distância possuem pinos de alimentação (VIN, GND), onde o pino VIN é a entrada de alimentação dos sensores, conectado a uma fonte de tensão de 5 V do ARN, fornecendo energia ao módulo VL530X. O pino GND representa o "terra" ou referência de tensão do circuito e deve ser conectado ao GND do ARN, completando o caminho da corrente e garantindo que todos os componentes compartilhem a mesma referência de tensão. Os pinos de comunicação (SCL, SDA) são responsáveis pela sincronização e transmissão de dados entre o sensor e o microcontrolador. O pino SCL atua como a linha de clock do protocolo I^2C, emitindo pulsos para sincronizar a comunicação entre os dispositivos e garantir que os dados sejam transmitidos com precisão temporal. Já o pino SDA é a linha de dados do protocolo I^2C, que carrega a informação transmitida entre o sensor e o microcontrolador. A combinação dos pinos SCL e SDA permite ao ARN realizar leituras precisas das medições de distância. O pino GPIO, que é uma entrada/saída digital que pode ser programada para diferentes funções, dependendo das configurações, e o pino XSHUT, usado para controlar o estado de alimentação do sensor. Ao manter o XSHUT em nível baixo (conectado ao GND), o sensor é desligado. Quando o pino está em nível alto, o sensor é ativado e funciona normalmente. Esse recurso é útil para gerenciar o consumo de energia ou para desativar temporariamente o sensor quando ele não está em uso, para diminuir o uso dos pinos no ARN. Pela falta de necessidade dessa configuração, esse pino não foi conectado ao ARN.

Os dois módulos reguladores de tensão LM2596 são usados para estabilizar a tensão fornecida pela bateria. O primeiro módulo mantém a tensão em 9 V para o ARN, garantindo que os componentes recebam a tensão correta. O regulador possui entradas e saídas de tensão (IN+ e IN- para entrada; OUT+ e OUT- para saída). Já o segundo LM2596 mantém a tensão em 6 V para os SVMs, que têm um limite de tensão de 7 V. Os módulos reguladores de tensão asseguram que, enquanto o robô estiver no experimento, as tensões permaneçam estáveis, tanto no ARN quanto nos motores. A necessidade dos controladores de tensão ocorre porque, caso a tensão no ARN fique próxima a 7 V, é possível que o sistema fique mais lento ou até reinicie, comprometendo o experimento. Para os motores, quanto menor a tensão, mais devagar eles giram, sendo necessário ajustar a rotação pelo ARN, o que também pode prejudicar o experimento.

São utilizados dois SVMs de rotação contínua conectados ao ARN para controle de

movimento. Cada motor possui três pinos: GND, VCC e PWM (sinal de controle). O pino PWM é conectado aos pinos de saída digital do ARN, que controla a velocidade angular do SVM enviando sinais de pulso. Os sensores de velocidade LM393 têm a função de monitorar a rotação dos motores utilizando um sensor óptico com uma fenda. Quando um objeto passa por essa fenda, ele bloqueia a luz, causando uma mudança de sinal detectada pelo comparador. Essa mudança gera um pulso digital na saída do pino D0, que pode ser lido pelo ARN para contar a rotação do motor. O pino A0 no módulo LM393 está presente, mas não possui função ativa neste módulo específico. Isso ocorre porque o LM393 é utilizado como um comparador digital que fornece uma saída binária (0 ou 1) no pino D0. Em módulos de sensores baseados no LM393, o pino A0 às vezes é incluído no design do PCB (placa de circuito impresso), mas não está conectado a nenhum circuito analógico. No caso desse módulo específico, o pino A0 não está configurado para saída analógica e, portanto, não fornece informações contínuas ou variáveis de sinal, limitando-se a uma função digital de detecção (D0).

Para alimentar o robô, foram utilizadas quatro baterias 18650 conectadas em série a um suporte que fornece energia ao sistema. A conexão das baterias ao regulador de tensão LM2596 garante que a tensão seja estabilizada antes de alimentar o ARN e outros componentes, protegendo o circuito de possíveis variações de tensão. O diagrama inclui as chaves SW1 e SW2, a chave SW1 é utilizada para ligar/desligar o ARN e seus módulos, enquanto a chave SW2 controla a alimentação dos motores.

2.3 Estrutura do Robô

2.3.1 FreeCAD

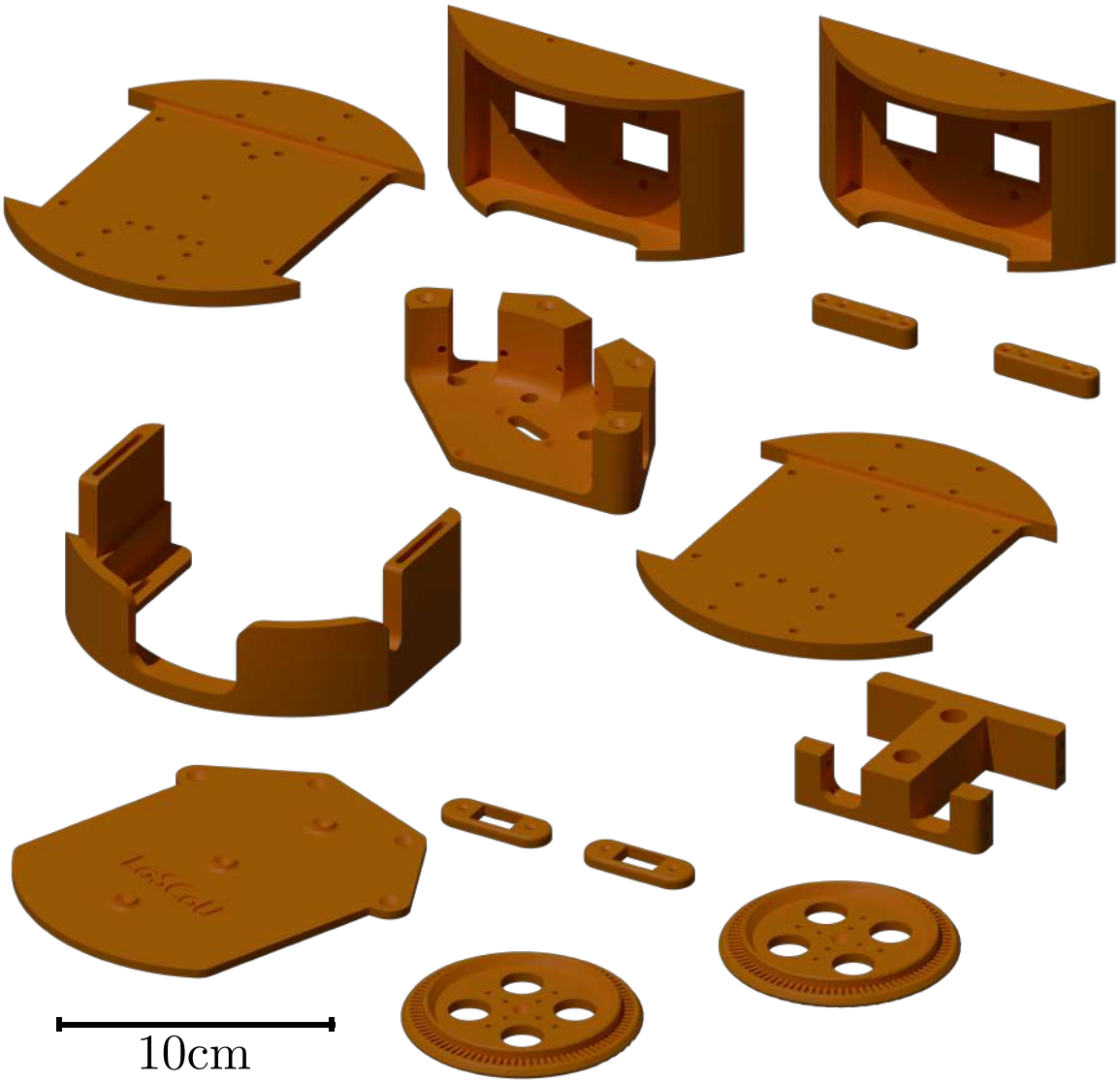

Figura 2.5: Modelos 3D criados no FreeCAD e renderizados no Blender.

O chassi e o suporte estrutural dos componentes do robô foram projetados utilizando o FreeCAD(2.5), um software de modelagem CAD 3D de código aberto amplamente utilizado em engenharia mecânica e design de produtos. Este software oferece um ambiente de modelagem paramétrica que permite ajustes rápidos e precisos nas dimensões das peças, facilitando a criação de componentes personalizados e intercambiáveis. Através de

sketches detalhados, definimos contornos e restrições geométricas para garantir o encaixe exato de motores, sensores e outros dispositivos no chassi do robô.

A arquitetura modular do FreeCAD e seu suporte a bibliotecas de código aberto, como a Open Cascade Technology (OCCT) para modelagem CAD avançada [62], permitiram configurar cada peça com precisão para a estrutura e funcionamento do robô. A funcionalidade de montagem no FreeCAD possibilitou uma pré-visualização detalhada da interação entre os componentes, verificando alinhamentos e evitando colisões antes da fabricação.

Roda

Figura 2.6: Vista frontal, de perfil e traseira da roda com disco encoder integrado.

A concepção da roda foi um dos principais motivadores para a construção do robô em 3D. O projeto exigia controle preciso dos motores, o que demandava um encoder capaz de contar o máximo possível de perfurações por rotação. Tradicionalmente, o disco do encoder e a roda são componentes separados. No entanto, optamos por integrá-los, transformando a própria roda em um encoder, como demonstrado na Fig. 2.3.

A roda possui uma largura total de 76 mm, dimensionada para garantir a estabilidade do robô e acomodar adequadamente os componentes de fixação e encoder. Esse design integrado maximiza a área útil para perfurações, permitindo um maior número de medições por rotação. O projeto final incluiu 90 perfurações uniformemente distribuídas na borda da roda. Cada perfuração possui uma largura de 1,12 mm e um comprimento de 4 mm. Esses valores foram determinados após diversas iterações no projeto, ajustando quantidade e dimensões das perfurações para otimizar a detecção pelo sensor do encoder. Testes realizados confirmaram que o encoder detectava com precisão as 90 perfurações durante uma rotação completa, sem falhas. Essa configuração trouxe as seguintes vantagens:

1. **Facilidade nos cálculos**: O encoder distingue claramente entre perfurações (luz transmitida) e espaços sólidos (luz bloqueada), o que permite contar 180 eventos

por rotação (90 perfurações e 90 espaços sólidos). Isso resulta em uma resolução angular de 2° por evento.

2. **Simplicidade na implementação**: A integração do disco do encoder à roda reduz a complexidade mecânica e os custos de fabricação, mantendo a precisão necessária para o funcionamento do robô.

Além das perfurações na borda, a roda possui nove perfurações internas utilizadas para fixação no suporte do motor. Essas perfurações são parafusadas e incluem uma perfuração central com chanfro. O chanfro, uma transição inclinada entre duas faces do objeto, foi projetado em um ângulo de 45° para melhorar a centralização do parafuso e o alinhamento com o eixo do motor. Quatro perfurações maiores foram incorporadas ao design para reduzir o consumo de filamento durante a impressão e facilitar o acesso interno ao robô. Isso permite parafusos serem ajustados sem a necessidade de desmontar a roda ou os motores, além de proporcionar visibilidade direta para monitorar as tensões exibidas nos reguladores de tensão.

Na visão de perfil da Fig. 2.6, a roda apresenta dois detalhes importantes. Primeiro, as bordas afinadas e abauladas foram projetadas para minimizar a área de contato com o solo, melhorando o desempenho do robô. Segundo, uma protuberância estrutural foi adicionada ao centro da roda para evitar deformações. Em versões anteriores, a ausência dessa estrutura resultava em rodas tortas após algum tempo de uso. A inclusão dessa protuberância garantiu maior durabilidade e alinhamento durante a operação. Na parte traseira, a roda possui uma região lisa projetada especificamente para facilitar a impressão 3D. Essa área lisa, que fica em contato direto com a base da impressora, foi incluída em todo o projeto para garantir estabilidade durante a impressão, evitando imperfeições ou desalinhamentos que poderiam comprometer a qualidade final da peça.

Base

Figura 2.7: Vista superior, perfil e perspectiva da base com perfurações para fixação e encaixe de componentes.

A base é um dos componentes mais essenciais para a estrutura do robô, servindo como suporte para todas as estruturas superiores e inferiores. Projetada para ser funcional e robusta, a base é construída em forma de um círculo com raio de 7 cm. Este formato foi escolhido para maximizar a área útil, garantir estabilidade ao robô e facilitar o encaixe de outros componentes. Ela é composta por duas partes idênticas: a base inferior e a base superior. Ambas possuem perfurações estrategicamente posicionadas para acomodar motores, sensores, suportes e as colunas. A parte inferior sustenta os motores e a estrutura principal, enquanto a parte superior serve de suporte para a tampa e outros componentes eletrônicos como o ARN, e os controladores de tensões na parte inferior. A distribuição das perfurações foi projetada para ser modular, permitindo adaptações e atualizações no design do robô. Reforços estruturais foram adicionados para evitar deformações e aumentar a durabilidade, especialmente considerando as vibrações geradas pelos motores durante o funcionamento e o peso total do robô, 780 gramas.

Suporte dos Sensores de Distância

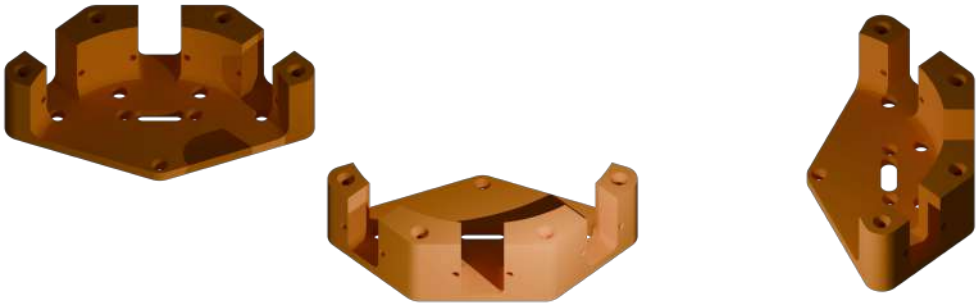

Figura 2.8: Suporte dos sensores de distância, com ângulos fixos.

O suporte dos sensores de distância foi projetado para garantir que os sensores laterais mantenham um ângulo fixo de 45° em relação ao sensor central (Fig. 2.8). Esse alinhamento otimizado permite uma cobertura ampla do ambiente ao redor do robô, aumentando a precisão na navegação e a detecção de obstáculos. Na parte inferior do suporte, existem três perfurações que permitem seu encaixe na base superior do robô. Isso assegura que o sensor central esteja perfeitamente paralelo ao movimento do robô, evitando erros de leitura. A geometria do suporte é baseada em um segmento de 3/8 de um octógono de raio externo igual ao da base, cujo centro está alinhado com o eixo do círculo da base. Essa escolha geométrica proporciona estabilidade e simplicidade na fabricação. Na parte superior, o suporte funciona também como base para a tampa do robô, conectando as partes inferior e superior de forma coesa.

Tampa e Coroa

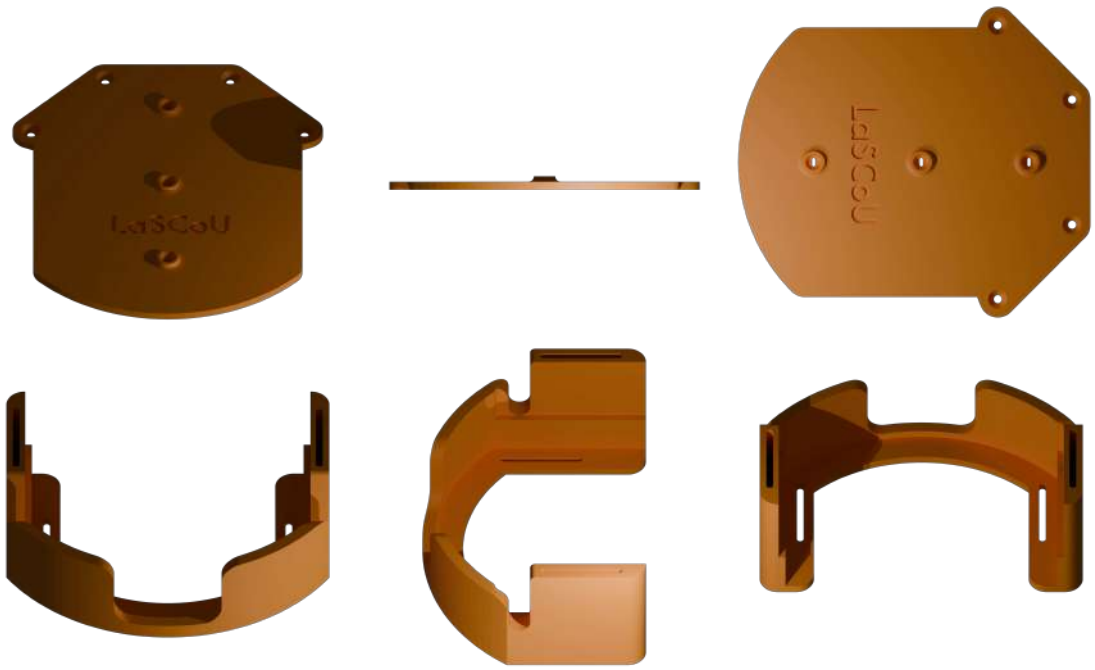

Figura 2.9: Tampa e coroa do robô, com encaixe para três LEDs e proteção contra interferência de luz.

A tampa tem como principal função proteger a parte central do robô, onde estão localizados componentes eletrônicos como o ARN. Esse design garante que reflexos ou luzes internas não interfiram no funcionamento do programa responsável por localizar o robô após a gravação em vídeo do experimento. A tampa possui três perfurações para LEDs na parte superior, como pode se observar na parte superior da Fig. 2.9, estrategicamente posicionadas para facilitar a localização visual do robô durante experimentos, o LED central está no centro geométrico da base do robô. A tampa também serve de suporte para a coroa, que envolve a lateral do robô. A coroa desempenha um papel duplo: diminui a saída de luz externa do robô e atua como suporte para o encoder integrado, localizado em sua parte inferior.

Coluna e Suporte para Switch

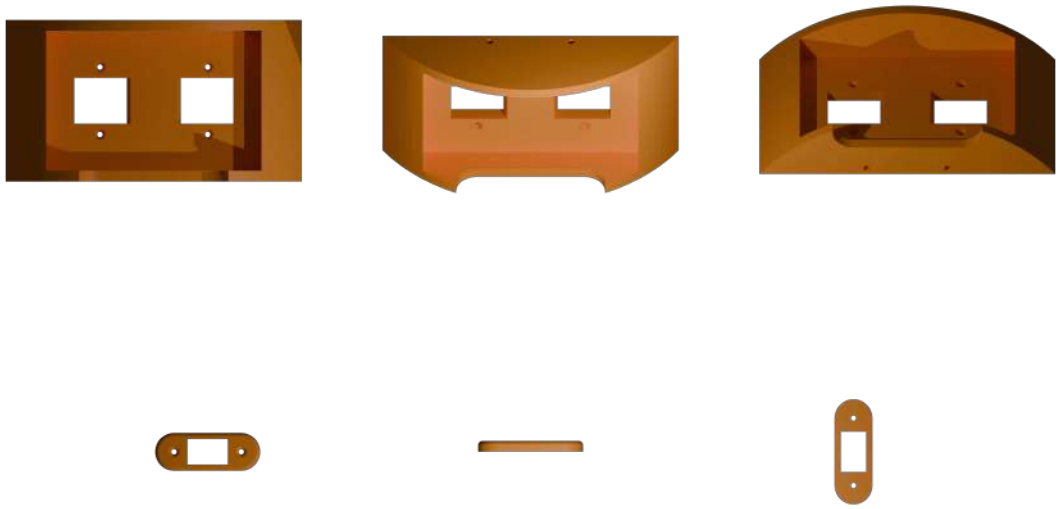

Figura 2.10: Modelo 3D da coluna e suporte para switch.

O robô utiliza duas colunas para conectar a base inferior à base superior. Cada coluna possui dois orifícios quadrados como se pode observar na Fig. 2.10. Na parte frontal, o orifício é utilizado para fixar o suporte do switch, enquanto o orifício traseiro serve como passagem para a fiação elétrica do robô, mantendo os cabos organizados e protegidos. As colunas também oferecem suporte estrutural adicional, garantindo que as bases se mantenham alinhadas mesmo sob vibrações.

Suporte para o Motor

Figura 2.11: Suporte para o motor, fixado na base inferior.

O suporte para o motor foi projetado para fixar os motores firmemente à base inferior do robô. Ele utiliza quatro parafusos para garantir estabilidade e evitar movimentos indesejados durante a operação. O suporte mantém o eixo de rotação do motor perfeitamente alinhado com o centro geométrico da base do robô, o que é essencial para a precisão na movimentação e rotação na colisão do robô. O design do suporte (Fig. 2.11) permite que diferentes tipos de motores sejam utilizados sem a necessidade de alterar o restante da estrutura do robô.

2.3.2 Conversão para G-code no PrusaSlicer

Após o desenvolvimento no FreeCAD, os arquivos foram exportados em formato STL e preparados para impressão no PrusaSlicer. O PrusaSlicer é um software de fatiamento de código aberto desenvolvido pela *Prusa Research* que converte modelos 3D em G-code, o formato lido pela impressora para executar a impressão. Esse software atua como uma ponte entre o design digital e a fabricação, transformando os arquivos STL em uma série de instruções detalhadas que orientam o movimento do extrusor, o aquecimento e a deposição de material camada por camada. O PrusaSlicer oferece um amplo conjunto de configurações para personalizar a impressão de acordo com as necessidades de cada peça. Parâmetros como:

- **Espessura de camada**: Define a resolução vertical, sendo camadas mais finas ideais para peças com detalhes complexos, embora aumentem o tempo de impressão, utilizamos a espessura de 0,2 mm.
- **Percentual de preenchimento**: Controla a densidade interna, afetando diretamente a resistência e o peso da peça; por exemplo, um preenchimento de 20% foi

utilizado para componentes leves, como a tampa, coroa, Suporte dos sensores de distância, enquanto as outras peças estruturais usaram preenchimentos de 100%.

- **Orientação de impressão**: Determina a posição da peça na mesa da impressora, o que pode influenciar a resistência e a qualidade superficial de cada face.

2.3.3 Impressão 3D e a K1 Max

Figura 2.12: **Esquerda:** Impressora 3D K1 Max utilizada no projeto. **Direita:** Modelo 3D renderizado do robô em sua configuração final.

A impressão 3D é uma tecnologia de manufatura aditiva que permite criar objetos tridimensionais ao depositar material camada por camada, conforme especificado em um design digital. O processo inicia-se com um modelo tridimensional desenvolvido em software CAD, que é posteriormente fatiado em camadas e convertido para o formato G-code, utilizado pela impressora para executar a fabricação. A *K1 Max* (Fig. 2.12), impressora 3D utilizada neste projeto, destaca-se por sua precisão e eficiência. Utilizando a tecnologia *FDM* (Fused Deposition Modeling), ela derrete filamentos plásticos — como *Polylactic Acid (PLA)*, *ABS* ou *PETG* — e os deposita camada a camada para construir o objeto final. A K1 Max é amplamente reconhecida por sua velocidade e precisão, características que a tornam ideal para a produção de peças complexas e detalhadas, atendendo com excelência aos requisitos técnicos do projeto.

Funcionamento da Impressão com a K1 Max

1. **Preparação do Material e da Impressora**: A K1 Max utiliza um rolo de filamento que é alimentado pelo extrusor. O filamento é aquecido no *hotend* (ponta quente) até o ponto de fusão. Esse material derretido é então depositado de forma precisa no leito de impressão, de acordo com o G-code gerado pelo software de fatiamento (PrusaSlicer).

2. **Movimento e Controle de Precisão**: O sistema de movimentação rápida e controlada em três eixos (X, Y e Z), com motores de passo que garantem precisão posicional. Cada camada é depositada sobre a anterior, e a precisão da K1 Max permite que as camadas sejam ajustadas com espessuras que podem variar entre 0,1 mm e 0,4 mm, conforme necessário para o detalhe da peça.

3. **Estrutura de Suporte e Adesão**: Para impressões complexas, a impressora 3D cria automaticamente suportes em áreas que não possuem material de base, garantindo que partes suspensas da peça sejam formadas corretamente. O leito de impressão aquecido ajuda na adesão da primeira camada e evita deformações, mantendo as peças firmemente fixadas durante o processo de impressão. Na criação do projeto 3D, ele foi desenhado de forma a não necessitar de suportes.

4. **Controle de Temperatura e Resfriamento**: A K1 Max permite o controle de temperatura no *hotend* e no leito de impressão, adaptando-se a diferentes tipos de materiais. O sistema de resfriamento da K1 Max garante que o material depositado solidifique rapidamente, proporcionando um acabamento de alta qualidade e evitando distorções nas camadas.

5. **Precisão e Velocidade Otimizadas**: Com sua alta taxa de deposição e movimentação rápida, a K1 Max reduz o tempo de impressão sem comprometer a precisão. Essa combinação foi essencial para este projeto, onde o tempo de impressão e a precisão eram fatores críticos para garantir que todas as peças tivessem o encaixe e a resistência necessários.

Qualidade e Montagem das Peças Impressas

A impressora 3D K1 Max foi configurada para replicar com alta fidelidade cada detalhe do design digital, resultando em peças que se encaixaram perfeitamente no chassi do robô, sem a necessidade de retrabalhos manuais significativos. O processo de impressão proporcionou componentes caracterizados por uma combinação de robustez, leveza e precisão, elementos cruciais para garantir uma montagem final eficiente e durável, contribuindo para a estabilidade e o desempenho geral do robô. Os ajustes realizados durante

o processo de impressão foram fundamentais para assegurar a resistência mecânica adequada das peças, ao mesmo tempo que reduziram o peso total do sistema, otimizando sua funcionalidade.

2.3.4 Fixação dos Componentes

Figura 2.13: Imagem real do robô impresso em 3D, com os componentes eletrônicos fixados.

Todos os sensores foram montados em posições previamente definidas no chassi, estrategicamente alinhados com o projeto desenvolvido no FreeCAD. A precisão do desenho permitiu a criação de encaixes específicos, otimizando a robustez estrutural e a funcionalidade do robô. Para a fixação, foram utilizados parafusos de 3 mm de diâmetro e 16 mm de comprimento. O processo de fixação consistiu em aquecer os parafusos com um ferro de solda, em temperaturas que variam entre 70 °C e 100 °C. Essa faixa térmica é suficiente para que o plástico utilizado na impressão 3D (PLA) passe do estado sólido para uma fase viscosa ou semilíquida, permitindo que os parafusos penetrem no material e formem roscas diretamente nos encaixes. Após o resfriamento, o PLA retorna ao estado

sólido, fixando os parafusos com segurança. Caso seja necessário remover ou reinstalar os parafusos, não é necessário reaquecê-los, pois os sulcos criados no PLA funcionam como uma rosca permanente. Ao todo, foram utilizados 36 parafusos na montagem do robô, cuja configuração final pode ser observada na Fig. 2.13.

O controlador central, um ARN, foi fixado em uma placa de cobre perfurada, conhecida como *placa universal*, que oferece uma solução modular devido às diversas perfurações. Os componentes eletrônicos foram fixados na placa por meio de solda direta nos pontos de contato, garantindo uma conexão elétrica robusta e confiável. A placa universal foi projetada para encaixar perfeitamente na base superior do robô, utilizando um sistema de fixação por pressão que elimina a necessidade de parafusos ou adesivos. Essa abordagem facilita o acesso ao ARN e aos demais componentes eletrônicos, permitindo ajustes e manutenções com rapidez e praticidade. Além disso, o ARN foi projetado para ser removível. Caso ocorra algum problema, é possível substituir o módulo simplesmente retirando o controlador antigo e encaixando um novo ARN no mesmo local, já com o programa do robô carregado.

2.4 Programação do Arduino

A programação do Arduino foi desenvolvida para garantir um comportamento simples e eficiente do robô. O robô segue em linha reta até detectar um obstáculo a 15 cm de distância, utilizando sensores de proximidade. Ao identificar um obstáculo, o robô para e utiliza um sensor adicional para calcular o ângulo necessário para desviar. Em seguida, ele realiza uma rotação em torno de seu próprio eixo até alcançar o ângulo de reflexão calculado e, então, retoma o movimento em linha reta. Para garantir que o robô se desloque de forma estável e com velocidade constante, utilizamos encoders para monitorar a quantidade de pulsos por segundo em cada roda. Isso permite que o sistema ajuste a velocidade automaticamente, corrigindo eventuais desvios e mantendo a trajetória reta. Todo o código do programa está disponível no Apêndice A.

2.5 Contorno do Bilhar

A montagem do Bilhar consiste na deformação de madeiras flexíveis em formato de limão, com o objetivo de estabelecer um ambiente controlado para experimentos com um robô. Esta seção descreve os procedimentos de preparação do contorno do bilhar, incluindo o processo de desenho no chão e a fixação de tábuas flexíveis de madeira para delimitar o perímetro. Além disso, são apresentadas as principais equações utilizadas para calcular os parâmetros geométricos do contorno, que garantem a consistência dimensional do experimento.

2.5.1 Configuração Geométrica e Fixação do Contorno

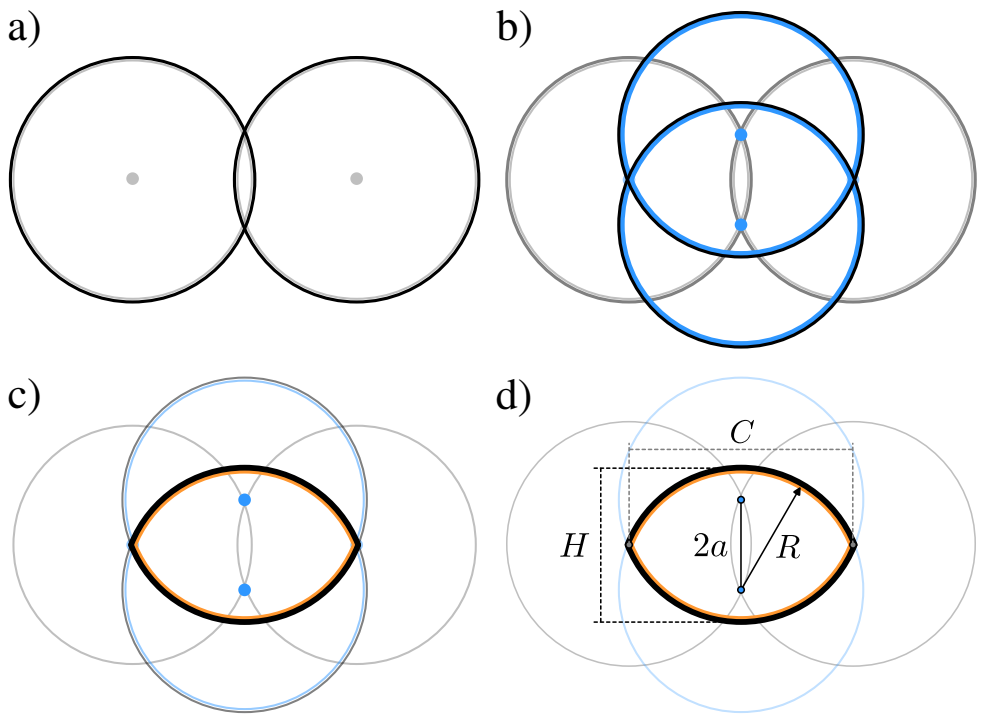

Figura 2.14: Esquema de procedimento utilizado para obter o contorno do limão.

Para a montagem do contorno do bilhar, inicialmente foram marcadas as quinas do bilhar, calculadas utilizando a eq. (2.3) e indicadas como C na Fig. 2.14d). Essa etapa é essencial, pois os dois pontos, conectados por uma linha reta, devem estar paralelos à parede do laboratório. Essa configuração serve como base para que a câmera também seja posicionada paralelamente à parede, minimizando a diferença angular entre as quinas do bilhar e o alinhamento da câmera. Por conveniência, esses pontos foram marcados sobre as juntas das lajotas, como mostrado na imagem superior da Fig. 2.15. Nela, observa-se o bilhar montado com as quinas sobrepondo as linhas das lajotas, enquanto a câmera está alinhada paralelamente a essas linhas, sendo essa imagem registrada sem correções angulares, ou seja, sem ajuste para deixar as quinas perfeitamente horizontais. Posteriormente, o raio foi calculado utilizando a eq. (2.1). Com o auxílio de um compasso improvisado — composto por um marcador em uma extremidade e um prego na outra —, os círculos foram desenhados, conforme ilustrado na Fig. 2.14a). A interseção entre dois desses círculos define o centro do círculo que compõe a borda do limão, como representado na Fig. 2.14b). Para garantir a preservação da geometria do bilhar, novos círculos foram traçados utilizando sempre o mesmo raio, evitando alterações no formato e, consequentemente, no parâmetros geométricos do limão. Caso contrário, o perímetro poderia ser comprometido, inviabilizando a funcionalidade do contorno. A delimitação das bordas foi realizada com duas tábuas flexíveis de madeira, cada uma medindo 250,5 cm de

comprimento, 20 cm de altura e 6 mm de espessura, resultando em um perímetro total aproximado de 501 cm. Essas tábuas foram fixadas ao chão por meio de encaixes e travas impressas em 3D (Fig. 2.15), posicionadas estrategicamente nas quinas para garantir a estabilidade e a precisão do contorno.

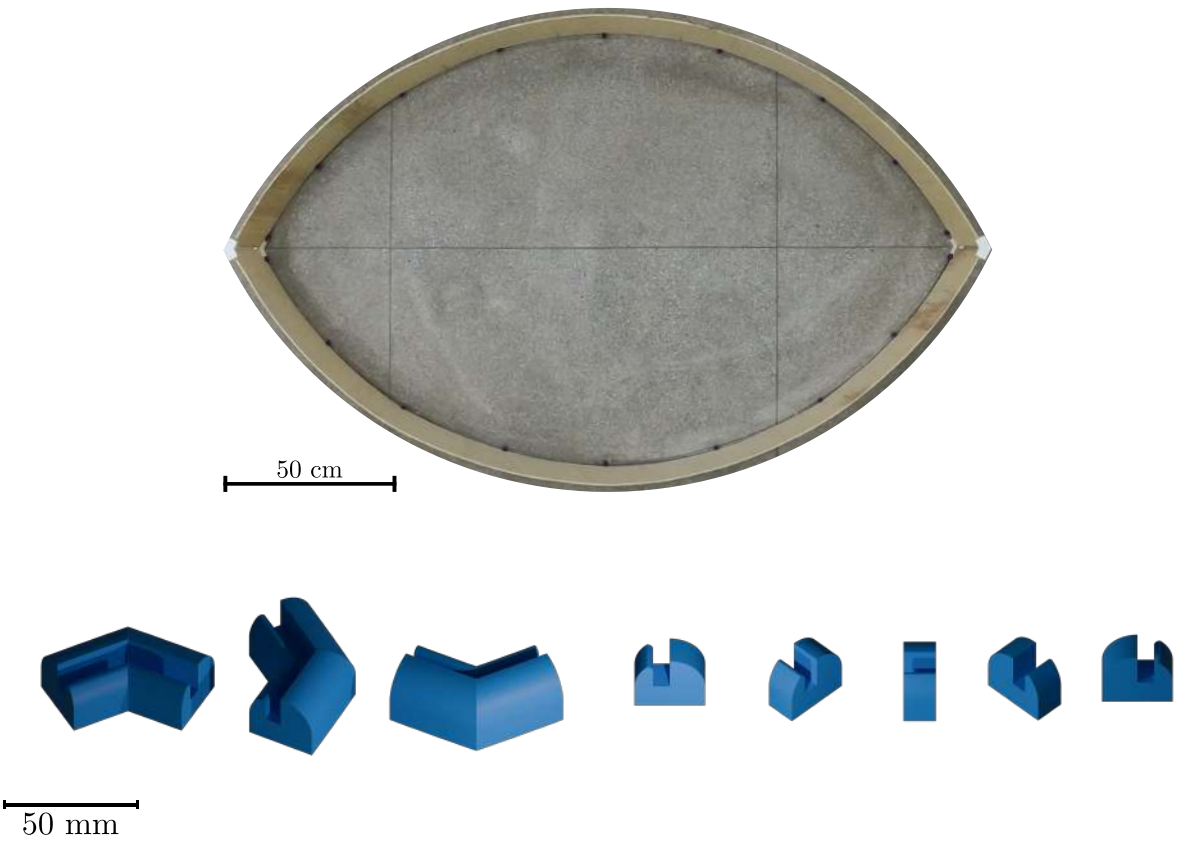

Figura 2.15: **Painel Superior:** Fotografia do contorno experimental em sua configuração final. **Painel Inferior:** Detalhes dos encaixes e travas impressas em 3D: à esquerda, os elementos utilizados nas quinas; à direita, as travas de fixação das tábuas. Na fotografia superior, as travas são visíveis em branco (quinas) e em pequenos pontos pretos (contorno).

2.5.2 Cálculo dos Parâmetros Geométricos

O bilhar limão é definido a partir de $(y \pm a)^2 + x^2 = R^2$, na qual há duas equações, uma para cada metade do contorno do bilhar. Para caracterizar o formato da borda, independentemente da escala, basta o conhecimento do parâmetro $\gamma \equiv a/R$. Esse parâmetro é o principal controlador do experimento. A partir dele e da equação acima, podemos obter todos os parâmetros geométricos relevantes do limão. São eles: o comprimento (C), altura (H), raio (R) e a para um perímetro fixado (L_0):

$$R(\gamma) = L_0 \frac{1}{4} \arcsin\left(\sqrt{1-\gamma^2}\right) \tag{2.1}$$

$$a(\gamma) = L_0 \frac{\gamma}{4} \arcsin\left(\sqrt{1-\gamma^2}\right) \tag{2.2}$$

$$C(\gamma) = L_0 \frac{\sqrt{1-\gamma^2}}{2} \arcsin\left(\sqrt{1-\gamma^2}\right) \tag{2.3}$$

$$H(\gamma) = L_0 \frac{(1-\gamma)}{2} \arcsin\left(\sqrt{1-\gamma^2}\right) \tag{2.4}$$

As distâncias entre as quinas, mostradas na Fig. 2.14 a), foram calculadas utilizando a eq. (2.3), que define a distância necessária para assegurar que o experimento permaneça paralelo à câmera GoPro. Os raios dos círculos foram calculados pela eq. (2.1), garantindo que todas as circunferências fossem congruentes. As interseções entre os círculos, destacadas por pontos laranja na Fig. 2.14, representam os centros das circunferências que definem o contorno do bilhar. Para concluir o formato do bilhar limão, o compasso foi posicionado nas interseções, repetindo os círculos conforme o desenho original. A região de interseção final entre os círculos resultou no formato de limão, com perímetro fixo de $L_0 = 501$ cm, como mostrado na Fig. 2.14 c).

Essas medidas e configurações foram fundamentais para garantir a exatidão do contorno experimental, proporcionando um ambiente controlado e replicável para a realização dos testes com o robô. Esse nível de precisão é crucial para minimizar variações nos experimentos e assegurar a consistência dos dados coletados.

Capítulo 3

Obtenção e Tratamento de Dados

A obtenção e o tratamento de dados são etapas centrais em experimentos científicos. Este capítulo apresenta os procedimentos adotados, bem como os desafios enfrentados durante a coleta e análise das informações.

A metodologia empregada foi desenvolvida com o objetivo de capturar, com precisão, a trajetória do robô em diferentes cenários experimentais. Para isso, foram utilizados dispositivos de captura de imagem de alta resolução e ferramentas de software, possibilitando um registro detalhado dos eventos e a análise minuciosa dos movimentos. A escolha dessas ferramentas foi guiada por critérios de fidelidade, eficiência e compatibilidade com o processamento subsequente, assegurando a qualidade dos dados mesmo em situações de baixa luminosidade. Um dos desafios encontrados foi a necessidade de minimizar distorções geométricas e interferências externas, como reflexos ou ruídos visuais, que poderiam comprometer a análise das trajetórias. Esses problemas foram mitigados por meio de configurações específicas, como o uso de modos de captura otimizados e o tratamento prévio das imagens em escala de cinza. Além disso, foram implementadas rotinas automatizadas para a extração e o processamento dos dados, otimizando o tempo de análise e reduzindo o impacto de erros humanos.

A análise dos dados coletados envolve técnicas de extração de quadros de vídeo para reconstrução das trajetórias. Neste processo, ferramentas como o *MATLAB* e o *FFmpeg* desempenharam um papel crucial, proporcionando uma integração eficiente entre a coleta dos dados e sua análise quantitativa. A combinação dessas ferramentas permitiu não apenas mapear com precisão a posição do robô, mas também extrair informações sobre sua velocidade e interação com as bordas do sistema experimental.

Para avaliar a eficácia do robô e validar o sistema experimental, foi conduzido um processo de calibração que utilizou como referência o BC. Este sistema oferece uma base sólida para identificar desvios no comportamento do robô, garantindo que os dados coletados reflitam com precisão as condições experimentais esperadas. A análise das reflexões e da linearidade das trajetórias serviu como critério para ajustar os parâmetros do robô e validar sua performance antes da aplicação em cenários mais complexos.

Este capítulo está organizado em etapas que estruturam a fundamentação experimental e analítica deste trabalho. Na primeira seção, são descritos os dispositivos de captura, as configurações técnicas e as condições experimentais adotadas para a aquisição dos dados, garantindo uma visão clara dos parâmetros que sustentam a coleta de informações. Em seguida, é apresentado o processo de tratamento dos dados, com destaque para as ferramentas computacionais e algoritmos utilizados, enfatizando a adequação e a precisão das metodologias aplicadas. Na sequência, discute-se a calibração do sistema, complementada pela análise dos resultados preliminares que orientaram o refinamento do experimento. Por fim, aborda-se a geração de dados voltados à análise da dinâmica e à mensuração do EL, consolidando as bases para o estudo aprofundado dos fenômenos caóticos e não-lineares, explorados nos capítulos subsequentes.

3.1 Procedimentos de Obtenção de Dados

Os procedimentos para a coleta de dados foram desenvolvidos com base na necessidade de registrar, de forma precisa e consistente, os movimentos do robô no ambiente experimental. Esta seção detalha os aspectos técnicos e operacionais envolvidos na aquisição das informações, incluindo a instrumentação empregada, as configurações adotadas para o registro das imagens e as estratégias implementadas para assegurar a qualidade e a integridade dos dados. A coleta de dados foi realizada utilizando uma câmera de alta resolução, configurada para capturar as trajetórias do robô em condições controladas de iluminação. Além disso, o espaço experimental foi organizado de forma a minimizar interferências externas e otimizar a definição dos elementos capturados. As subseções seguintes descrevem, de forma detalhada, os equipamentos utilizados e os parâmetros técnicos definidos para a aquisição, bem como as características principais dos dados obtidos durante os experimentos.

3.1.1 Instrumentação e Configuração Experimental

A principal ferramenta utilizada no experimento foi a câmera **GoPro HERO 9 Black** [63]. Este equipamento oferece recursos de captura de imagem e vídeo, proporcionando boa resolução e ampla flexibilidade na definição do campo de visão. Dotada de um sensor *CMOS* de 23,6 MP, a câmera disponibiliza uma variedade de modos de gravação, incluindo 5K a 30 fps, 4K a 60 fps, 2,7K a 120 fps e 1080p (Full HD) a 240 fps. Essas opções permitem ajustar detalhamento, taxa de quadros e qualidade visual, adequando a configuração a diferentes demandas experimentais – desde a captura de movimentos sutis até a análise de eventos em alta velocidade. A óptica da **GoPro HERO 9 Black** incorpora uma lente super grande angular, oferecendo modos de campo de visão como **SuperView**, **Wide**, **Linear** e **Narrow**. Em termos de conectividade, o equipamento

conta com *Wi-Fi*, *Bluetooth* e porta *USB-C*, recursos que simplificam o controle remoto (via aplicativo ou comando de voz), bem como a transferência rápida de arquivos. A capacidade de armazenamento foi reforçada com a utilização de um cartão microSD de 128 GB, garantindo amplo espaço para registros prolongados em alta resolução. A alimentação é fornecida por uma bateria removível de 1720 mAh, possibilitando cerca de 2 h e 30 min de gravação contínua, dependendo dos ajustes empregados. O corpo compacto, com dimensões de 71,8 mm × 55,8 mm × 33,6 mm e peso de 158 g, facilita a fixação em suportes, tripés ou estruturas experimentais. Essas características tornam a **GoPro HERO 9 Black** uma escolha ideal para o estudo da trajetória do robô, assegurando alta definição, confiabilidade na coleta dos dados e fidelidade na representação dos movimentos, resultando em análises mais rigorosas e interpretações mais precisas dos fenômenos investigados.

3.1.2 Descrição dos Dados Obtidos

Para a aquisição dos dados de cada experimento, optou-se pela configuração em 1080 p (Full HD) a 30 fps, durante um período de 30 min, utilizando o modo **Narrow** *(estreito)*. Embora ângulos mais amplos sejam vantajosos para capturar o máximo de informação, eles podem introduzir distorções do tipo *"olho de peixe"* nas bordas da imagem. Por sua vez, o modo **Narrow** minimiza significativamente essas distorções, preservando a geometria original dos elementos na cena e garantindo maior fidelidade visual, em troca de menor campo de visão. Tal característica é essencial em análises que demandam precisão no delineamento de trajetórias e movimentos, assegurando resultados mais acurados e confiáveis.

3.2 Tratamento de Dados

A partir das gravações realizadas durante os experimentos, adotamos um fluxo sistemático de tratamento de dados que envolveu duas etapas principais: a extração de quadros dos vídeos e a análise das imagens para localizar a posição do robô. Essas etapas foram conduzidas com ferramentas específicas, como o *FFmpeg* e o *MATLAB*, permitindo a extração eficiente das informações relevantes para os objetivos do estudo.

Na primeira etapa, o *FFmpeg*, uma ferramenta amplamente reconhecida por sua versatilidade no processamento de mídias digitais, foi utilizado para converter os vídeos gravados em sequências de imagens. Esse processo foi automatizado por meio de *scripts*, que garantiram a extração de frames em escala de cinza e sua organização sequencial, facilitando o armazenamento e a manipulação subsequente dos dados. Essa abordagem não apenas otimizou o uso de espaço em disco, mas também acelerou as etapas subsequentes. Na segunda etapa, o *MATLAB* foi empregado para analisar as imagens extraídas

e identificar a posição do robô em cada quadro. A localização foi determinada a partir dos picos de intensidade luminosa gerados pela fonte de luz central (LED) instalada no robô. Para isso, foi utilizada uma máscara circular que limitou a análise à região de interesse, reduzindo interferências externas. As informações extraídas foram organizadas em um formato adequado para a reconstrução da trajetória do robô e para futuras análises quantitativas e qualitativas.

3.2.1 FFmpeg

O *FFmpeg* é uma biblioteca de código aberto, ele suporta vários formatos multimídia, tornando-se essencial para profissionais que precisam manipular mídias digitais. Suas funcionalidades abrangem desde conversões de formatos até operações mais complexas, como compressão, edição, captura e streaming de conteúdos multimídia. Adicionalmente, ele é capaz de extrair informações detalhadas sobre os arquivos, como metadados, taxas de quadros e resoluções. Uma das principais características do *FFmpeg* é sua interface baseada na linha de comando, que permite executar operações sofisticadas com comandos diretos e scripts automatizados.

Para o tratamento de dados, é necessário coletar todos os frames dos vídeos gravados pela GoPro. Como a GoPro utiliza um cartão SD que, por questões de compatibilidade, geralmente é formatado em `FAT`, os vídeos são divididos em arquivos de até 4 GB. Dessa forma, torna-se imprescindível converter todos os vídeos de um experimento em frames organizados em ordem cronológica. Com o *FFmpeg* o processo é automatizado por meio de um *script* em Bash, que realiza a criação de um diretório para armazenar as imagens e executa um *loop* para processar múltiplos vídeos de forma sequencial. O comando principal do *script* é:

```
1 mkdir JPEG
2 for i in *.MP4; do
3     FFmpeg -i "$i" -vf "scale=iw:ih,format=gray" "JPEG/${i%.MP4}-%07d.
          jpg"
4 done
```

A seguir, os elementos do comando são detalhados:

- `mkdir JPEG`: cria o diretório `JPEG`, onde serão armazenadas as imagens extraídas.
- `for i in *.MP4; do ... done`: percorre todos os arquivos com extensão `.MP4` presentes no diretório.
- `FFmpeg -i "$i-vf "scale=iw:ih,format=grayJPEG/${i%.MP4}-%07d.jpg"`: este comando extrai os frames de cada vídeo e os salva como imagens em preto e branco no diretório `JPEG`.

Os parâmetros do comando *FFmpeg* possuem as seguintes funções:

- `scale=iw:ih`: mantém as dimensões originais do vídeo, onde `iw` e `ih` representam a largura e a altura, respectivamente.
- `format=gray`: converte os frames para um formato em escala de cinza (*grayscale*), que reduz significativamente o tamanho das imagens. Essa abordagem não apenas acelera o processamento dos dados subsequente, mas também torna o processamento pelo próprio *FFmpeg* mais rápido, dado que a conversão para preto e branco é menos computacionalmente intensiva do que o formato `RGB`.
- `%07d`: numera as imagens extraídas com 7 dígitos, assegurando a correta ordenação mesmo para um grande número de frames.

3.2.2 MATLAB

O *MATLAB* (*Matrix Laboratory*) é um ambiente de programação utilizado em engenharias e ciências. Ele foi projetado para facilitar operações matriciais, análise numérica, visualização de dados e desenvolvimento de algoritmos. Seu ambiente integrado combina ferramentas de edição de código, depuração e visualização em uma interface amigável, permitindo que usuários desenvolvam soluções de maneira eficiente [64].

O *MATLAB* também oferece uma vasta coleção de *toolboxes* especializadas, como as de processamento de imagem, aprendizado de máquina, controle de sistemas e simulação. Essas extensões tornam o *MATLAB* uma boa escolha para resolver problemas específicos em diversos campos de pesquisa e desenvolvimento. Outro ponto forte do *MATLAB* é a sua capacidade de criar gráficos e visualizações de alta qualidade, o que o torna ideal para análises exploratórias e apresentação de resultados.

Embora existam alternativas como *Python* e *C*, o *MATLAB* destaca-se pela sua facilidade de uso e versatilidade. No contexto deste experimento, sua escolha foi motivada pelo fato de que o programa necessário para a execução já havia sido desenvolvido previamente pelo nosso grupo de pesquisa. Além disso, o *MATLAB* foi utilizado com sucesso em experimentos anteriores, incluindo o estudo no Bilhar de Bunimovich [1], consolidando-se como uma ferramenta confiável e eficiente para este tipo de aplicação.

Funcionamento do Programa de Rastreamento do Robô

O programa de rastreamento do robô processa uma sequência de imagens capturadas no bilhar para determinar a posição do robô em cada quadro do vídeo. Ele opera seguindo etapas principais bem definidas: Primeiramente, as imagens são carregadas a partir de um diretório especificado, a pasta de saida do *FFmpeg*, e uma máscara circular é aplicada para limitar a análise à área de interesse, a região onde o robô se movimenta. Essa máscara reduz a interferência de ruídos externos e otimiza o processamento ao focar exclusivamente

na área relevante. Para cada imagem, o programa identifica a posição do robô com base nos picos de intensidade luminosa dentro da região delimitada, não necessitando das cores. Esse procedimento é realizado por meio de uma janela de busca ajustável, que considera a posição estimada do robô no quadro anterior. Essa abordagem garante eficiência no rastreamento, mesmo em situações de movimento suave ou deslocamento limitado entre quadros. Após localizar a região de maior intensidade luminosa, o programa calcula as coordenadas médias do robô e os desvios padrão correspondentes. Essas métricas são usadas para determinar a posição central do robô e avaliar a dispersão dos dados, fornecendo informações sobre o deslocamento. Os resultados são armazenados em coordenadas, permitindo análises posteriores. Além disso, o programa pode gerar gráficos opcionais que mostram a evolução do rastreamento, incluindo a trajetória do robô ao longo do tempo e a região alvo identificada em cada quadro, com custo alto em seu processamento. Por fim, os dados de posição e erro em pixeis, são exportados em arquivos `.txt`, assegurando que os resultados possam ser acessados e analisados em outras etapas de análise. O tempo de processamento é eficiente, equivalendo aproximadamente ao tempo de gravação, com uma taxa de 30 quadros por segundo. É importante destacar a necessidade de gravar os experimentos em condições de baixa luminosidade. Como o robô possui uma fonte de iluminação localizada em seu centro, A configuração do robô facilita a identificação e o rastreamento durante o processamento, minimizando interferências externas e otimizando a precisão do programa.

3.3 Calibração

Com a construção do robô concluída, a estruturação da fronteira do bilhar finalizada e a capacidade de aquisição de dados plenamente estabelecida, o próximo passo consiste em avaliar a eficiência do robô na execução dos experimentos. Para isso, três aspectos fundamentais foram analisados:

1. **Detecção da fronteira**: Verificou-se a capacidade do sistema de identificar eventos de colisão com precisão, assegurando que nenhuma interação relevante fosse perdida durante o experimento.

2. **Deslocamento em linha reta**: Avaliou-se a habilidade do robô de manter trajetórias retilíneas, um critério de alta importância na construção e na calibração do sistema.

3. **Reflexões especulares**: Confirmar que o ângulo de reflexão do robô corresponde ao ângulo de incidência, para caracterizar reflexões próximas à ideal.

A calibração inicial foi realizada no BC, devido à sua dinâmica conservar o ângulo de incidência, em outras palavras, o momento angular é conservado. Assim, a escolha do

BC ($\gamma = 0$) para iniciar os experimentos não apenas facilita a análise das reflexões, mas também fornece uma base confiável para a validação do sistema antes de transitar para experimentos com presença de caos.

Dadas as dificuldades iniciais na construção e programação do robô, foi desenvolvido um software em *Python* chamado `Validação_do_Limão.py`. Esse programa foi projetado especialmente para avaliar quantitativamente o desempenho do robô e verificar sua precisão em relação aos critérios estabelecidos. Nas subseções a seguir, detalha-se a aplicação do programa e os resultados obtidos na avaliação do robô.

3.3.1 Colisão

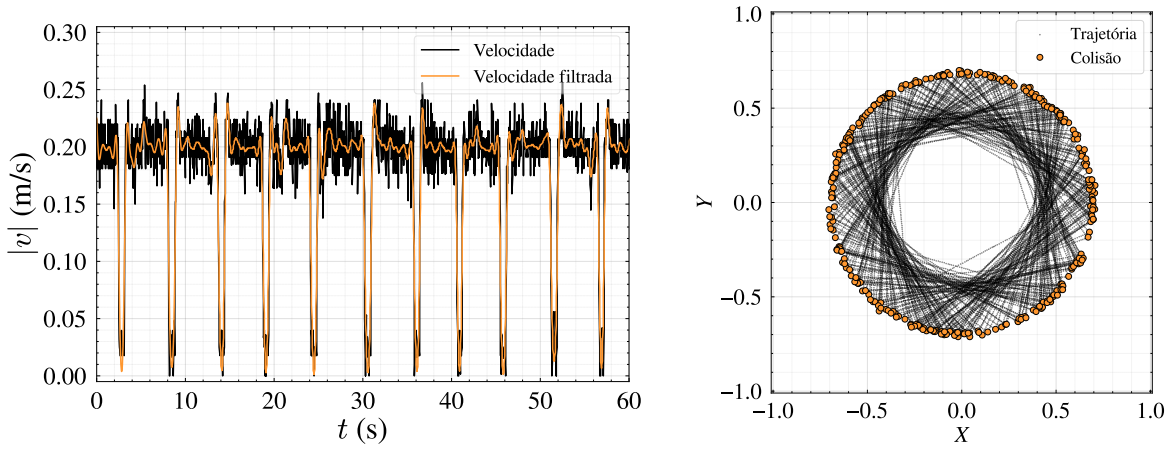

Figura 3.1: **Esquerda:** Gráfico com o primeiro minuto da velocidade. A curva laranja representa a velocidade filtrada utilizando um filtro passa-baixa. **Direita:** A trajetória do robô é representada por pontos pretos, com cada ponto correspondendo a um frame da captura, totalizando aproximadamente 54.000 pontos ao longo do experimento. As colisões detectadas são destacadas por pontos laranja de maior tamanho, com contorno preto, indicando os eventos identificados durante o movimento.

Para determinar a velocidade, utilizamos os dados extraídos das imagens capturadas pela câmera. Inicialmente, a trajetória do robô foi identificada e a taxa de captura da câmera foi considerada. Em seguida, as dimensões da trajetória, originalmente expressas em pixeis, foram normalizadas para unidades métricas. A velocidade foi calculada como a razão entre o deslocamento espacial e o intervalo de tempo correspondente aos pontos consecutivos. À esquerda da Fig. 3.1, observa-se a evolução da velocidade ao longo do primeiro minuto do experimento. A curva laranja representa a velocidade filtrada por um filtro passa-baixa, destacando as flutuações mais suaves e facilitando a identificação das quedas bruscas de velocidade, que correspondem aos momentos de colisão. A detecção

de colisões está diretamente relacionada com sua velocidade, quando o robô detecta a borda, o sistema interrompe o movimento, calcula os parâmetros necessários e realiza uma reflexão que consiste em uma rotação em torno de seu próprio eixo. Durante essa rotação, o LED centralizado tende a permanecer fixo em sua posição, sem apresentar deslocamento aparente. Esse comportamento é identificado como uma colisão na fronteira efetiva, detectada pela queda brusca na velocidade, e a posição correspondente é registrada para análise. Na Fig. 3.1 à direita é possível ver os pontos de colisões, demonstrando a eficacia do programa, ao observar que todas as colisões estão de fato nas bordas do bilhar.

3.3.2 Voo Livre

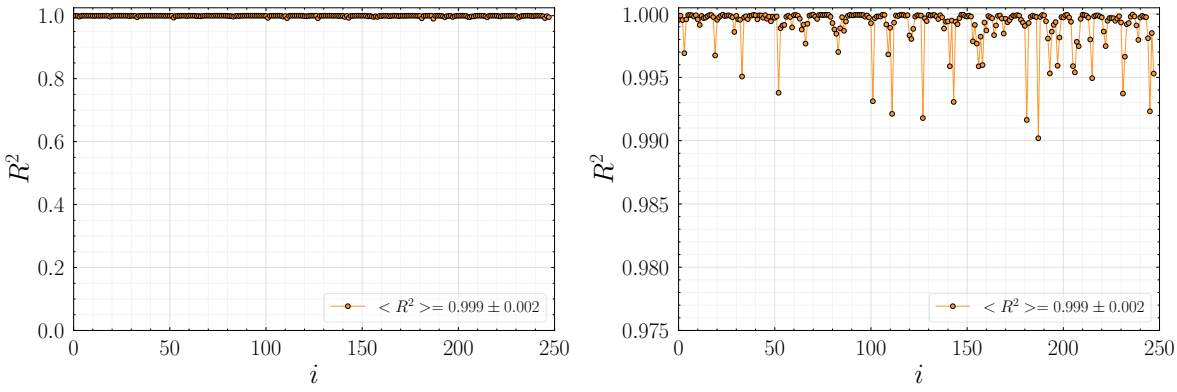

Figura 3.2: **Esquerda:** Representação de 250 trajetórias retilíneas com R^2 calculados para cada uma. O eixo y varia de 0 a 1, evidenciando que todos os R^2 estão muito próximos de 1, indicando alta consistência na linearidade das trajetórias. **Direita:** A mesma figura apresentada à esquerda, mas com o eixo y limitado ao intervalo de 0,975 a 1, permitindo uma visualização detalhada das pequenas variações em R^2.

Entre cada colisão, durante o voo livre (*Trajetória entre as colisões*) do robô, o sistema é programado para manter a rotação das rodas constante, conforme detalhado no Cap. 2. Esse controle é realizado monitorando a quantidade de pulsos por segundo que os encoders de cada roda devem registrar, o que permite regular a velocidade do robô a aproximadamente 0,2 m/s. Essa estabilidade na velocidade pode ser observada na Fig. 3.1, onde as variações ao longo do tempo são aceitáveis, evidenciando a eficácia do controle implementado para garantir que o robô se desloque de forma constante e em linha reta durante os intervalos entre colisões.

Para verificar se o deslocamento do robô está alinhado com o ideal, adotamos um método baseado na análise da trajetória. Inicialmente, o programa traça uma linha reta entre os pontos de colisões consecutivas, assumindo que o deslocamento ideal seria per-

feitamente alinhado com essa linha. A análise quantitativa desse alinhamento é realizada por meio do **coeficiente de determinação** (R^2), que avalia a proporção da variabilidade na posição do robô explicada por essa trajetória ideal. O R^2 é definido matematicamente como:

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}}, \tag{3.1}$$

onde SSE (Soma dos Quadrados dos Erros) representa a variabilidade não explicada pelo modelo, calculada como $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, e SST (Soma Total dos Quadrados) mede a variabilidade total dos dados em relação à média, dada por $\sum_{i=1}^{n}(y_i - \bar{y})^2$. Nessa fórmula, y_i são as posições reais do robô, $\hat{y}_i$ são as posições previstas pela linha ideal, e $\bar{y}$ é a média das posições observadas. Valores próximos a 1 de R^2 indicam que o robô se deslocou de maneira consistente e alinhada com a trajetória ideal, enquanto valores baixos sugerem desvios possivelmente causados por imperfeições no sistema, como diferenças na tração das rodas ou erros de calibração dos sensores. Esse método fornece uma métrica robusta para avaliar a precisão do deslocamento do robô e ajustar o sistema de controle, garantindo a confiabilidade do movimento retilíneo entre as colisões. Conforme ilustrado na Fig. 3.2, o valor obtido de $R^2 = 0{,}999 \pm 0{,}002$ para o experimento demonstra que a trajetória do robô está alinhada com o ideal, com pequenos desvios atribuíveis a ruídos experimentais. Essa análise confirma a eficácia do controle implementado e permite uma avaliação quantitativa da qualidade dos voos.

3.3.3 Reflexão

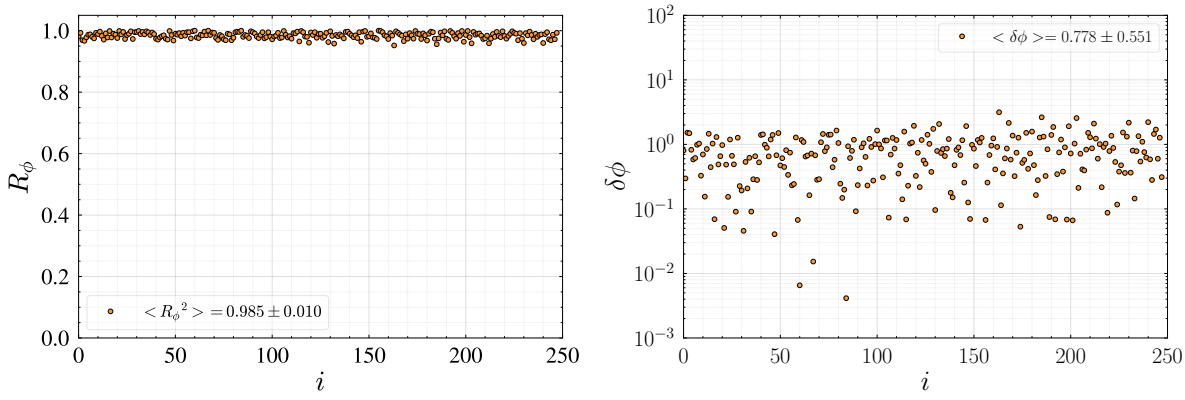

Figura 3.3: **Esquerda:** R_ϕ em função do número i da colisão. Valores próximos a 1 indicam reflexões especulares. **Direita:** Diferença entre o ângulo de colisão atual e o ângulo da colisão anterior em graus ($\delta\phi = |\phi_i - \phi_{i-1}|$), avaliando a consistência do sistema durante o experimento.

A análise da reflexão é fundamental para verificar se o robô mantém um comportamento especular ao longo do experimento, o que é essencial para garantir a validade dos resultados. Com a capacidade de detectar as bordas do bilhar de forma consistente e sincronizar a velocidade das rodas, torna-se necessário avaliar a precisão das reflexões. Essa análise justifica a escolha de um BC para a calibração e inicialização dos experimentos. Na Fig.3.3, à esquerda, observa-se a razão R_ϕ entre o maior e o menor ângulo entre os de incidência e reflexão ao longo de aproximadamente 250 a 350 colisões. Valores médios de 0,985 $\pm$0,010 indicam que as reflexões do robô são próximas ao comportamento especular, sem desvios significativos. Já na Fig. 3.3, à direita, é apresentada a diferença entre os ângulos de colisões consecutivas. Essa análise destaca a constância da reflexão, evidenciando que, em um BC, a variação teórica do ângulo deve ser nula. Os dados experimentais corroboram essa expectativa, apresentando uma média de 0,778 $\pm$ 0,551°, com variações mínimas. Esses resultados são fundamentais para validar o robô em experimentos mais complexos, uma vez que erros grosseiros de reflexão poderiam comprometer a análise de dinâmicas caóticas. Portanto, a utilização de um BC no experimento inicial permite garantir a ausência de caos e avaliar detalhadamente o desempenho do robô. Para fins de ajustes, foram configurados dois bilhares distintos: o BC, que serve para calibração, e o bilhar em formato de limão, utilizado para os experimentos finais.

3.4 Geração de Dados para Análise da Dinâmica

Para extrair os dados, utilizamos um programa denominado `Lemon.py` para constituir o núcleo central das análises realizadas. Após a coleta dos dados no *MATLAB*, utilizamos amplamente o *Python* para processar e tratar os dados obtidos. Inicialmente, é feita a correção do ângulo de gravação. Durante a configuração experimental, buscou-se posicionar a *GoPro* paralela ao piso, garantindo que a construção do bilhar também apresentasse quinas alinhadas ao plano horizontal. Este alinhamento foi implementado para minimizar desvios angulares indesejados, conforme descrito em 2.5.1. Apesar de ser um procedimento mais trabalhoso, este método assegura que as quinas do bilhar fiquem paralelas às linhas do piso, como ilustrado na Fig. 2.15. Essa abordagem reduz significativamente a diferença angular entre a câmera e o BL, mas não a elimina completamente. Para sistemas caóticos, em que o robô percorre trajetórias próximas às quinas do Bilhar e explora quase todo o contorno ao longo do experimento, é possível utilizar os pontos registrados para determinar e corrigir a diferença angular entre a *GoPro* e o BL. Essa correção é feita aplicando uma rotação aos dados, zerando a discrepância angular.

Após essa etapa, procede-se à centralização dos dados em torno da média dos eixos X e Y, seguida da normalização das unidades de pixel para metros. A centralização é realizada calculando-se o centro de massa dos pontos e ajustando os dados para que este coincida com a origem. Para a normalização, utilizamos as dimensões conhecidas do piso,

cuja aresta apresenta aproximadamente 532 pixeis, equivalentes a 112 cm. Essa relação fornece um fator de conversão preciso, permitindo a normalização espacial dos dados.

Após corrigir a diferença angular, as colisões são identificadas pelas quedas na velocidade, conforme ilustrado na Fig. 3.1, à esquerda. Uma vez extraídas todas as colisões, utilizamos esses dados para estimar os parâmetros dos círculos associados aos conjuntos superior e inferior do BL. Para essa análise, empregamos o método de ajuste `curve_fit`, disponível no módulo `scipy.optimize`. Com os parâmetros ajustados, obtemos as características geométricas dos círculos, incluindo a distância entre os centros, suas posições e seus respectivos raios. Essas informações permitem descrever de forma precisa os parâmetros do BL, fundamentando-se nas equações (2.1–2.4).

Não menos importante, ao determinar os centros dos círculos inferior e superior, torna-se possível calcular o vetor normal, que aponta sempre na direção do centro do círculo correspondente. Quando a colisão ocorre na parte inferior do BL, o vetor normal aponta para o centro deste círculo, como ilustrado na Fig. 1.10, à esquerda. Com essas informações, é possível determinar os ângulos de incidência e reflexão a partir de três vetores principais: o vetor de incidência, o vetor de reflexão e o vetor normal. Utilizando operações vetoriais, determinamos os ângulos de incidência e reflexão para todas as colisões, exceto a primeira e a última, por não existir a colisão anterior ou a posterior.

Por fim, o programa salva quatro arquivos `.txt` contendo os dados processados. O primeiro arquivo, denominado `(nome_do_arquivo)_traj.txt`, armazena a trajetória normalizada e centralizada, sendo útil para a plotagem dos pontos em pretos sem contorno, como ilustrado na Fig. 3.1. O segundo arquivo, `(nome_do_arquivo)_Colisão.txt`, registra as localizações das colisões, representadas pelos contornos pretos na mesma figura. O terceiro arquivo, `(nome_do_arquivo)_(Per_SinTheta).txt`, é utilizado para a construção do MP, como mencionado anteriormente, com base nas CB, que são essenciais para as análises e revelam a estrutura do EF. Por fim, o quarto arquivo, `(nome_do_arquivo)_(T_Th_Ka_V).txt`, contém, os parâmetros necessários para o calculo do expoente de Lyapunov, detalhado na seção a seguir.

3.5 Obtenção do Expoente de Lyapunov

O Expoente de Lyapunov (EL) (λ) é a medida quantitativa para avaliar a sensibilidade às CI em sistemas dinâmicos. Ele reflete o comportamento de trajetórias próximas no EF e como a separação entre as trajetórias divergem ao longo do tempo. Sendo particularmente relevante no atual estudo, o cálculo de λ foi definido pela expressão:

$$\lambda = \lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{n-1} \ln \frac{||\delta_{i+1}||}{||\delta_i||}, \tag{3.2}$$

em que $||\delta_{i+1}||$ representa a norma do vetor de separação entre duas trajetórias no instante $i+1$ e $||\delta_i||$ é a norma no instante i.

O cálculo do EL começa com a definição do vetor de separação inicial $\delta_0 = [\delta_\ell, \delta_v]^\top$, cujos componentes são escolhidos aleatoriamente e, em seguida, normalizados para garantir que $||\delta_0|| = 1$. Esse vetor inicializado será submetido a transformações sucessivas que refletem a evolução do sistema dinâmico. Em cada passo de tempo, o vetor δ é atualizado por meio de matrizes de evolução, sendo normalizado após cada passo para evitar divergências.

A dinâmica das separações é modelada por duas matrizes. A primeira, ($\mathbf{J}_o$), descreve o voo livre entre duas colisões num intervalo de tempo τ, sendo representada por:

$$\mathbf{J}_o = \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix}. \tag{3.3}$$

Sua aplicação ao vetor δ resulta em: $\delta' = \mathbf{J}_o \cdot \delta$, o que, explicitamente, implica em:

$$\delta'_\ell = \delta_\ell + \tau \cdot \delta_v, \quad \delta'_v = \delta_v. \tag{3.4}$$

Essa matriz representa um movimento retilíneo uniforme, na qual δ_ℓ recebe um acréscimo de $\tau\delta_v$, enquanto δ_v permanece inalterado.

Em seguida, a matriz $\mathbf{J}_c$, responsável por capturar os efeitos das reflexões especulares, é aplicada ao vetor transformado. A matriz $\mathbf{J}_c$ é expressa como:

$$\mathbf{J}_c = -\begin{bmatrix} 1 & 0 \\ \frac{2\cdot\kappa}{\cos(\phi)} & 1 \end{bmatrix}, \tag{3.5}$$

onde κ representa a curvatura local e ϕ o ângulo da reflexão. A aplicação de $\mathbf{J}_c$ ao vetor δ' resulta em, $\delta'' = \mathbf{J}_c \cdot \delta'$, o que leva a:

$$\delta''_\ell = -\delta'_\ell, \quad \delta''_v = -\left(\frac{2\cdot\kappa}{\cos(\phi)} \cdot \delta'_\ell + \delta'_v\right). \tag{3.6}$$

A transformação total ao longo de um passo temporal é obtida pela combinação dessas duas matrizes, resultando em, $\mathbf{J} = \mathbf{J}_c \cdot \mathbf{J}_o$. Substituindo-se as expressões de $\mathbf{J}_c$ e $\mathbf{J}_o$, tem-se:

$$\mathbf{J} = -\begin{bmatrix} 1 & \tau \\ \frac{2\cdot\kappa}{\cos(\phi)} & \frac{2\cdot\kappa}{\cos(\phi)} \cdot \tau + 1 \end{bmatrix}. \tag{3.7}$$

Esta matriz descreve a evolução completa do vetor δ em um único passo, considerando os efeitos acumulados do voo livre e da colisão.

Ao longo de n passos, o vetor δ evolui sucessivamente, com a matriz $\mathbf{J}$ adaptada a cada passo em função dos parâmetros κ_i, ϕ_i e τ_i. Assim, a evolução temporal completa é

expressa por:

$$\delta_{\text{final}} = \mathbf{J}_n \cdot \mathbf{J}_{n-1} \cdots \mathbf{J}_1 \cdot \delta_0. \tag{3.8}$$

Em cada passo, a norma do vetor δ é calculada como:

$$||\delta_i|| = \sqrt{\delta_{\ell,i}^2 + \delta_{v,i}^2}, \tag{3.9}$$

e seu logaritmo natural é calculado para estimar o crescimento exponencial. Após o cálculo da norma, o vetor δ_i é renormalizado para evitar crescimento excessivo durante os cálculos subsequentes.

Por fim o programa mensura o expoente médio de Lyapunov, que é determinado pela expressão:

$$\lambda = \frac{1}{T} \sum_{i=1}^{n} \ln ||\delta_i||,$$

em que $T = \sum_{i=1}^{n} \tau_i$ é o tempo total considerado entre cada colisão. Este cálculo fornece uma caracterização precisa da sensibilidade do sistema às CI e é amplamente discutido em [1, 10].

Capítulo 4

Resultados e Discussão

Após a construção do robô e a validação de sua capacidade de locomoção retilínea entre colisões, bem como sua habilidade de realizar reflexões especulares, foram conduzidos os experimentos no bilhar. O intervalo de análise foi definido como $\gamma \in [0, 0{,}5]$. Os resultados apresentados a seguir foram obtidos utilizando o programa descrito nas seções 3.4 e 3.5. Para a interpretação dos dados, também foi empregada a métrica da *Recurrence Rate* ou Taxa de Recorrência (RR), cuja definição e aplicação serão detalhadas posteriormente. Como o robô não toca diretamente a parede, o contorno utilizado é fictício, sendo este usado para calcular o parâmetro experimental γ. Para diferenciá-lo, denominamos o parâmetro calculado como *gamma efetivo* (γ_{eff}). Nos gráficos apresentados, serão exibidos tanto os dados experimentais quanto a fotografia correspondente a cada experimento. O contorno efetivo, representado por um contorno branco, será sobreposto aos dados, como exemplificado na Fig. 4.1.

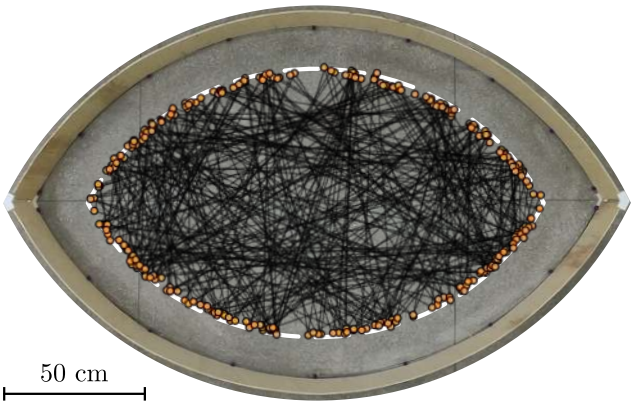

Figura 4.1: Dados de trajetória (em preto) e colisões (em laranja), sobrepostos à fotografia do experimento correspondente ao $\gamma_{\text{eff}} \simeq 0{,}5$.

4.1 Validação dos Dados Experimentais

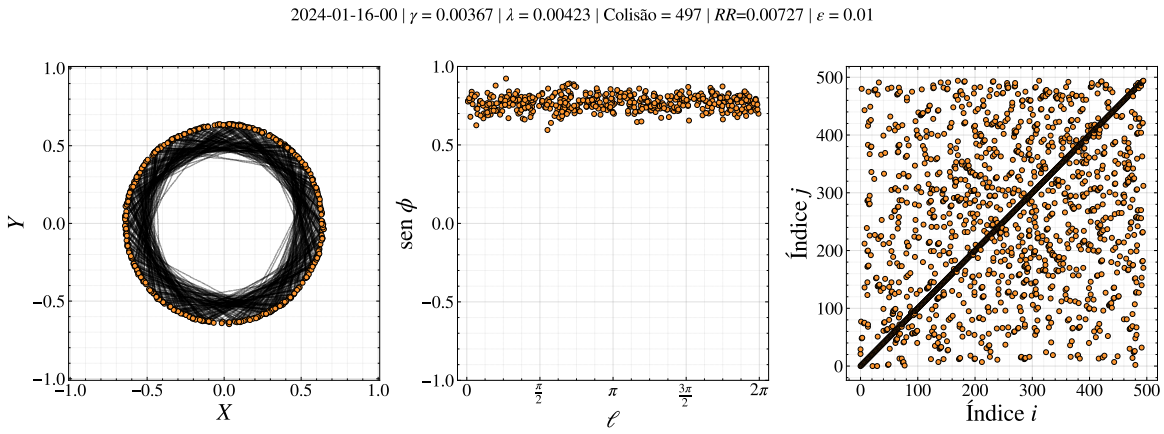

Figura 4.2: Para $\gamma_{\text{eff}} \simeq 0$, da esquerda para a direita: A trajetória representada em preto e as colisões em laranja, o MP do experimento, e sua matriz de recorrência associada.

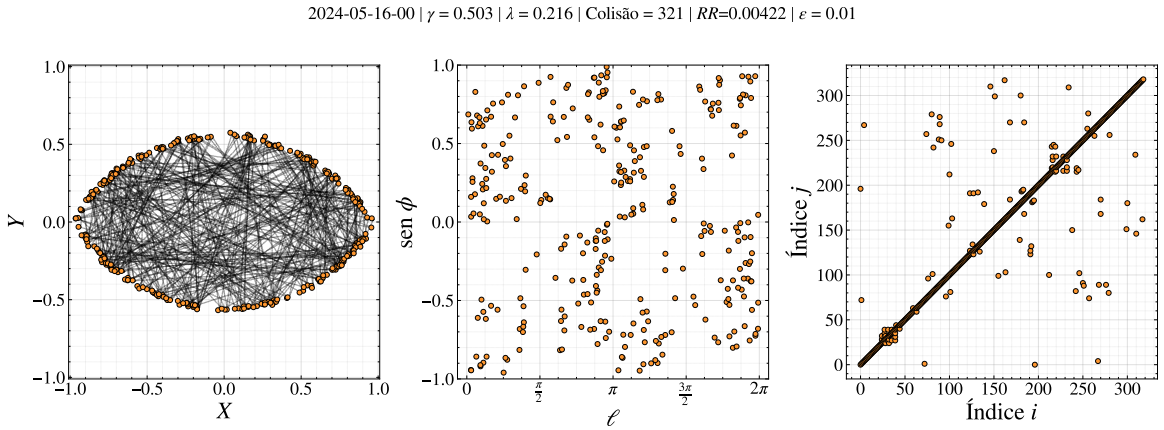

Figura 4.3: Para $\gamma_{\text{eff}} \simeq 0{,}5$, da esquerda para a direita: A trajetória representada em preto e as colisões em laranja, o MP do experimento, e sua matriz de recorrência associada.

A Fig. 4.2 apresenta um exemplo típico dos dados obtidos a partir das trajetórias do sistema. Os gráficos incluem as trajetórias completas, pontos de colisão, o MP e a matriz da recorrência, considerando o parâmetro ε avaliado para $\varepsilon = 0{,}01$. Estão indicados a data do experimento e os dois últimos dígitos da ordem sequencial do experimento, que começam em 00. Adicionalmente, destaca-se o valor de γ_{eff}, associado à mensuração dos círculos, o EL, o número total de colisões registradas e a RR. Estes dados fornecem uma visão abrangente sobre o comportamento da dinâmica e servem de base para as análises subsequentes.

Para $\gamma_{\text{eff}} \simeq 0$, observa-se que o EL converge a muito próximo de zero, mesmo que o comportamento do sistema não seja exatamente regular. Neste caso, $\gamma_{\text{eff}} \simeq 0{,}003$ (Fig. 4.2).

Em contraste, para $\gamma_{\text{eff}} \simeq 0{,}5$ (Fig. 4.3), o sistema apresenta características totalmente caóticas. Como discutido no Capítulo 1, o EL positivo é típico de sistemas caóticos. A comparação entre esses dois regimes é evidente ao se analisar CI próximas para $\gamma_{\text{eff}} \simeq 0$ e $\gamma_{\text{eff}} \simeq 0{,}5$. A divergência entre as trajetórias é observada na Fig. 4.4.

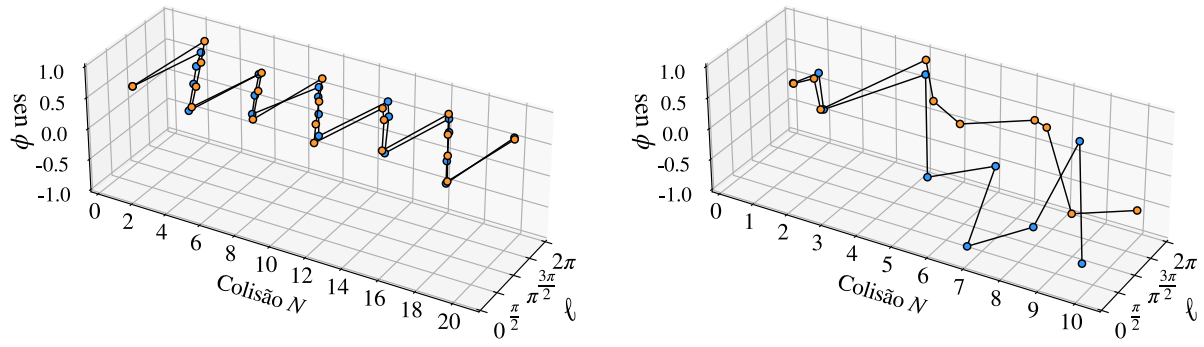

Figura 4.4: Gráficos de séries temporais discretas para duas trajetórias com CI próximas. **Painel esquerdo**: $\gamma_{\text{eff}} \simeq 0$, onde as trajetórias permanecem próximas ao longo do tempo, confirmando dinâmicas regulares ($\lambda_{\text{exp}} \simeq 0{,}02$). **Painel direito**: $\gamma_{\text{eff}} \simeq 0{,}503$, CI na região caótica, com divergência exponencial das trajetórias ($\lambda_{\text{exp}} \simeq 0{,}19$).

A Fig. 4.4 ilustra a evolução temporal de duas trajetórias com CIs próximas em dois regimes dinâmicos distintos. No painel esquerdo ($\gamma_{\text{eff}} \simeq 0$), observa-se que as trajetórias permanecem praticamente sobrepostas ao longo de 20 colisões, caracterizando um comportamento regular. A taxa de divergência é $\lambda_{\text{exp}} \simeq 0{,}02$, evidenciando a estabilidade do sistema neste regime. Por outro lado, no painel direito ($\gamma_{\text{eff}} \simeq 0{,}502$), que corresponde ao regime caótico, as trajetórias divergem rapidamente já nas três primeiras colisões. Esse comportamento demonstra a sensibilidade às CIs. Após algumas iterações, as trajetórias tornam-se visualmente indistinguíveis das CIs, refletindo a divergência exponencial e o alto valor do expoente de Lyapunov ($\lambda_{\text{exp}} \simeq 0{,}19$). Esses resultados confirmam a transição do comportamento regular para o caótico com o aumento de γ_{eff}. A análise destaca a relação entre a dinâmica do sistema e a taxa de divergência das trajetórias no EF.

4.1.1 Taxa de Recorrência

A Taxa de Recorrência (RR) é uma métrica que mede a frequência com que uma trajetória em um sistema dinâmico retorna a estados próximos de um estado anterior no EF. Ela é particularmente útil na análise de sistemas caóticos, pois permite identificar padrões recorrentes e caracterizar diferentes comportamentos dinâmicos, como a presença de *stickiness*. A RR é definida como:

$$RR = \frac{1}{N^2} \sum_{i,j=1}^{N} R_{ij}, \tag{4.1}$$

onde R_{ij} é um elemento de uma matriz binária de recorrência, que é definida como:

$$R_{ij}(\varepsilon) = \begin{cases} 1, & \text{se } \|x(t_i) - x(t_j)\| < \varepsilon, \\ 0, & \text{caso contrário,} \end{cases} \tag{4.2}$$

e ε é um limiar de distância que define o quão próximos dois estados $x(t_i)$ e $x(t_j)$ devem estar para serem considerados recorrentes. Cada entrada igual a 1 na Matriz de Recorrência representa uma recorrência da trajetória analisada, indicando que $x(t_i)$ e $x(t_j)$ estão dinamicamente ε-próximos um do outro. Como o foco está na análise de recorrência em tempo finito, é importante observar que $0 < t_i < t_j \leq N$, onde N é o tempo máximo de iteração da trajetória. A matriz de recorrências possui dimensão $N \times N$.

Na prática, a RR fornece a proporção de pontos no EF que retornam à vizinhança de estados anteriores, sendo altamente dependente do número de iterações, do limiar ε e das CIs. Essa métrica tem sido usada para identificar regiões de alta recorrência associadas a *stickiness* e para distinguir comportamentos caóticos de movimentos quase-periódicos em sistemas não lineares [19, 65].

4.2 Espaços de Fase

Nesta seção, apresentamos os resultados dos Espaços de Fase (EFs) obtidos a partir dos experimentos realizados, com destaque para diferentes valores de γ_{eff}. Cada gráfico exibe uma amostra representativa dos experimentos, escolhida dentre um total de 98 conjuntos analisados. Em cada EF, a simulação correspondente é exibida ao fundo, em tons de cinza, permitindo observar padrões gerais e possíveis características, como regiões de caos e ilhas de estabilidade. As simulações foram elaboradas e realizadas por membros do nosso grupo de pesquisa. As ilhas destacam áreas que permanecem inacessíveis na região caótica durante a simulação. Os pontos experimentais estão representados em laranja, facilitando a comparação com as simulações. Vale ressaltar que todas as simulações foram realizadas considerando até 50.000 colisões, enquanto os experimentos limitam-se a cerca de 300 colisões por experimento, devido às condições práticas e restrições experimentais.

Considerando $\gamma_{\text{eff}} \simeq 0{,}5$, a dinâmica do BL torna-se completamente caótica, caracterizando um regime onde as trajetórias não apresentam nenhum padrão regular e exibem sensibilidade exponencial às CIs. Esse regime caótico completo contrasta fortemente com os EFs mistos observados para valores de parâmetro no intervalo $0 < \gamma_{\text{eff}} < 0{,}5$. Em [36], Makino *et al.* demonstraram que, nesse intervalo, os EFs são compostos por regiões de comportamento caótico intercaladas com ilhas de estabilidade, que representam trajetórias periódicas ou quase periódicas. Essa coexistência de dinâmicas regulares e caóticas faz dos EFs mistos um tema central para o estudo de transições entre ordem e caos em sistemas dinâmicos.

4.2.1 Ilhas de Estabilidade

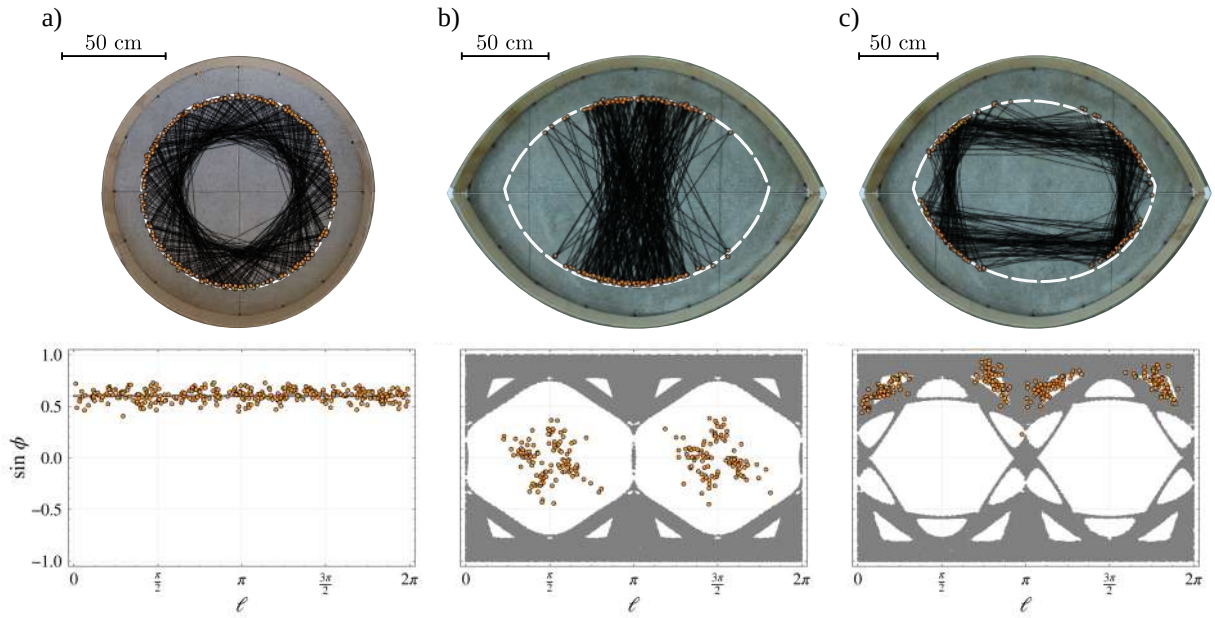

Figura 4.5: **Painel Superior:** Trajetórias experimentais (preto) com os pontos de colisões destacados em laranja. A linha tracejada branca representa a região do bilhar efetivo. **Painel inferior:** Planos de fase experimentais (laranja) e numéricos (cinza) para $\gamma_{\text{eff}} \simeq 0{,}00$, $\gamma_{\text{eff}} \simeq 0{,}24$ e $\gamma_{\text{eff}} \simeq 0{,}30$. Cada figura corresponde a um experimento distinto.

Para $\gamma_{\text{eff}} \simeq 0$, o sistema se comporta como aproximadamente um BC, exibindo movimento quase regular com momento angular aproximadamente constante. No EF, as trajetórias experimentais acompanham as previsões numéricas, com um ruído de aproximadamente 0,1 no momento angular. A Fig. 4.5a) ilustra esse regime, no qual o expoente de Lyapunov registrado é $\lambda \simeq 0{,}002$, indicando alta estabilidade. Quando γ_{eff} aumenta para valores próximos de 0,28, o sistema entra em uma região de EF misto. Neste caso, uma cadeia de duas ilhas de estabilidade é formada, como mostrado na Fig. 4.5b). As trajetórias experimentais iniciadas em uma ilha de KAM permanecem restritas a essas regiões, com um EL, $\lambda \simeq 0{,}004$. Para $\gamma_{\text{eff}} \simeq 0{,}30$, a dinâmica revela cadeias de quatro pequenas ilhas de estabilidade, evidenciadas na Fig. 4.5c). As trajetórias experimentais confirmam que o movimento se mantém dentro dessas regiões, mesmo em condições de ruído, com um EL, $\lambda \simeq 0{,}08$. Entre $\gamma_{\text{eff}} \simeq 0{,}226$ e 0,4, o EF é caracterizado por regiões caóticas intercaladas por cadeias de ilhas menores, tornando o subconjunto caótico mais acessível. Os resultados experimentais mostram excelente concordância com as previsões teóricas, mesmo em regiões de alta complexidade, validando o modelo numérico e destacando a robustez das medições realizadas.

4.2.2 Mares de Caos

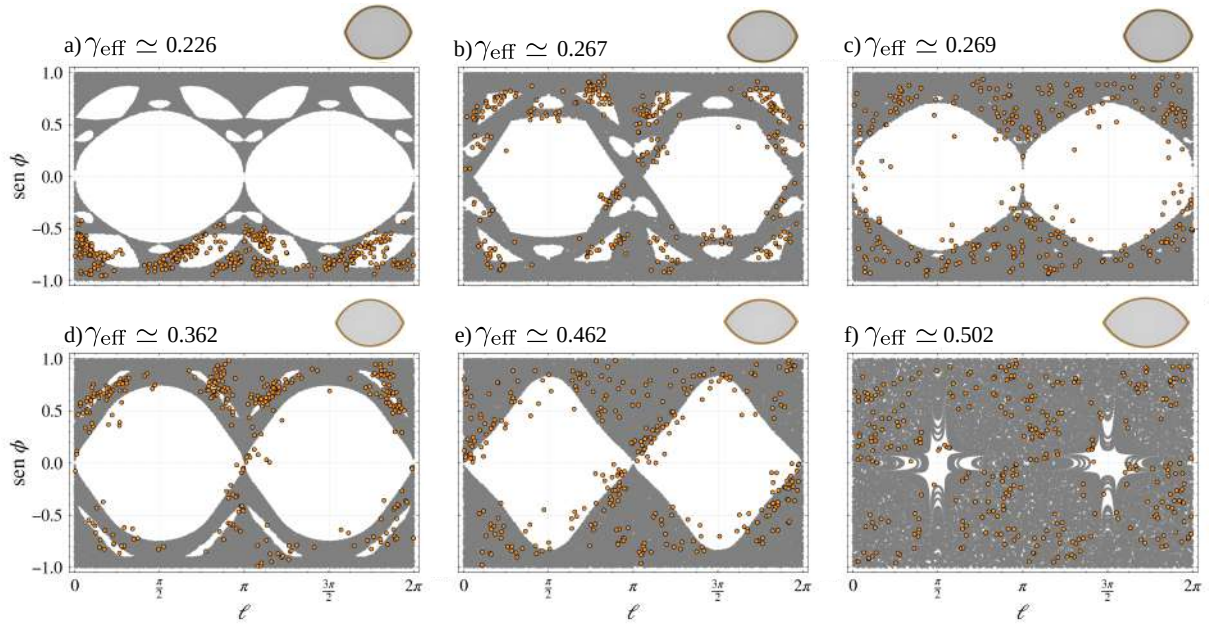

Figura 4.6: Espaços de Fase (EFs) $(\ell, \operatorname{sen}\phi)$ para diferentes valores de γ_{eff}. Pontos laranjas representam dados experimentais, enquanto pontos cinza correspondem a simulações numéricas.

Os sistemas de bilhar exibem comportamentos dinâmicos variados, evidentes nos seus EFs mistos, onde trajetórias caóticas coexistem com dinâmicas regulares. Essa natureza híbrida permite estudar transições entre ordem e caos, fornecendo contribuições sobre fenômenos não lineares. A Fig. 4.6 apresenta seis diferentes configurações de EFs $(\ell, \operatorname{sen}\phi)$ para valores crescentes de γ_{eff}, no intervalo $0 < \gamma_{\text{eff}} \lesssim 0{,}5$. No caso de $\gamma_{\text{eff}} \simeq 0{,}226$ (Fig. 4.6a)), o EF é dominado por grandes ilhas de KAM, que correspondem a regiões de movimento regular. Essas ilhas fragmentam o EF, dificultando o acesso às regiões caóticas. Os dados experimentais, representados pelos pontos laranja, alinham-se bem às previsões numéricas, com um EL de $\lambda_{\text{a}} \simeq 0{,}15$, indicando uma dinâmica caótica. Conforme γ_{eff} aumenta, como nas Figs. 4.6b)) a 4.6e), as ilhas de KAM tornam-se menores e mais espaçadas. Essa transformação facilita o acesso às regiões caóticas do EF, refletindo-se nos EL medidos: $\lambda_b \simeq 0{,}10$, $\lambda_c \simeq 0{,}07$, $\lambda_d \simeq 0{,}09$ e $\lambda_e \simeq 0{,}14$. A presença de caos mais pronunciado é evidente, embora as trajetórias regulares ainda coexistam. Finalmente, para $\gamma_{\text{eff}} \simeq 0{,}502$ (Fig. 4.6f), as ilhas de estabilidade tornam-se insignificantes, nessa condição, o EF é amplamente dominado por trajetórias caóticas, com os pontos experimentais cobrindo quase toda a extensão do plano. O EL atinge o valor de $\lambda_f \simeq 0{,}19$. Esses resultados demonstram a capacidade do nosso arranjo experimental de capturar, com precisão, a evolução das estruturas dinâmicas no EF misto. A combinação de simulações numéricas e dados experimentais oferece uma compreensão abrangente das transições entre ordem e caos em sistemas de bilhar, reforçando seu valor como plataforma para o estudo de

sistemas não lineares complexos.

4.3 Expoente de Lyapunov e Sensibilidade às Condições Iniciais

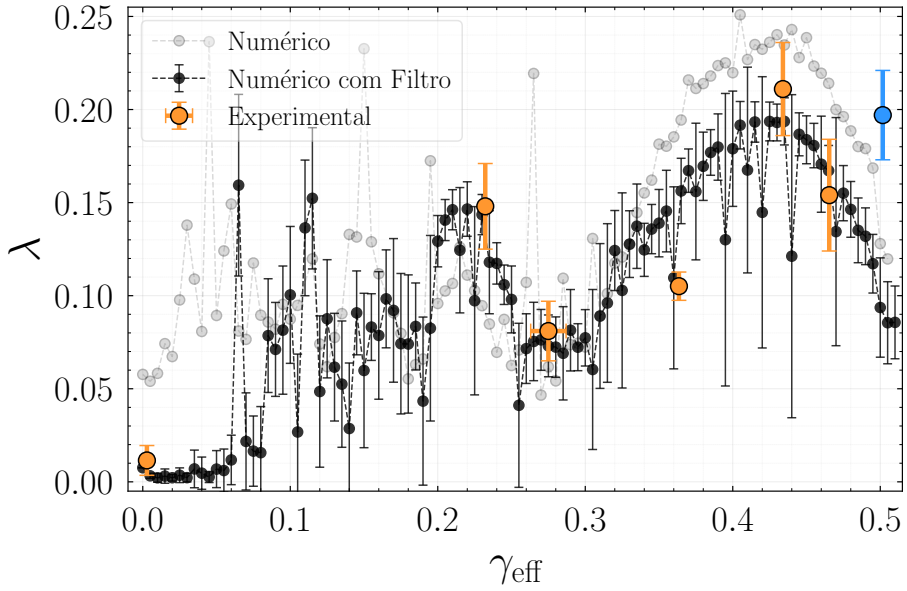

Figura 4.7: Comportamento do expoente de Lyapunov λ em função do parâmetro de controle γ_{eff}. Os pontos pretos são após a seleção para baixo *stickiness*, enquanto os cinzas são sem filtro, representam os resultados numéricos obtidos a partir de $N = 500$ colisões com a fronteira do bilhar. Os pontos laranjas representam os resultados experimentais, calculados a partir de cinco execuções experimentais com $N \simeq 350$ colisões. Observa-se boa concordância entre os resultados experimentais e numéricos, exceto para $\gamma_{\text{eff}} \simeq 0{,}5$, devido ao efeito de *stickiness*, que é ausente no arranjo experimental, mas altamente influente na abordagem numérica.

A Fig. 4.7 apresenta o comportamento do EL experimental, λ_{exp}, como função do parâmetro de controle γ_{eff} (símbolos laranjas). Os pontos pretos correspondem aos resultados numéricos λ_{num}. Como evidenciado, os resultados experimentais estão em boa concordância com as previsões numéricas. Cada valor experimental mostrado na Fig. 4.7 representa a média de cinco medições, cada uma calculada a partir de uma série temporal distinta. O comportamento de λ em função de γ_{eff} apresenta características não triviais quando comparado ao bilhar de Bunimovich [1], conforme revelado pela abordagem numérica.

O principal desafio da análise numérica foi excluir trajetórias afetadas pelo efeito de *stickiness*, fornecendo assim valores mais representativos de λ para comparação com os dados experimentais. Foram computadas $N = 500$ colisões com a fronteira do bilhar, quantidade suficiente para garantir a convergência do EL numérico. Cada ponto numérico

foi obtido a partir da média de CIs selecionadas aleatoriamente, restringindo-se apenas a trajetórias em torno da menor RR [42]. Como esperado, $\lambda_{\text{num}} \simeq 0$ para $\gamma_{\text{eff}} \simeq 0$.

A curva na Fig. 4.7 revela um comportamento intrincado, com máximos e mínimos locais, sendo o máximo global localizado em $\gamma_{\text{eff}} \simeq 0{,}43$. Para valores de $\gamma_{\text{eff}} < 0{,}1$, a elevada densidade de ilhas de estabilidade no EF contribui para um erro significativo em λ_{num}. Resultados análogos podem ser encontrados em [36, 38, 40] para outras grandezas características do bilhar.

Para $\gamma_{\text{eff}} \simeq 0{,}5$, uma discrepância notável entre os resultados experimentais e numéricos é observada (ponto azul na Fig. 4.7). Este desvio é atribuído à influência significativa do efeito de *stickiness* neste regime do EF. Quando o EF contém grandes ilhas de estabilidade, as regiões de *stickiness* ficam confinadas a camadas finas ao redor das ilhas. Por outro lado, para EFs com ilhas menores, as regiões de *stickiness* são maiores e mais acessíveis. Como ilustrado na Fig. 4.6f), regiões próximas a $(\ell, \operatorname{sen}\phi) = (\pi/2, 0)$ e $(\ell, \operatorname{sen}\phi) = (3\pi/2, 0)$ são cercadas por várias órbitas periódicas marginalmente instáveis (do inglês *MUPOs*, ver [36], Fig. 5c), criando áreas de *stickiness* proeminentes. Devido à impossibilidade de reproduzir trajetórias fortemente afetadas por essas áreas em nosso experimento, observamos valores de λ mais elevados do que os obtidos numericamente. Mesmo utilizando análise de recorrência de tempo finito para atenuar parcialmente os efeitos de *stickiness* nas simulações, a discrepância permanece.

Capítulo 5

Conclusão e Perspectivas

Durante o desenvolvimento deste trabalho, foi necessário adquirir conhecimentos abrangentes em diversas áreas de estudo, que incluíram Robótica e Eletrônica, destacando o desenvolvimento do robô, soldagem de componentes e integração de hardware. Programação e Análise de Dados, com o uso de linguagens baseadas em C/C++, Python e MATLAB. Processamento de Imagens, aplicado na extração de dados experimentais diretamente de vídeos. Além de Prototipagem e Fabricação, com foco em modelagem e impressão 3D. Paralelamente, conceitos de Física Experimental desempenharam papel central, direcionando a análise da dinâmica de partículas em bilhares e o estudo de parâmetros geométricos. Essa experiência foi essencial para compreender processos de prototipagem rápida e integração de hardware e software em experimentos de sistemas complexos.

A construção de um sistema experimental versátil, envolvendo tecnologias como Arduino, modelagem, impressão 3D e processamento de imagens, revelou-se eficaz para explorar a dinâmica de partículas ativas em sistemas complexos. Este estudo demonstrou a viabilidade de integrar robótica e análise computacional para investigar de maneira controlada e precisa a transição entre diferentes regimes dinâmicos. Os resultados obtidos validaram os métodos experimentais e modelos teóricos, contribuindo para o entendimento de transições dinâmicas em sistemas caóticos e mistos. A abordagem empregada destacou-se pela personalização e adaptação, permitindo a análise de diferentes cenários geométricos e dinâmicos. Além disso, o sistema desenvolvido apresenta potencial de expansão, podendo futuramente ser adaptado para investigar os efeitos de perturbações externas e explorar novos cenários geométricos.

O desenvolvimento do sistema experimental abrangeu desde a concepção e montagem de um robô baseado em Arduino, com módulos como sensores de distância e servomotores, até a implementação de programação baseada C/C++ para controle do Arduino e Python para análise de dados. Uma câmera foi utilizada para capturar vídeos da dinâmica do robô, que serviram como fonte de dados para análise quantitativa. Utilizamos técnicas de processamento de imagens em MATLAB e Python para extrair informações relevantes, como trajetórias e velocidades em diferentes parâmetros geométricos. A modelagem

3D permitiu o desenho preciso dos componentes necessários para o experimento. Esses conhecimentos foram fundamentais para personalizar e integrar partes do robô, ajustar o ambiente experimental, como o contorno do bilhar, e garantir a qualidade no parâmetro γ_{eff} almejado. Foi realizada a soldagem de componentes eletrônicos para conectar os diferentes módulos do robô, assegurando a integridade das conexões elétricas e o funcionamento adequado do sistema, contribuindo para a confiabilidade do robô durante os experimentos.

Os experimentos foram realizados em um domínio com perímetro fixo de 5,01 m, variando-se o parâmetro geométrico γ_{eff} no intervalo de 0 a 0,5. Essa variação permitiu a transição controlada entre regimes dinâmicos regulares, mistos e completamente caóticos, possibilitando uma análise abrangente da dinâmica do sistema. Para determinar o expoente de Lyapunov (λ_{exp}), utilizamos o método baseado no espaço tangente [1], e comparamos os resultados experimentais com previsões numéricas. Os resultados demonstraram uma concordância satisfatória entre os dados experimentais e os modelos teóricos, como ilustrado na Fig. 4.7, validando tanto a abordagem experimental quanto a precisão dos modelos empregados. Medimos os espaços de fase da dinâmica do bilhar, revelando a coexistência de dinâmicas regulares e caóticas dependendo das condições iniciais (Figs. 4.5 e 4.6). Este resultado constitui uma verificação experimental de um espaço de fase misto em bilhares fora do escopo da óptica [44]. Para evidenciar explicitamente a sensibilidade às condições iniciais, plotamos as coordenadas do espaço de fase em função do tempo. Os resultados obtidos durante esta dissertação deram origem a um artigo científico entitulado *"Mixed-phase space of an active particle in experimental Lemon Billiards"*. Inicialmente submetido para a revista *Physical Review Letters*, ainda sob análise até o presente momento.

Utilizando ainda os mesmos dados obtidos, pretendemos fazer uma análise mais profunda e variada de quantidades de recorrência [66]. A citada nesta dissertação é apenas uma das quantidades disponíveis na literatura para esta análise. Nosso arranjo experimental reproduziu com sucesso um modelo altamente idealizado, programando as respostas do robô para aderirem às condições de validade do modelo. Esses modelos teóricos elucidam os mecanismos subjacentes a sistemas complexos. A interação robô-ambiente fornece uma plataforma excelente para estudos experimentais em sistemas físicos [67–73], incluindo aspectos adicionais de caos e sistemas dinâmicos, como partículas confinadas autopropelidas [74] e partículas ativas auto-excludentes [75]. Além disso, o estudo de enxames de robôs vem recebendo grande atenção da comunidade científica [76]. Isso se deve ao fato de os robôs poderem ser utilizados para imitar o comportamento não apenas de uma partícula de bilhar confinada, mas também o movimento de um cardume em um aquário [77–79]. Dessa forma, é possível estudar a dinâmica coletiva de determinados grupos de animais utilizando robôs [80].

Assim, este trabalho não apenas contribuiu para o avanço na compreensão de sistemas

dinâmicos complexos, mas também reforçou a relevância de plataformas experimentais baseadas na integração de robótica e computação, oferecendo ferramentas inovadoras para estudos avançados em física não linear e sistemas dinâmicos.

Apêndice A

Programação do Arduino

```
#include <Wire.h>      // Inclui a biblioteca Wire para comunicação I2C
#include <VL53L0X.h>  // Inclui a biblioteca para o sensor de distância VL53L0X
#include <Servo.h>    // Inclui a biblioteca para controlar os servos

// Definição de pinos e constantes
const int ServoDPin = 11;  // Pino do servo motor direito
const int ServoEPin = 10;  // Pino do servo motor esquerdo
#define ENCODER_PIN_E 3    // Pino do encoder do motor esquerdo
#define ENCODER_PIN_D 2    // Pino do encoder do motor direito
Servo ServoD;              // Cria um objeto Servo para o motor direito
Servo ServoE;              // Cria um objeto Servo para o motor esquerdo

// Definição de pinos para múltiplos sensores VL53L0X
#define SHUT_A 6
#define SHUT_B 7
#define SHUT_C 8
#define SensorA_endereco 42  // Endereço I2C do sensor A
#define SensorB_endereco 43  // Endereço I2C do sensor B
#define SensorC_endereco 44  // Endereço I2C do sensor C

// Cria objetos para os sensores VL53L0X
VL53L0X sensorA;  // Sensor A (direito)
VL53L0X sensorB;  // Sensor B (esquerdo)
VL53L0X sensorC;  // Sensor C (frente)

// Definições de limites de distância
const int distanciaMin = 0;
const int distanciaMax = 5000;
bool detectado = false;

// Definições para ajustes de deslocamento dos sensores
#define SA 45//40//50  //-13 Ajuste Direito
```

```
33 #define SB 38//39  //-27 Ajuste Esquerdo
34 #define SC 40//43  // -00 Ajuste Frente
35
36 // Constantes de tempo para operação do robô
37 #define ESP 25  // Tempo de espera (em milissegundos)
38 #define LER 2   // Tempo de leitura (em milissegundos)
39
40 // Distâncias para controle de sensores e movimento
41 #define MAX 500  // Distância máxima de detecção
42 #define Lim 150  // Distância mínima para desvio
43
44 // Parâmetros de rotação e parada
45 #define ROT 1800  // Tempo para rotação completa
46 #define ROT0 50   // Tempo mínimo de rotação
47
48 #define STOP_E 1428  // Valor para parada dos motores
49 #define STOP_D 1435  // Valor para parada dos motores
50
51 // Valores para movimentação dos servos
52 #define EF 1513  // Valor para servo esquerdo frente
53 #define DF 1371  // Valor para servo direito frente
54
55 #define ET 1355  // Valor para servo esquerdo trás
56 #define DT 1516  // Valor para servo direito trás
57
58 // Constantes matemáticas e físicas
59 float Pi = 3.14159265359;  // Valor de Pi
60
61 //----------------ENCODER-------------//
62 int lastStateE = HIGH;  // Estado anterior do encoder esquerdo
63 int lastStateD = HIGH;  // Estado anterior do encoder direito
64 int EFF = 1513;         // Frequência efetiva para o servo esquerdo
65 int DFF = 1371;         // Frequência efetiva para o servo direito
66
67 // Valores para cálculo da rotação
68 int r = 16;     // Raio da roda (em cm)
69 int RTT = 100;  // Tempo para retorno (em milissegundos)
70
71 // Definições de pinos para sensores adicionais
72 #define vcc 5
73 #define gnd 4
74 #define pino_D0 2
75 #define pino_D1 3
76
77 // Variáveis para contagem e cálculo de distância percorrida
78 unsigned long countE;    // Contador de pulsos do encoder esquerdo
79 unsigned long countD;    // Contador de pulsos do encoder direito
```

```
80 float Time;                // Variável para tempo
81 float X2 = Time;           // Variável auxiliar para tempo
82 float Rr = 12.7;           // Raio da roda (em cm)
83 float Rp = 6.8;            // Raio do pino (em cm)
84 float FurosTotal = 126;    // Total de furos para 180 graus de rotação
85 int Giro = 0;              // Contador de giros
86 int A = 1;
87 // Variável auxiliar
88
89 void setup() {
90   // Configuração inicial dos pinos e componentes
91
92   pinMode(13, OUTPUT);      // Configura o pino 13 como saída
93   digitalWrite(13, HIGH);   // Ativa o pino 13 (pode ser usado para
       sinalização)
94
95   // Inicialização dos servos
96   ServoD.attach(ServoDPin);  // Conecta o servo no pino definido para o
       servo direito
97   ServoE.attach(ServoEPin);  // Conecta o servo no pino definido para o
       servo esquerdo
98
99   // Configuração dos pinos SHUT para os sensores VL53L0X
100  pinMode(SHUT_A, OUTPUT);  // Configura o pino SHUT_A como saída
101  pinMode(SHUT_B, OUTPUT);  // Configura o pino SHUT_B como saída
102  pinMode(SHUT_C, OUTPUT);  // Configura o pino SHUT_C como saída
103
104  // Comunicação Serial
105  Serial.begin(115200);  // Inicia a comunicação serial a 115200 bps
106  Wire.begin();          // Inicia a comunicação I2C
107
108  // Configuração e inicialização dos sensores de distância
109  pinMode(SHUT_A, INPUT);                // Muda o pino SHUT_A para
       entrada
110  delay(10);                             // Pequeno atraso para
       estabilização
111  sensorA.setAddress(SensorA_endereco);  // Define endereço I2C para o
       sensor A
112
113  pinMode(SHUT_B, INPUT);                // Muda o pino SHUT_B para
       entrada
114  delay(10);                             // Pequeno atraso para
       estabilização
115  sensorB.setAddress(SensorB_endereco);  // Define endereço I2C para o
       sensor B
116
```

```
117   pinMode(SHUT_C, INPUT);                  // Muda o pino SHUT_C para
          entrada
118   delay(10);                               // Pequeno atraso para
          estabilização
119   sensorC.setAddress(SensorC_endereco);  // Define endereço I2C para o
          sensor C
120
121   // Inicializa e configura os sensores VL53L0X
122   sensorA.init();  // Inicializa o sensor A
123   sensorB.init();  // Inicializa o sensor B
124   sensorC.init();  // Inicializa o sensor C
125
126   sensorA.setTimeout(500);  // Define timeout para o sensor A
127   sensorB.setTimeout(500);  // Define timeout para o sensor B
128   sensorC.setTimeout(500);  // Define timeout para o sensor C
129
130   sensorA.startContinuous();  // Inicia a medição contínua para o sensor
          A
131   sensorB.startContinuous();  // Inicia a medição contínua para o sensor
          B
132   sensorC.startContinuous();  // Inicia a medição contínua para o sensor
          C
133
134   // Configuração dos Encoders
135   pinMode(vcc, OUTPUT);     // Configura o pino VCC do encoder como
          saída
136   pinMode(gnd, OUTPUT);     // Configura o pino GND do encoder como
          saída
137   pinMode(pino_D0, INPUT);  // Configura o pino D0 do encoder como
          entrada
138   pinMode(pino_D1, INPUT);  // Configura o pino D1 do encoder como
          entrada
139
140   digitalWrite(vcc, HIGH);  // Ativa o VCC do encoder
141   digitalWrite(gnd, LOW);   // Desativa o GND do encoder
142
143   // Configuração das interrupções para os encoders
144   attachInterrupt(1, contadorE, CHANGE);  // Interrupção para o encoder
          esquerdo
145   attachInterrupt(0, contadorD, CHANGE);  // Interrupção para o encoder
          direito
146
147   delay(1000);  // Atraso final para estabilização do setup
148 }
149   void loop() {
150     //LerSensores();
151     Robo();
```

```
152     //Motor();
153   }
154   void LerSensores() {
155     // Função para ler os valores dos sensores de distância VL53L0X
156
157     // Parando os servos para estabilização durante a leitura dos
          sensores
158     ServoD.writeMicroseconds(STOP_D);  // Parar o servo direito
159     ServoE.writeMicroseconds(STOP_E);  // Parar o servo esquerdo
160     delay(ESP);                        // Pequena pausa para garantir
          que os servos estejam parados
161
162     // Declaração de variáveis para armazenar as distâncias lidas
163     float DistA = 0;            // Distância lida pelo sensor A
164     float DistB = 0;            // Distância lida pelo sensor B
165     float DistC = 0;            // Distância lida pelo sensor C
166     const int numReadings = 1;  // Número de leituras para a média
167     // Leitura dos sensores múltiplas vezes para obter uma média mais
          estável
168     for (int i = 0; i < numReadings; i++) {
169       // Lê e ajusta a distância de cada sensor
170       DistA += sensorA.readRangeContinuousMillimeters() + SA;
171       DistB += sensorB.readRangeContinuousMillimeters() + SB;
172       DistC += sensorC.readRangeContinuousMillimeters() + SC;
173       delay(LER);  // Pequeno atraso entre cada leitura para
            estabilização
174     }
175
176     // Calcula a média das distâncias lidas
177     DistA /= numReadings;
178     DistB /= numReadings;
179     DistC /= numReadings;
180
181     // Imprime as distâncias no monitor serial
182     Serial.print(DistA);
183     Serial.print(" D-F ");  // Indica Distância Direita-Frente
184     Serial.print(DistC);
185     Serial.print(" F-E ");  // Indica Distância Frente-Esquerda
186     Serial.println(DistB);  // Indica Distância Esquerda
187   }
188   void Robo() {
189   int DistA = sensorA.readRangeContinuousMillimeters() + SA;
190   int DistB = sensorB.readRangeContinuousMillimeters() + SB;
191   int DistC = sensorC.readRangeContinuousMillimeters() + SC;
192
193   if (DistA > Lim and DistB > Lim and DistC > (Lim)) {
194     RT();
```

```
195   }
196   else {
197     //Identificando Possivel Obstaculo
198     ServoD.writeMicroseconds(STOP_D); // move o servo 1 para a frente
199     ServoE.writeMicroseconds(STOP_E); // move o servo 2 para a frente
200     delay(ESP);
201
202     float DistA = 0;
203     float DistB = 0;
204     float DistC = 0;
205     const int numReadings = 1; // Número de leituras a serem feitas
206
207     for (int i = 0; i < numReadings; i++) {
208       DistA += sensorA.readRangeContinuousMillimeters() + SA;
209       DistB += sensorB.readRangeContinuousMillimeters() + SB;
210       DistC += sensorC.readRangeContinuousMillimeters() + SC;
211       delay(LER);
212     }
213
214     DistA /= numReadings;
215     DistB /= numReadings;
216     DistC /= numReadings;
217
218     if (DistC < DistB and DistC < DistA) {
219       ServoD.writeMicroseconds(DF); // move o servo 1 para a frente
220       ServoE.writeMicroseconds(ET); // move o servo 2 para a frente
221       //Calculo do desvio
222       float A = DistA;
223       float B = DistB;
224       float C = sqrt(pow(A, 2) + pow(B, 2));
225       float D = A * B / C;
226       float N = pow(A, 2) / C;
227       float M = pow(B, 2) / C;
228       float H = acos((pow(A, 2) + pow(D, 2) - pow(N, 2)) / (2 * A * D));
229       float J = acos((pow(B, 2) + pow(D, 2) - pow(M, 2)) / (2 * B * D));
230       float Alfa = H - Pi / 4;
231       float Beta = J - Pi / 4;
232       float V_Alfa = Pi - 2 * Alfa;
233       float V_Beta = Pi - 2 * Beta;
234
235       if (DistA <= DistB) {
236         float Time = (V_Beta) / (Pi);
237         ServoD.writeMicroseconds(DF); // move o servo 1 para a frente
238         ServoE.writeMicroseconds(ET); // move o servo 2 para a frente
239         Serial.println(Time);
240         X2 = Time;
241         RTE();
```

```
242       }
243       else {
244         float Time = (V_Alfa) / (Pi);
245         ServoD.writeMicroseconds(DT); // move o servo 1 para a frente
246         ServoE.writeMicroseconds(EF); // move o servo 2 para a frente
247         Serial.println(Time);
248         X2 = Time;
249         RTD();
250
251       }
252
253     }
254     else { //Preparação para o desvio
255       if (DistA < DistB) {
256         ServoD.writeMicroseconds(DF); // move o servo 1 para a frente
257         ServoE.writeMicroseconds(ET); // move o servo 2 para a frente
258         //Calculo do desvio
259         float A = DistA;
260         float B = DistB;
261         float C = DistC;
262         float D = sqrt(pow(A, 2) + pow(C, 2) - A * C * sqrt(2));
263         float Alfa = acos((-pow(A, 2) + pow(D, 2) + pow(C, 2)) / (2 * C
              * D));
264         float Time = (2 * Alfa) / (Pi);
265         ServoD.writeMicroseconds(DF); // move o servo 1 para a frente
266         ServoE.writeMicroseconds(ET); // move o servo 2 para a frente
267         Serial.println(Time);
268         X2 = Time;
269         RTE();
270       }
271       if (DistB < DistA ) {
272         ServoD.writeMicroseconds(DT); // move o servo 1 para a frente
273         ServoE.writeMicroseconds(EF); // move o servo 2 para a frente
274         //Calculo do desvio
275         float A = DistA;
276         float B = DistB;
277         float C = DistC;
278         float E = sqrt(pow(B, 2) + pow(C, 2) - B * C * sqrt(2));
279         float Beta = acos((-pow(B, 2) + pow(E, 2) + pow(C, 2)) / (2 * C
              * E));
280         float Time = (Beta * 2) / (Pi);
281         ServoD.writeMicroseconds(DT); // move o servo 1 para a frente
282         ServoE.writeMicroseconds(EF); // move o servo 2 para a frente
283
284         //Tempo para o desvio
285         Serial.println(Time);
286         X2 = Time;
```

```
287           RTD();
288
289         }
290       }
291     }
292     // Serial.print(X2);
293 }
294 void AC() {
295     int I = 1500;
296     ServoD.writeMicroseconds(I);
297     ServoE.writeMicroseconds(I);
298     delay(100);
299 }
300 void RT() {
301     Serial.println("Rodando");
302     attachInterrupt(1, contadorE, CHANGE);
303     attachInterrupt(0, contadorD, CHANGE);
304     ServoD.writeMicroseconds(DFF);  // move o servo 1 para a frente
305     ServoE.writeMicroseconds(EFF);  // move o servo 2 para a frente
306     delay(RTT);                     // espera 1/20 segundo
307
308
309     //Desabilita interrupcao durante o calculo
310     detachInterrupt(0);
311     detachInterrupt(1);
312     Serial.print("CountE = ");
313     Serial.println(countE);
314     Serial.print("CountD = ");
315     Serial.println(countD);
316     if (countE == 0 and countD == 0) {
317       ServoD.writeMicroseconds(STOP_D);  // move o servo 1 para a frente
318       ServoE.writeMicroseconds(STOP_E);  // move o servo 2 para a frente
319       delay(5000);
320       //AC();
321     } else
322     //Mostra o valor de RPM no serial monitor
323     {
324       Serial.print("RPME = ");
325       Serial.println(countE);
326       Serial.print("RPMD = ");
327       Serial.println(countD);
328       EFF = countE < r ? EFF + 1 : EFF - 1;
329       DFF = countD < r ? DFF - 1 : DFF + 1;
330     }
331     //Habilita interrupcao
332     countE = 0;
333     countD = 0;
```

```
334
335   Serial.print("EFF = ");
336   Serial.println(EFF);
337   Serial.print("DFF = ");
338   Serial.println(DFF);
339   Serial.println("Fim Rodando");
340 }
341 void contadorE()
342 {
343   int currentStateE = digitalRead(3);
344   countE++;
345 }
346 void contadorD()
347 {
348   int currentStateD = digitalRead(2);
349   countD++;
350 }
351
352
353 void RTD() {
354   Serial.println("Rodando Para Direita");
355   Giro = (A == 0) ? (X2 * 0.5 + 0.5) * FurosTotal : X2 * FurosTotal;
356
357   Serial.print("Giro: ");
358   Serial.println(Giro);
359
360   attachInterrupt(1, contadorE, CHANGE);
361   attachInterrupt(0, contadorD, CHANGE);
362
363   while (countE < Giro && countD < Giro) {
364     ServoD.writeMicroseconds(DT);
365     ServoE.writeMicroseconds(EF);
366     // Removido delay para resposta mais rápida
367   }
368   delay(5);
369   detachInterrupt(1);
370   detachInterrupt(0);
371
372   countD = 0;
373   countE = 0;
374   A = 1;
375
376   Serial.println("Finalizando Rodando Para Direita");
377 }
378
379 void RTE() {
380   Serial.println("Rodando Para Esquerda");
```

```
381   Giro = (A == 0) ? (X2 * 0.5 + 0.5) * FurosTotal : X2 * FurosTotal;
382
383   Serial.print("Giro: ");
384   Serial.println(Giro);
385
386   attachInterrupt(1, contadorE, CHANGE);
387   attachInterrupt(0, contadorD, CHANGE);
388
389   while (countE < Giro && countD < Giro) {
390     ServoE.writeMicroseconds(ET);
391     ServoD.writeMicroseconds(DF);
392     // Removido delay para resposta mais rápida
393   }
394   delay(5);
395   detachInterrupt(1);
396   detachInterrupt(0);
397
398   countD = 0;
399   countE = 0;
400   A = 1;
401
402   Serial.println("Finalizando Rodando Para Esquerda");
403 }
404 void Motor() {
405
406   RT();
407 }
```

Bibliografia

[1] João VA Vasconcelos et al. "Interaction between a robot and Bunimovich stadium billiards". Em: *Scientific Reports* 12.1 (2022), p. 4983.

[2] Cunyuan Jiang et al. "Dispersionless Flat Mode and Vibrational Anomaly in Active Brownian Vibrators Induced by Stringlike Dynamical Defects". Em: *Phys. Rev. Lett.* 133 (2024), p. 188302.

[3] Angelo Barona Balda et al. "Playing with active matter". Em: *Am. J. Phys.* 92.11 (2024), pp. 847–858.

[4] Isaac Newton. *Keynes MSS 130.6, Book 3; 130.5, Sheet 3*. Manuscript in the Keynes Collection at the Library of King's College, Cambridge. Cambridge, UK, 1983.

[5] Richard S Westfall. *Never at rest: A biography of Isaac Newton*. Cambridge, UK: Cambridge University Press, 1983, p. 544.

[6] Jean Le Rond d'Alembert. *Opuscules mathématiques ou Mémoires sur différens sujets de géométrie, de méchanique, etc.* Vol. 8. 1780.

[7] Henri Poincaré. *Les méthodes nouvelles de la mécanique céleste*. Vol. 3. Gauthier-Villars et fils, 1899.

[8] Joseph Louis Lagrange. *Mécanique analytique*. Vol. 1. Mallet-Bachelier, 1853.

[9] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[10] Arkady Pikovsky e Antonio Politi. *Lyapunov exponents: a tool to explore complex dynamics*. Cambridge University Press, 2016.

[11] Giancarlo Benettin, Luigi Galgani e Jean-Marie Strelcyn. "Kolmogorov entropy and numerical experiments". Em: *Physical Review A* 14.6 (1976), p. 2338.

[12] Troy Shinbrot et al. "Chaos in a double pendulum". Em: *American Journal of Physics* 60.6 (1992), pp. 491–499.

[13] Edward N Lorenz. "Deterministic nonperiodic flow". Em: *Journal of atmospheric sciences* 20.2 (1963), pp. 130–141.

[14] Edson Denis Leonel. *Invariância de escala em sistemas dinâmicos não lineares*. Editora Blucher, 2019.

[15] Narendra K Pareek, Vinod Patidar e Krishan K Sud. “Image encryption using chaotic logistic map”. Em: *Image and vision computing* 24.9 (2006), pp. 926–934.

[16] Thomas D Rogers e David C Whitley. “Chaos in the cubic mapping”. Em: *Mathematical Modelling* 4.1 (1983), pp. 9–25.

[17] Michael Benedicks e Lennart Carleson. “The dynamics of the Hénon map”. Em: *Annals of Mathematics* 133.1 (1991), pp. 73–169.

[18] Edward Ott. *Chaos in dynamical systems*. Cambridge university press, 2002.

[19] Matheus S Palmero, Iberê L Caldas e Igor M Sokolov. “Finite-time recurrence analysis of chaotic trajectories in Hamiltonian systems”. Em: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 32.11 (2022).

[20] Boris V Chirikov. “A universal instability of many-dimensional oscillator systems”. Em: *Physics reports* 52.5 (1979), pp. 263–379.

[21] Suhan Ree e Linda E Reichl. “Classical and quantum chaos in a circular billiard with a straight cut”. Em: *Physical Review E* 60.2 (1999), p. 1607.

[22] Tiago Araújo Lima e Flávio M de Aguiar. “Classical billiards and quantum fluids”. Em: *Physical Review E* 91.1 (2015), p. 012923.

[23] Tiago Araújo Lima e Ricardo B do Carmo. “Classical and quantum elliptical billiards: Mixed phase space and short-range correlations in singlets and doublets”. Em: *Physica D: Nonlinear Phenomena* 458 (2024), p. 134018.

[24] Sylvio Ferraz-Mello. *Caos e planetas: dinâmica caótica de sistemas planetários*. LF Editorial, 2021.

[25] George D Birkhoff. “On the Periodic Motions of Dynamical Systems.” Em: *Hamiltonian Dynamical Systems*. CRC Press, 2020, pp. 154–174.

[26] Alex H Barnett e Timo Betcke. “Quantum mushroom billiards”. Em: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 17.4 (2007).

[27] Eric J Heller e Steven Tomsovic. “Postmodern quantum mechanics.” Em: *Physics Today* 46.7 (1993), pp. 38–46.

[28] Michael Drexler e Martin J Gander. “Circular billiard”. Em: *SIAM review* 40.2 (1998), pp. 315–323.

[29] Yakov G Sinai. “Dynamical systems with elastic reflections”. Em: *Russian Mathematical Surveys* 25.2 (1970), p. 137.

[30] Jacques Hadamard. “Les surfaces à courbures opposées et leurs lignes géodésiques”. Em: (1988).

[31] Dynamical Billiard. “Yakov G. Sinai, Abel Prize Laureate 2014”. Em: *Positions* 2 (2014), p. 2.

[32] Leonid A Bunimovich. “On the ergodic properties of nowhere dispersing billiards”. Em: *Communications in Mathematical Physics* 65 (1979), pp. 295–312.

[33] Eduardo Canale et al. “A lower bound for chaos on the elliptical stadium”. Em: *Physica D: Nonlinear Phenomena* 115.3-4 (mai. de 1998), pp. 189–202.

[34] André LP Livorati, Alexander Loskutov e Edson D Leonel. “A family of stadium-like billiards with parabolic boundaries under scaling analysis”. Em: *Journal of Physics A: Mathematical and Theoretical* 44.17 (2011), p. 175102.

[35] Nikolai Chernov e Roberto Markarian. *Chaotic billiards*. 127. American Mathematical Soc., 2006.

[36] H Makino, T Harayama e Y Aizawa. “Quantum-classical correspondences of the Berry-Robnik parameter through bifurcations in lemon billiard systems”. Em: *Physical Review E* 63.5 (2001), p. 056203.

[37] Jingyu Chen et al. “Ergodicity of the generalized lemon billiards”. Em: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 23.4 (2013).

[38] Črt Lozej. “Stickiness in generic low-dimensional Hamiltonian systems: A recurrence-time statistics approach”. Em: *Physical Review E* 101.5 (2020), p. 052204.

[39] Črt Lozej, Dragan Lukman e Marko Robnik. “Effects of stickiness in the classical and quantum ergodic lemon billiard”. Em: *Physical Review E* 103.1 (2021), p. 012204.

[40] Črt Lozej, Dragan Lukman e Marko Robnik. “Phenomenology of quantum eigenstates in mixed-type systems: Lemon billiards with complex phase space structure”. Em: *Physical Review E* 106.5 (2022), p. 054203.

[41] Matheus S Palmero et al. “Ensemble separation and stickiness influence in a driven stadium-like billiard: A Lyapunov exponents analysis”. Em: *Communications in Nonlinear Science and Numerical Simulation* 65 (2018), pp. 248–259.

[42] Matheus S Palmero, Iberê L Caldas e Igor M Sokolov. “Finite-time recurrence analysis of chaotic trajectories in Hamiltonian systems”. Em: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 32.11 (2022).

[43] George M Zaslavsky. *The Physics of Chaos in Hamiltonian Systems*. World Scientific, 2007.

[44] Linran Fan et al. “Real-time observation and control of optical chaos”. Em: *Science advances* 7.3 (2021), eabc8448.

[45] Marc Sciamanna e K Alan Shore. “Physics and applications of laser diode chaos”. Em: *Nature photonics* 9.3 (2015), pp. 151–162.

[46] Xuefeng Jiang et al. “Chaos-assisted broadband momentum transformation in optical microresonators”. Em: *Science* 358.6361 (2017), pp. 344–347.

[47] Faraz Monifi et al. “Optomechanically induced stochastic resonance and chaos transfer between optical fields”. Em: *Nature Photonics* 10.6 (2016), pp. 399–405.

[48] Massimo Banzi e Michael Shiloh. *Getting started with Arduino*. Maker Media, Inc., 2022.

[49] Arduino. *Arduino*. Accessed: 2024-10-27. 2024. URL: https://www.arduino.cc/.

[50] Arduino. *Nano | Arduino Documentation*. Accessed: 2024-10-27. 2024. URL: https://docs.arduino.cc/hardware/nano/.

[51] Kalatec Blog. *O que é Servo Motor?* Acessado: 22 jan. 2025. URL: https://blog.kalatec.com.br/o-que-e-servo-motor/.

[52] Mantech Electronics. *FS5103R Datasheet*. Acessado em: 29 jan. 2025. 2024. URL: https://www.mantech.co.za/datasheets/products/FS5103R-20240810A.pdf.

[53] STMicroelectronics. *VL53L0X Time-of-Flight Ranging Sensor*. Datasheet, DS11555 - Rev 6. STMicroelectronics. Jun. de 2024. URL: https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.

[54] CB Electronics. *LM2596S - Voltage Regulator Product Page*. Acesso: 26 jan. 2025. 2025. URL: https://cb-electronics.com/products/lm2596s/.

[55] Texas Instruments Incorporated. *LM2596 SIMPLE SWITCHERő Power Converter 150-kHz 3-A Step-Down Voltage Regulator*. Rev. G. Acessado: 27 de outubro de 2024. Mar. de 2023. URL: https://www.ti.com/product/LM2596.

[56] Hitecnologia. *O que é Encoder, para que serve, como escolher e como interfacear*. Acesso: 27 nov. 2024. 2024. URL: https://materiais.hitecnologia.com.br/blog/o-que-%C3%A9-encoder-para-que-serve-como-escolher-como-interfacear/#:~:text=Encoder%20s%C3%A3o%20dispositivos%2Fsensores%20eletro,rotacionar%20bra%C3%A7os%20rob%C3%B3ticos%20e%20etc..

[57] Fermarc. *Sensor de Velocidade Encoder LM393 - Página do Produto*. Acesso: 26 jan. 2025. 2025. URL: https://www.fermarc.com/sensor-de-velocidade-encoder-lm393.

[58] Renishaw. *Introduction to Encoder Systems*. Available at: https://www.renishaw.com/en/introduction-to-encoder-systems-47256. Accessed: 2024-12-15.

[59] Robert L Boylestad e Louis Nashelsky. *Dispositivos eletrônicos e teoria de circuitos*. Vol. 6. Prentice-Hall do Brasil, 1984.

[60] Rajguru Electronics. *ADIY LM393 Comparator - Datasheet*. Acesso: 27 nov. 2024. 2024. URL: https://www.rajguruelectronics.com/Product/8748/ADIY%20LM393%20Comparator_Datasheet.pdf.

[61] *Datasheet: 013-2600HC*. Acessado em: 22 jan. 2025. URL: https://santanaimport.com.br/central-de-midias/013-2600HC/013-2600_datasheet.pdf.

[62] *About FreeCAD - FreeCAD Documentation — wiki.freecad.org*. `https://wiki.freecad.org/About_FreeCAD`. [Accessed 11-11-2024].

[63] GoPro. *GoPro HERO 9 Black Manual do Usuário*. Acesso: 5 dez. 2024. 2024. URL: `https://gopro.com/en/us/support/`.

[64] The MathWorks, Inc. *MATLAB — The Language of Technical Computing*. Accessed: 2024-06-17. MathWorks. Natick, Massachusetts, United States, 2024. URL: `https://www.mathworks.com/products/matlab.html`.

[65] Rodrigo Simile Baroni et al. "Time recurrence analysis of a near singular billiard". Em: *Mathematical and Computational Applications* 24.2 (2019), p. 50.

[66] L Lober, MS Palmero e FA Rodrigues. "Predictive Non-linear Dynamics via Neural Networks and Recurrence Plots". Em: *arXiv preprint arXiv:2410.23408* (2024).

[67] Frank Schweitzer, Werner Ebeling e Benno Tilch. "Complex motion of Brownian particles with energy depots". Em: *Physical Review Letters* 80.23 (1998), p. 5044.

[68] Olivier Dauchot e Vincent Démery. "Dynamics of a Self-Propelled Particle in a Harmonic Trap". Em: *Phys. Rev. Lett.* 122 (6 fev. de 2019), p. 068002.

[69] Marco Leoni et al. "Surfing and crawling macroscopic active particles under strong confinement: Inertial dynamics". Em: *Phys. Rev. Res.* 2 (4 dez. de 2020), p. 043299.

[70] Shengkai Li et al. "Robotic swimming in curved space via geometric phase". Em: *Proceedings of the National Academy of Sciences* 119.31 (2022), e2200924119.

[71] Shengkai Li et al. "A robophysical model of spacetime dynamics". Em: *Scientific Reports* 13.1 (2023), p. 21589.

[72] Daniel I Goldman e D Zeb Rocklin. "Robot swarms meet soft matter physics". Em: *Science Robotics* (2024).

[73] Somnath Paramanick et al. "Programming tunable active dynamics in a self-propelled robot". Em: *The European Physical Journal E* 47.5 (2024), p. 34.

[74] Rubens H Damascena, Leonardo RE Cabral e Clécio C de Souza Silva. "Coexisting orbits and chaotic dynamics of a confined self-propelled particle". Em: *Physical Review E* 105.6 (2022), p. 064608.

[75] Thijs Albers et al. "Billiards with Spatial Memory". Em: *Physical Review Letters* 132.15 (2024), p. 157101.

[76] Lorenzo Garattoni e Mauro Birattari. "Autonomous task sequencing in a robot swarm". Em: *Science Robotics* 3.20 (2018), eaat0430.

[77] Fransisco CB Leal et al. "Avalanche dynamics of zebrafish schools: Unveiling self-organization and phase transitions". Em: *Physica A: Statistical Mechanics and its Applications* 651 (2024), p. 130040.

[78] Antonio R de C Romaguera et al. “Multifractal fluctuations in zebrafish (Danio rerio) polarization time series”. Em: *The European Physical Journal E* 47.5 (2024), p. 29.

[79] Adauto JF de Souza et al. “Speckle statistics as a tool to distinguish collective behaviors of Zebrafish shoals”. Em: *Scientific Reports* 14.1 (2024), p. 15835.

[80] Liu Lei et al. “Computational and robotic modeling reveal parsimonious combinations of interactions between individuals in schooling fish”. Em: *PLoS computational biology* 16.3 (2020), e1007194.