

Universidade Federal de Minas Gerais  
Escola de Engenharia  
Curso de Graduação em Engenharia de Controle e Automação

**Montagem, Modelagem e Simulação de um Sistema  
*Ball and Plate* com Manipulador Paralelo de Três  
Graus de Liberdade**

Leonardo Torreão Ferreira

Orientador: Prof. Luciano Antonio Frezzatto Santos, Dr.

Belo Horizonte, Dezembro de 2022



## **Monografia**

### **Montagem, Modelagem e Simulação de um Sistema *Ball and Plate* com Manipulador Paralelo de Três Graus de Liberdade**

Monografia submetida à banca examinadora designada pelo Colegiado Didático do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Minas Gerais, como parte dos requisitos para aprovação na atividade Projeto Final de Curso II.

Belo Horizonte, Dezembro de 2022

# Resumo

Um sistema *ball and plate* tem como objetivo equilibrar uma bolinha sobre uma plataforma lisa, utilizando-se de atuadores rotacionais ou prismáticos para causar a inclinação necessária para manter a esfera parada. Tal tipo de planta, além de ser útil para o estudo de controle de sistemas não lineares, pode ser aplicada para simular a marcha humana. Por esses motivos, este projeto tem como principal objetivo preparar um sistema que sirva como ferramenta de aprendizado para outros alunos, descrever em seu texto uma montagem de baixo custo e portátil, utilizar um sistema de visão computacional para realizar a localização da posição da bolinha e identificar o modelo do sistema. Para realização da ação de controle, foram utilizados dois microcontroladores: um Arduíno Mega e um *Raspberry Pi*, de forma que o primeiro realiza o acionamento dos servomotores enquanto o último executa o algoritmo de visão computacional, realiza o cálculo da ação de controle e envia os resultados para o Arduíno. É esperado que, ao fim do projeto, seja possível replicar a planta, transportá-la e ter todas as ferramentas necessárias para avaliar os melhores métodos de controle para o seu correto funcionamento.

**Palavras-chave:** *Ball and Plate*, Visão Computacional, Identificação Caixa Branca.



# Abstract

A ball and plate system aims to balance a ball over a flat surface, using rotational or prismatic actuators to incline the plate and prevent any movement. This kind of system, in addition to being useful to study non-linear systems control, can be used to simulate human gait. For that reason, the main objective here is to build a system that can be used as a control theory teaching device, describe a portable low cost equipment, develop a computer vision tool to track the ball position and identificate the system model. It's expected that, by the project's completing, the system can be rebuilt, transportable and it must have all the tools needed to evaluate the optimal control method for that specific ball and plate are available.

**Keywords:** Ball and Plate, Computer Vision, White-Box Identification.



# Agradecimentos

Primeiramente, agradeço à Helena, Joel e Lara, que sempre estiveram comigo.

Ao professor Luciano, pelo constante acompanhamento e compartilhamento de conhecimento.

À Samara, que tanto ajudou na montagem do manipulador paralelo.

Ao meu amigo Lucas, pelo suporte intelectual.

Agradeço também aos meus amigos Romero, Rafael, Ariel, Vitor e Camila pela motivação e conversas.

Por último, mas não menos importante, aos amigos que me acompanharam durante a graduação: Gabriel, Caio, Felipe e João.



# Sumário

|   |            |
|---|------------|
| <b>Resumo</b>   | <b>i</b>   |
| <b>Abstract</b>   | <b>iii</b> |
| <b>Agradecimentos</b>   | <b>v</b>   |
| <b>Lista de Figuras</b>   | <b>ix</b>  |
| <b>Lista de Tabelas</b>   | <b>xi</b>  |
| <b>1 Introdução</b>   | <b>1</b>   |
| 1.1 Motivação e Justificativa . . . . .   | 1          |
| 1.2 Objetivos do Projeto . . . . .  | 2          |
| 1.3 Local de Realização . . . . .   | 2          |
| 1.4 Estrutura da Monografia . . . . .   | 2          |
| <b>2 Descrição do Processo</b>  | <b>5</b>   |
| 2.1 Sistema <i>Ball and Plate</i> . . . . .   | 5          |
| 2.2 Revisão da Literatura . . . . .   | 6          |
| 2.3 Design e Montagem da planta . . . . .   | 8          |
| 2.3.1 Plataforma Paralela de 3 Graus de Liberdade . . . . .                                     | 8          |
| 2.3.2 Base . . . . .  | 8          |
| 2.3.3 Plataforma Móvel . . . . .  | 8          |
| 2.3.4 Sistema de Visão Computacional . . . . .  | 9          |
| 2.3.5 Microcontrolador Raspberry Pi . . . . .   | 9          |
| 2.4 Identificação e Análise Cinemática da Planta . . . . .                                      | 9          |
| 2.4.1 Modelo matemático do sistema <i>ball and plate</i> . . . . .                              | 9          |
| 2.4.2 Estudo da cinemática inversa do manipulador paralelo de três graus de liberdade . . . . . | 13         |
| 2.5 Resumo do Capítulo . . . . .  | 18         |

|  |           |
|--|-----------|
| <b>3 Descrição da Visão Computacional</b>                      | <b>21</b> |
| 3.1 Sistema de Visão Computacional . . . . .                   | 21        |
| 3.2 Análise do Problema . . . . .                              | 22        |
| 3.3 Implementação do Algoritmo . . . . .                       | 22        |
| 3.4 Resumo do Capítulo . . . . .                               | 24        |
| <b>4 Simulação e Controle</b>                                  | <b>25</b> |
| 4.1 Malha de controle . . . . .                                | 25        |
| 4.2 Projeto de controlador . . . . .                           | 26        |
| 4.3 Implementação da cinemática inversa . . . . .              | 26        |
| 4.4 Processo . . . . .   | 28        |
| 4.5 Resultados . . . . .                                       | 30        |
| 4.6 Resumo do Capítulo . . . . .                               | 31        |
| <b>5 Resultados</b>  | <b>33</b> |
| 5.1 Atividades do Projeto . . . . .                            | 33        |
| 5.2 Requisitos do Sistema . . . . .                            | 34        |
| 5.3 Testes . . . . .   | 34        |
| 5.4 Análise dos Resultados . . . . .                           | 35        |
| 5.4.1 Montagem da planta . . . . .                             | 35        |
| 5.4.2 Sistema de visão computacional . . . . .                 | 35        |
| 5.4.3 Obtenção do modelo e controlabilidade . . . . .          | 36        |
| 5.4.4 Aplicação do controlador na planta física . . . . .      | 37        |
| 5.5 Resumo do Capítulo . . . . .                               | 38        |
| <b>6 Conclusões</b>  | <b>39</b> |
| 6.1 Considerações Finais . . . . .                             | 39        |
| 6.2 Propostas de Continuidade . . . . .                        | 40        |
| 6.2.1 Conclusão . . . . .                                      | 41        |
| <b>Referências Bibliográficas</b>                              | <b>42</b> |
| <b>A Manual de Uso</b>   | <b>45</b> |
| A.1 Montando a Planta . . . . .                                | 45        |
| A.1.1 Conectando o <i>Raspberry Pi</i> e o Arduíno . . . . .   | 45        |
| A.1.2 Posicionando a câmera . . . . .                          | 45        |
| A.2 Detalhamento do Algoritmo de Visão Computacional . . . . . | 47        |
| A.3 Comunicação <i>Raspberry - Arduíno</i> . . . . .           | 51        |

# Listas de Figuras

|     |   |    |
|-----|---|----|
| 2.1 | Foto da plataforma utilizada.   | 6  |
| 2.3 | Desenho esquemático do sistema <i>ball and plate</i> .  | 13 |
| 2.4 | Cadeia cinemática vetorial de uma perna $i$   | 14 |
| 2.5 | Visão da base: coordenadas do vetor $B_i$ .   | 15 |
| 2.6 | Visão da plataforma móvel: Coordenadas do vetor $A_i$ .   | 16 |
| 2.7 | Obtenção do ângulo de inclinação $\alpha$ .   | 19 |
| 2.8 | Obtenção do ângulo de inclinação $\beta$  | 19 |
| 3.1 | Posição e dimensão padrão ao iniciar o algoritmo.   | 23 |
| 3.2 | Escala HSV.   | 23 |
| 3.3 | Detecção de duas bolas de tênis de mesa de cores diferentes.  | 24 |
| 4.1 | Malha de controle do processo.  | 25 |
| 4.2 | Ação de controle necessária para fazer com que a bolinha realize um percurso circular em torno da origem. | 26 |
| 4.3 | Blocos que realizam o cálculo dos ângulos $\Delta_i$ .  | 27 |
| 4.4 | Cálculo das alturas virtuais.   | 28 |
| 4.5 | Cálculo dos ângulos $\Delta_i$ a partir de $L_i$ .  | 29 |
| 4.6 | Bloco do processo, composto pelos atuadores e o sistema não linearizado.                                  | 29 |
| 4.7 | Representação em blocos do modelo não linearizado.  | 30 |
| 4.8 | Trajetória circular com centro na origem e raio de 10 cm.   | 30 |
| 5.1 | Fotografia da planta montada no LSI.  | 36 |
| 5.2 | Posição da bolinha lida em cada um dos quatro quadrantes da chapa.  | 37 |
| A.1 | <i>Raspberry Pi 3 B+</i>  | 46 |
| A.2 | Como conectar o Arduíno aos servomotores.   | 46 |
| A.3 | Exemplo de posicionamento da câmera.  | 47 |



# **Lista de Tabelas**

|     |  |    |
|-----|--|----|
| 2.1 | Momentos de inércia para esferas maciças e ocas . . . . .  | 12 |
| 2.2 | Entradas e saídas de cada sistema do modelo linearizado. . . . .   | 13 |
| 2.3 | Valores dos ângulos utilizados na montagem. . . . .  | 16 |
| 5.1 | Ângulos $\alpha$ e $\beta$ necessários para mover a bolinha para a origem, resultado da simulação, quando submetido a uma trajetória circular. . . . . | 37 |

# Capítulo 1

## Introdução

O sistema *ball and plate* consiste em uma planta bastante instável e não linear. Trata-se de uma plataforma que tem como objetivo controlar a posição de um corpo esférico que movimenta-se sobre uma superfície lisa (a mesa de trabalho), a qual é inclinada pela atuação de um manipulador paralelo. O projeto de um controlador para tal processo é um desafio que explora diversos conhecimentos de um estudante de engenharia, e pode ser resolvido utilizando várias técnicas de controle.

O projeto de controle e montagem desse sistema é bastante conhecido dentro da academia, tendo sido discutido em outros trabalhos como em Coelho e Mariani (2006), que traz uma abordagem mais moderna para a solução do problema, utilizando técnicas de *soft computing*, e como em Núñez et al. (2020), que resolve o projeto de compensador utilizando estratégias mais clássicas, usando representação do modelo em espaço de estados e projeto de controlador por realimentação de estados.

Este trabalho tem como objetivos realizar a montagem, a identificação e a simulação do sistema *ball and plate*, reportando detalhadamente cada etapa para permitir a reprodução por outros alunos. Como ele traz uma abordagem geral de todo e qualquer procedimento que envolve tal planta, é importante consultar também os trabalhos referenciados por este, que apresentam diferentes estratégias de controle para o mesmo problema, tornando este projeto uma oportunidade de aprendizado e aperfeiçoamento de conhecimentos para toda a comunidade acadêmica.

### 1.1 Motivação e Justificativa

O projeto descrito neste trabalho apresenta aspectos importantes e necessários para a formação completa de um Engenheiro de Controle e Automação, a partir do momento que envolve áreas importantes que são estudadas no curso, como, por exemplo, os conhecimentos relacionados ao projeto de controladores, a habilidade de manipular

um sistema de visão computacional, o qual é utilizado para acompanhar a posição do corpo esférico sobre a mesa. A realização deste projeto pode ser utilizada para fins didáticos com outros alunos, além de trazer referências diferentes para a solução do problema aqui discutido, oferecendo assim um amplo acesso à prática e aprendizado das habilidades do engenheiro.

## 1.2 Objetivos do Projeto

Tendo em vista o exposto acima, este projeto tem por objetivos:

- a) Construir e descrever o processo de montagem da planta, de forma que seja possível reproduzi-lo em ambientes diversos;
- b) Desenvolver um sistema de visão computacional para realizar o *tracking* da posição da bola sobre a chapa;
- c) Realizar a modelagem e identificação de parâmetros da planta, a princípio utilizando método caixa-branca;
- d) Para testar o modelo obtido, elaborar uma estratégia de controle simples que seja capaz de equilibrar o corpo esférico na chapa.

## 1.3 Local de Realização

O projeto será desenvolvido no Laboratório de Sistemas Inteligentes da Universidade Federal de Minas Gerais. O laboratório tem à disposição para uso dos alunos as ferramentas necessárias para realização do projeto além dos componentes que podem vir a ser necessários, como componentes eletrônicos básicos, fontes de alimentação, osciloscópios, chaves necessárias para apertar as porcas das pernas mecânicas que sustentam a base, bancadas de trabalho apropriadas para montagem, conexão à internet e computadores com os *softwares* necessários para modelagem e realização do controle da planta. O laboratório está localizado no Bloco 1 da Escola de Engenharia da UFMG, sendo o responsável por ele o prof. Dr. Ricardo de Oliveira Duarte.

## 1.4 Estrutura da Monografia

O trabalho está dividido em seis capítulos. Este capítulo apresentou uma introdução ao projeto a ser descrito nesta monografia, seus objetivos e o laboratório onde

o trabalho foi realizado. O Capítulo 2 descreve o sistema de forma mais detalhada, passando por sua montagem e funcionamento, de forma a deixar clara a operação da planta, identificando por último o processo montado. O Capítulo 3 descreve o sistema de visão computacional desenvolvido, explicitando as características que permitem a portabilidade da planta.

O Capítulo 4 aborda a metodologia de desenvolvimento da estratégia de controle, seguida pelo desenho da malha de controle no *Simulink*. No Capítulo 5, é feita a análise dos resultados obtidos pela montagem, identificação, visão computacional, simulação e aplicação do controlador na planta real. Por último, no Capítulo 6 tem-se a conclusão da monografia, algumas sugestões, e dificuldades encontradas na realização do projeto. Anexo ao texto, há um manual de montagem e uso da planta discorrido no Apêndice A, após as principais referências bibliográficas utilizadas durante o desenvolvimento do sistema.



# Capítulo 2

## Descrição e Identificação do Sistema *Ball and Plate*

Neste capítulo, é discutida a estrutura física do sistema utilizado para elaboração deste trabalho, descrevendo o processo de montagem e a identificação do modelo correspondente.

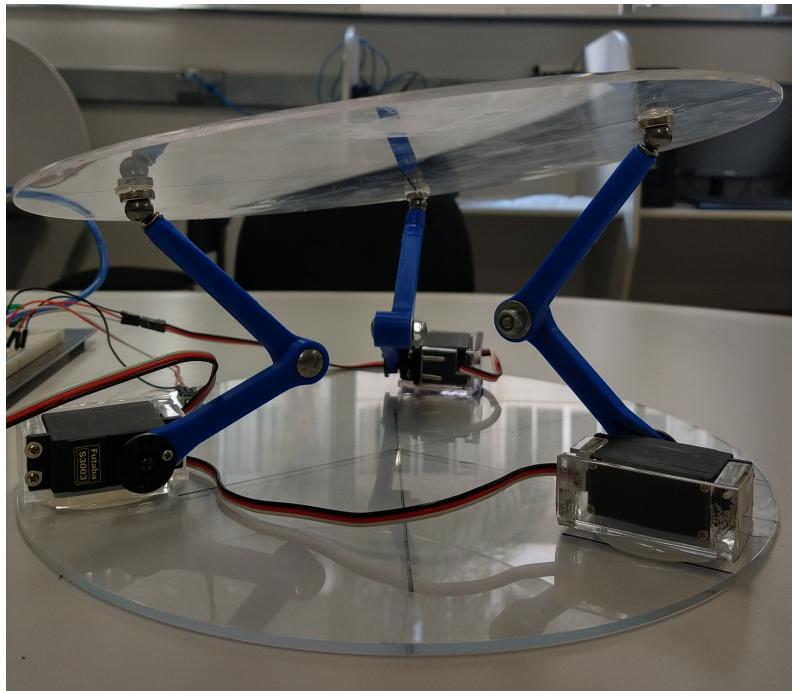
### 2.1 Sistema *Ball and Plate*

Um sistema físico *ball and plate* genérico é composto por três partes. A primeira é a plataforma móvel, também chamada de chapa, que, ao ser inclinada, movimenta a bola sobre sua superfície. Possui 4 graus de liberdade, 2 que dizem respeito às inclinações sofridas ( $\alpha, \beta$ ) e 2 referentes à posição da bola ( $x, y$ ). Acima desta plataforma móvel, é posicionada uma câmera que é responsável por realizar o *tracking* da bolinha, realimentando o sistema com as coordenadas atuais da posição e permitindo o controle.

A segunda parte é referente à plataforma que atua mecanicamente sobre a chapa para causar as inclinações descritas acima. São frequentes os trabalhos na academia que utilizam um sistema com 2 ou 6 graus de liberdade, este último chamado de Plataforma de Stewart, como por exemplo em Bang e Lee (2018) e Kassem et al. (2015). No caso deste projeto, foi escolhida uma plataforma com 3 graus de liberdade (*Degrees of Freedom*, na sigla em inglês *DOF*) apenas, pois tem mais flexibilidade de movimento que o caso com dois DOF, e é menos complexa que a de 6 graus.

A terceira parte do sistema é a base, que sustenta as duas outras partes citadas acima, onde estão posicionados os atuadores da plataforma de 3 *DOF*. Ela deve permanecer sobre uma superfície plana e reta, como o chão ou uma mesa, o que na prática é relativamente difícil de encontrar, mas espera-se que mediante estratégia de controle adequada, o sistema consiga contornar essa adversidade. A base também não deve

movimentar-se durante a atuação, visto que isso removeria a planta do campo de visão estabelecido pela configuração da câmera. O sistema utilizado neste trabalho pode ser observado na Figura 2.1.



**Figura 2.1:** Foto da plataforma utilizada.

Fonte: Repositório de imagens do autor.

## 2.2 Revisão da Literatura

O uso de métodos tradicionais para projeto de controladores PID (proporcional, integral e derivativo) são eficientes para sistemas de ordem reduzida, lineares e que apresentam parâmetros invariáveis com o tempo. Por sua vez, para processos mais complexos, como o *ball and plate* que é discutido neste trabalho, que é de alta ordem, instável e não linear, métodos mais modernos, que utilizam técnicas de *soft computing* por exemplo, podem apresentar resultados mais promissores.

Em Coelho e Mariani (2006), os autores utilizam técnicas de *soft computing* para elaborar um controlador para o sistema *ball and plate*, identificando o sistema utilizando uma rede neural de funções radiais de base e um algoritmo genético para sintonizar e otimizar os ganhos do controlador multivariável. Abordagens utilizando métodos inteligentes têm se tornado alvo de estudos, devido à complexidade que o contexto industrial pode vir a atingir nos dias atuais.

Como ponto forte, o projeto reportado em Coelho e Mariani (2006) traz uma abordagem diferente do projeto clássico de controlador PID, o que torna o compensador

menos dependente do ajuste humano - consequentemente, evitando o erro do mesmo - além de ter ajustes mais rápidos dependendo dos estímulos externos aos quais o sistema for submetido. Por outro lado, o artigo não traz comparações do método proposto com outras técnicas de *soft computing*, como o uso de diferentes tipos de redes neurais, ou uma estratégia de controle própria para sistemas não lineares, como *sliding mode control* (SMC).

Por sua vez, em Núñez et al. (2020), o manipulador paralelo do processo *ball and plate* possui dois graus de liberdade. Os autores obtêm a representação em espaço de estados do sistema e, para resolver o problema de centralização da bola, elaboram um controlador de realimentação de estados usando um sistema de visão computacional para medir a posição da esfera. Diferente de outros artigos encontrados, esse traz uma abordagem mais simples e clássica, que pode estar sujeita a problemas que são contornados em outros artigos, por exemplo o erro humano que pode ocorrer na sintonização dos ganhos do controlador e a lentidão da adaptação do sistema. Em compensação, tal estratégia se mostrou eficiente para controlar o processo *ball and plate*.

Bang e Lee (2018) trazem um trabalho voltado para a montagem da plataforma para fins educacionais, onde explicam com detalhes uma estratégia de construção do processo de forma que seja barato, eficiente e que seja possível controlar. Nessa montagem, os autores usam um painel *touchscreen* para obter a posição do corpo esférico e uma plataforma de Stewart que utiliza seis servomotores.

Em Moreno-Armendariz et al. (2010), o mesmo processo é modelado e controlado utilizando controle adaptativo indireto. Nesse trabalho, redes neurais da subclasse FCMAC (*Fuzzy cerebellar model articulation controller*) identificam e controlam o sistema. De acordo com Guan et al. (2017), redes CMAC têm a capacidade de reconhecer e controlar sistemas dinâmicos não lineares pela sua capacidade de generalização, simplicidade e rápido aprendizado.

De acordo com Guan et al. (2017), juntando as lógicas de controle nebuloso e redes CMAC (gerando assim as redes FCMAC), a precisão da função de aproximação de uma CMAC comum é bastante melhorada. O algoritmo então passa a modelar, identificar e controlar sistemas não lineares de forma mais flexível, independente do quanto incertos os sistemas forem.

Em Okafor et al. (2021) um controle PID baseado em aprendizado para o *ball and plate* é desenvolvido, sob a premissa de que a sintonização de ganhos do controle PID manual é tediosa. Os autores então realizam a comparação de duas soluções de controle para esse problema: três variantes de um controlador PID baseado em inteligência artificial de aprendizado profundo reforçado comparadas com um algoritmo genético e um controlador PID clássico.

## 2.3 Design e Montagem da planta

### 2.3.1 Plataforma Paralela de 3 Graus de Liberdade

A plataforma paralela que movimenta a chapa é composta por três servomotores Futaba S3003 (USINAINFO, 2022), utilizando um *horn* circular que vem com o kit básico do motor, o qual foi parafusado com o primeiro segmento da perna. Para fixar os servos, foi construída uma peça de acrílico personalizada que permite a colagem entre dois materiais iguais, sem risco de danificar os motores com a cola utilizada.

Além dos servos, os atuadores ainda são compostos pelas pernas do manipulador, divididas em dois segmentos. Para fins de desenvolvimento matemático, o primeiro pode ser considerado como uma extensão do *horn*, e é fixado no segundo segmento utilizando parafuso francês e porca auto-travante para evitar que o movimento rotacional da junta afrouxe o parafuso. No final do segundo segmento há uma bolinha metálica parafusada (junta esférica) que é atraída magneticamente pelos ímãs de neodímio fixados na chapa, permitindo o "deslize" da ponta da perna sob a superfície atuada. O design das pernas foi feito com base no que foi disponibilizado no Instructables (2019), em um projeto de um *Ball Balancing System*.

### 2.3.2 Base

A base utilizada para fixar a estrutura da planta é um círculo de acrílico de diâmetro igual a 30 centímetros, com 3 milímetros de espessura. Nela, as peças de acrílico usadas para acomodar os servomotores foram coladas utilizando cola quente. Foi feita a escolha desse tipo de cola visto que a sua temperatura de fusão é menor que a do acrílico, permitindo troca do material caso haja necessidade. No entanto, entende-se que essa não é a melhor maneira de fixação das peças e, para estágios futuros do projeto, seria interessante substituí-la por um método mais permanente, como fitas dupla faces de maior aderência.

### 2.3.3 Plataforma Móvel

A peça usada para a chapa é idêntica à usada para a base. Esse formato e essas medidas foram escolhidas por conveniência, mas o sistema poderia utilizar uma chapa maior para aumentar o tempo máximo de atuação dos motores até que a bola escape da chapa, mas deve-se atentar para possíveis mudanças no cálculo da cinemática inversa. Abaixo dela foram colados, também com cola quente, os ímãs de neodímio para fixação com as esferas das pernas. Os ímãs têm uma de suas faces plana e a outra no formato de semi-esfera, permitindo assim o "deslize" citado anteriormente.

### 2.3.4 Sistema de Visão Computacional

Uma câmera, conectada via USB ao *Raspberry Pi*, é posicionada com a lente aproximadamente paralela à plataforma móvel, acima dela e sustentada por um pedestal para microfone flexível da marca *SYANG*. Essa câmera será responsável por realizar o sensoriamento do sistema, utilizando um algoritmo na linguagem *Python* e a biblioteca *OpenCV*. O sistema construído para executar essa função será discutido com maior detalhamento no Capítulo 3.

### 2.3.5 Microcontrolador Raspberry Pi

Para realizar o controle do sistema *ball and plate* são utilizadas as placas *Raspberry Pi 3 B+* e *Arduíno Mega*, sendo a primeira responsável por ler a saída da câmera (sensor), calcular os ângulos necessários para estabilizar a posição da bola e enviá-los como sinal de controle para a segunda placa, que escreve-as nos servomotores (atuadores). O *setup* do sistema, o desenvolvimento do controlador e o algoritmo de controle utilizados serão discutidos com maior profundidade nos Capítulos 4, 5 e no Apêndice A.

## 2.4 Identificação e Análise Cinemática da Planta

Nesta seção é realizada a identificação fenomenológica do sistema *ball and plate* e a análise cinemática da plataforma que sustenta a chapa onde a bola se movimenta, de forma que a combinação desses dois itens representará o modelo da planta.

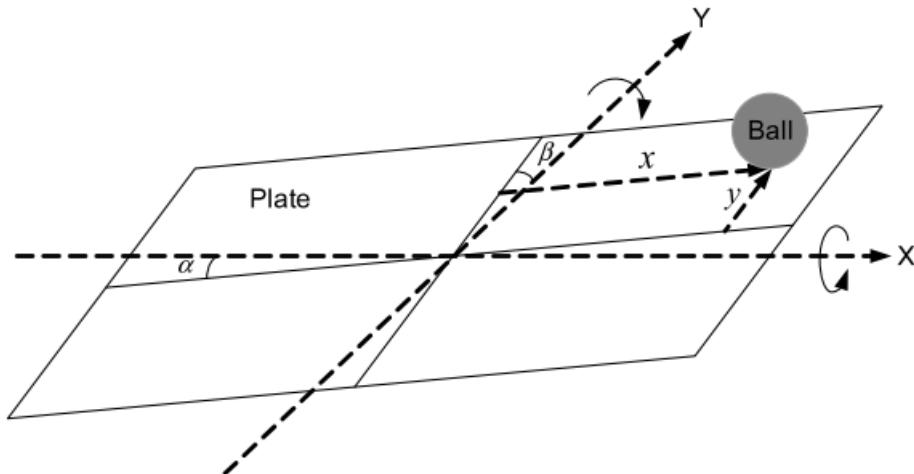
### 2.4.1 Modelo matemático do sistema *ball and plate*

O modelo de um sistema *ball and plate* genérico, representado na Figura 2.2, pode ser obtido utilizando a equação geral de Euler-Lagrange, a qual pode ser usada para representar uma ampla gama de sistemas mecânicos, como discutido em Ali et al. (2019). A relação entre as energias cinética e potencial do sistema pode ser expressa da seguinte forma:

$$\mathcal{L} = T - V, \quad (2.1)$$

sendo  $\mathcal{L}$  o parâmetro Lagrangiano,  $T$  representa as energias cinéticas da bola e da plataforma e  $V$  corresponde à energia potencial da bola. A equação geral de Euler-Lagrange tem a seguinte forma:

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = S, \quad (2.2)$$



**Figura 2.2:** Esquemático de um sistema *ball and plate* genérico.

Fonte: Bang e Lee (2018)

sendo  $S$  utilizado para expressar a direção do sistema coordenado na qual a equação está sendo aplicada.

Para ser possível desenvolver um modelo geral para o comportamento da chapa, faz-se necessário assumir como verdadeiras quatro proposições:

- i) A bola nunca perde contato com a superfície;
- ii) A bola não escorrega sobre a plataforma;
- iii) Todas as forças de fricção e movimentos rotacionais são desprezados;
- iv) A chapa não perde contato com o suporte, portanto não se move horizontalmente.

Tanto Ali et al. (2019) quanto Kassem et al. (2015) nos mostram que, independentemente do suporte que sustenta a chapa, o seu comportamento na prática será o mesmo e terá quatro graus de liberdade: dois derivados do movimento da bola e outros dois provenientes da inclinação da chapa.

Os ângulos de inclinação da chapa serão representados pelas variáveis rotacionais  $\alpha$  e  $\beta$ , e  $x$  e  $y$  representarão a posição da bola no plano coordenado visualizado sobre a posição de repouso da chapa. Sendo  $I_b$  o momento de inércia da bola,  $m$  sua massa e  $r$  o seu raio, a energia cinética da bola é dada por:

$$T_b = \frac{1}{2}(m + \frac{I_b}{r^2})(\dot{x}^2 + \dot{y}^2), \quad (2.3)$$

e sendo  $I_p$  o momento de inércia da chapa, a energia cinética dessa pode ser expressa da seguinte forma:

$$T_p = \frac{1}{2}(I_p + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}m(x\dot{\alpha} + y\dot{\beta}) \quad (2.4)$$

Por último, a energia potencial da bola em referência ao centro da chapa é expressa pelas equações:

$$V_x = mgx\text{sen}(\alpha) \quad (2.5)$$

$$V_y = mgys\text{en}(\beta) \quad (2.6)$$

Substituindo as equações de (2.3) a (2.6) na equação (2.1), chegamos à seguinte representação para a energia do sistema:

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}(I_p + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}\left(m + \frac{I_b}{r^2}\right)(\dot{x}^2 + \dot{y}^2) \\ & + \frac{1}{2}m(x\dot{\alpha} + y\dot{\beta})^2 - mg(x\text{sen}(\alpha) + y\text{sen}(\beta)) \end{aligned} \quad (2.7)$$

A partir de (2.7), é possível obter as derivadas parciais em relação a cada variável que representa um grau de liberdade do sistema e suas primeiras derivadas, obtendo assim um conjunto de 8 expressões que podem ser posteriormente substituídas em (2.2), permitindo explicitar as quatro expressões que representam a dinâmica do sistema básico *ball and plate*:

$$\begin{aligned} (I_p + I_b + mx^2)\ddot{\alpha} + 2m\dot{x}\dot{\alpha} + mxy\ddot{\beta} + m\dot{x}\dot{\beta} + mx\dot{y}\dot{\beta} + mgx\cos(\alpha) &= \tau_x \\ (I_p + I_b + my^2)\ddot{\beta} + 2my\dot{y}\dot{\beta} + mxy\ddot{\alpha} + m\dot{x}\dot{y}\dot{\alpha} + mx\dot{y}\dot{\alpha} + mgy\cos(\beta) &= \tau_y \\ \left(m + \frac{I_b}{r^2}\right)\ddot{x} + mg\text{sen}(\alpha) - mx\dot{\alpha}^2 - my\dot{\alpha}\dot{\beta} &= F_x \\ \left(m + \frac{I_b}{r^2}\right)\ddot{y} + mg\text{sen}(\beta) - mx\dot{\beta}^2 - my\dot{\alpha}\dot{\beta} &= F_y \end{aligned} \quad (2.8)$$

Trata-se de um sistema multivariável, não linear e de alto grau de acoplamento. Partindo das suposições feitas anteriormente, podemos assumir que  $F_x = F_y = 0$  devido à ausência de escorregamento da bola sobre a chapa, visto que tais variáveis representam forças externas que atuam sobre a posição da bola. Também é possível

descartar as duas primeiras equações de (2.8), pois assume-se que os momentos de torque do sistema são desprezíveis.

Para simplificar o sistema ainda mais, é necessário assumir que os ângulos de rotação são pequenos a ponto de ser possível aproximar o seno dos ângulos ao próprio valor desse. Também é ignorada a força centrífuga sobre a bola, visto que ela é pequena em comparação à gravidade (BANG; LEE, 2018).

O sistema então reduz-se à:

$$\ddot{x} + \frac{5}{7}g\alpha = 0 \quad (2.9)$$

$$\ddot{y} + \frac{5}{7}g\beta = 0 \quad (2.10)$$

$$\ddot{x} + \frac{3}{5}g\alpha = 0 \quad (2.11)$$

$$\ddot{y} + \frac{3}{5}g\beta = 0 \quad (2.12)$$

em que (2.9) e (2.10) referem-se ao sistema que usa uma bola maciça, (2.11) e (2.12) ao que utiliza uma esfera oca, cuja desigualdade se dá pela diferença entre os momentos de inércia. Os valores de cada um podem ser consultados em tabelas de momentos de inércia de objetos de diversos formatos, mas um recorte com o valor para cada tipo de esfera pode ser observado na Tabela 2.1:

**Tabela 2.1:** Momentos de inércia para esferas maciças e ocas

|               | Momento de inércia |
|---------------|--------------------|
| Esfera maciça | $\frac{2}{3}MR^2$  |
| Esfera oca    | $\frac{2}{5}MR^2$  |

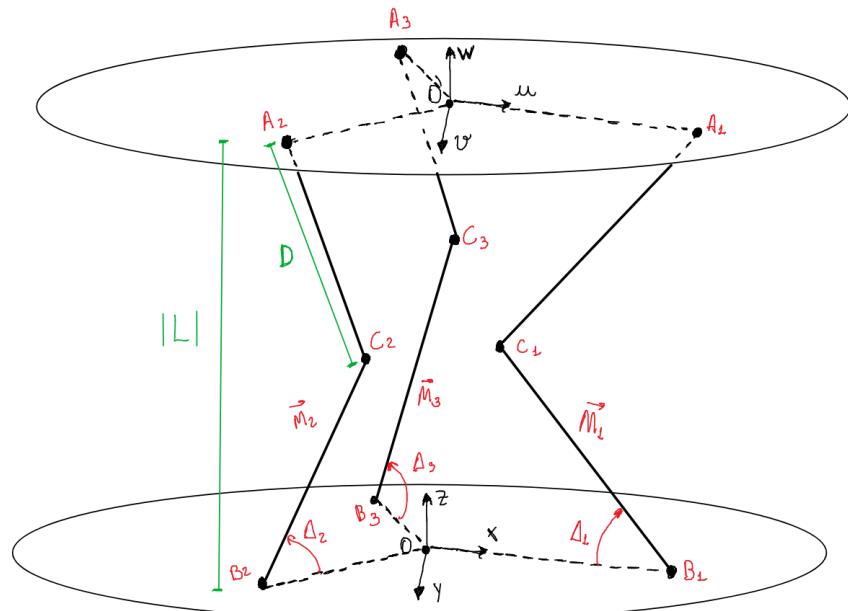
As equações de (2.9) a (2.12) são desacopladas e lineares, de forma que é possível tratá-las como dois sistemas *single input, single output* (SISO) diferentes, que é uma forma de enxergar o sistema mais simples e criativa do que simplesmente assumi-lo como *multiple input, multiple output* (MIMO). Assim, será preciso desenvolver apenas um controlador que poderá ser usado para ambos os sistemas, devido à similaridade existente entre eles. A Tabela 2.2 mostra a entrada e a saída de cada sistema.

**Tabela 2.2:** Entradas e saídas de cada sistema do modelo linearizado.

|                           | Entrada  | Saída |
|---------------------------|----------|-------|
| Sistema 1 (2.9) e (2.11)  | $\alpha$ | x     |
| Sistema 2 (2.10) e (2.12) | $\beta$  | y     |

## 2.4.2 Estudo da cinemática inversa do manipulador paralelo de três graus de liberdade

Além do modelo descrito acima, é necessário descrever o comportamento do manipulador paralelo utilizado para causar as inclinações. Um esquema geral da planta é mostrado na Figura 2.3. A atuação rotatória nos permite utilizar os servomotores e os *horns* ao invés de atuadores prismáticos, que para plataformas de Stewart (6 DOF), por exemplo, são mais comumente utilizados. O objetivo do estudo da cinemática inversa é mapear os ângulos de rotação de cada servomotor com os ângulos de *pitch* e *roll*, de forma que as entradas do sistema em questão se tornem os ângulos  $\Delta_i$  de cada servo e não os ângulos  $\alpha$  e  $\beta$ . Tal estudo será realizado com base no que é mostrado em Szufnarowski (2013) e Bang e Lee (2018).

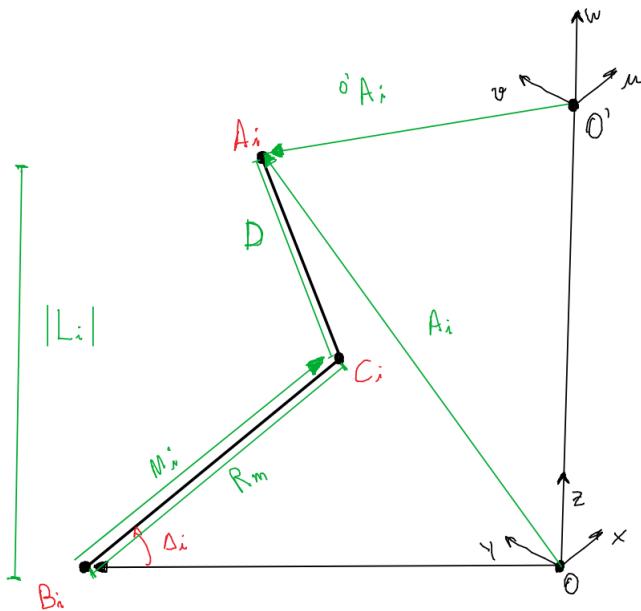


**Figura 2.3:** Desenho esquemático do sistema *ball and plate*.

Fonte: Elaborado pelo autor.

A Figura 2.4 mostra a relação entre os principais vetores utilizados para o desenvolvimento. A partir da figura, podemos concluir que o vetor  $L_i$  cujo módulo representa a altura virtual de uma perna, pode ser obtido a partir da diferença entre os vetores dos pontos de conexão do manipulador paralelo na chapa e na base:

$$L_i = A_i - B_i \quad (2.13)$$



**Figura 2.4:** Cadeia cinemática vetorial de uma perna  $i$

Fonte: Elaborado pelo autor.

As matrizes de rotação e translação,  $\mathcal{R}_b$  e  $\mathcal{T}_b$  respectivamente, variam de acordo com a movimentação da chapa. Utilizando-as, é possível relacionar os vetores de posição que partem das origens dos sistemas de coordenadas da base e da chapa até o ponto de conexão  $A_i$  entre as pernas e a plataforma móvel:

$$A_i = \mathcal{R}_b A_i^{O'} + \mathcal{T}_b \quad (2.14)$$

que ao ser substituído em (2.13):

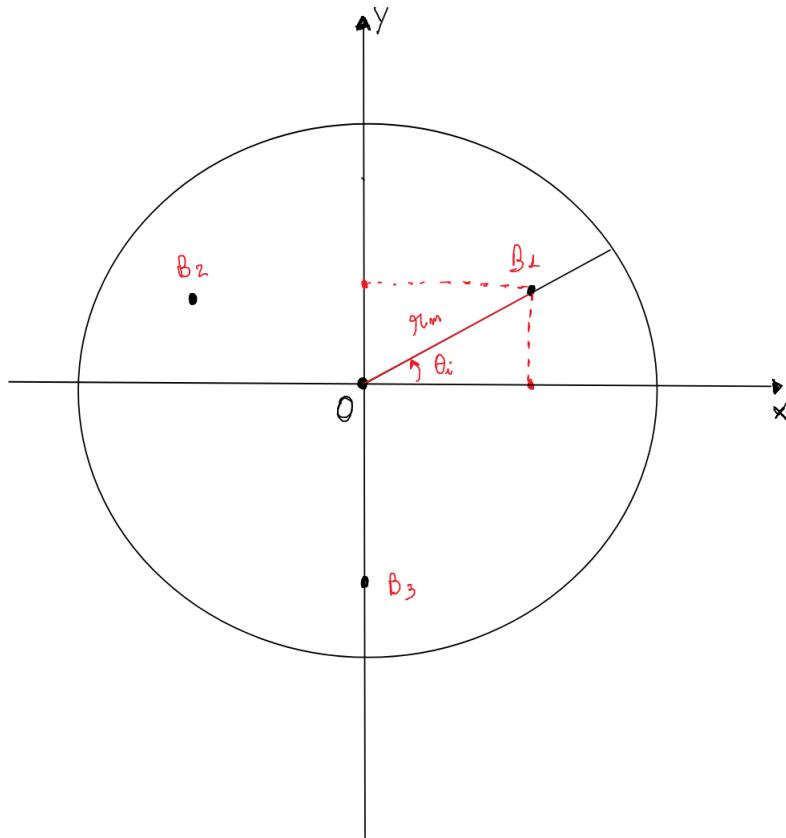
$$L_i = \mathcal{R}_b A_i^{O'} + \mathcal{T}_b - B_i \quad (2.15)$$

$B_i$  representa o vetor que parte da origem do sistema coordenado O-xyz e vai até um dos pontos de fixação na base, que aqui é aproximado para o ponto em que o primeiro segmento da perna liga-se ao motor, visto que a pequena diferença de altura trará discrepâncias desprezíveis. Por sua vez,  $A_i^{O'}$  é o vetor que parte do sistema de coordenadas O'-uvw até um dos pontos de fixação das pernas nas plataformas móveis. A partir da análise das Figuras 2.5 e 2.6, esses vetores podem ser dados como:

$$B_i = \begin{bmatrix} r_m \cos \theta_i \\ r_m \sin \theta_i \\ 0 \end{bmatrix} \quad (2.16)$$

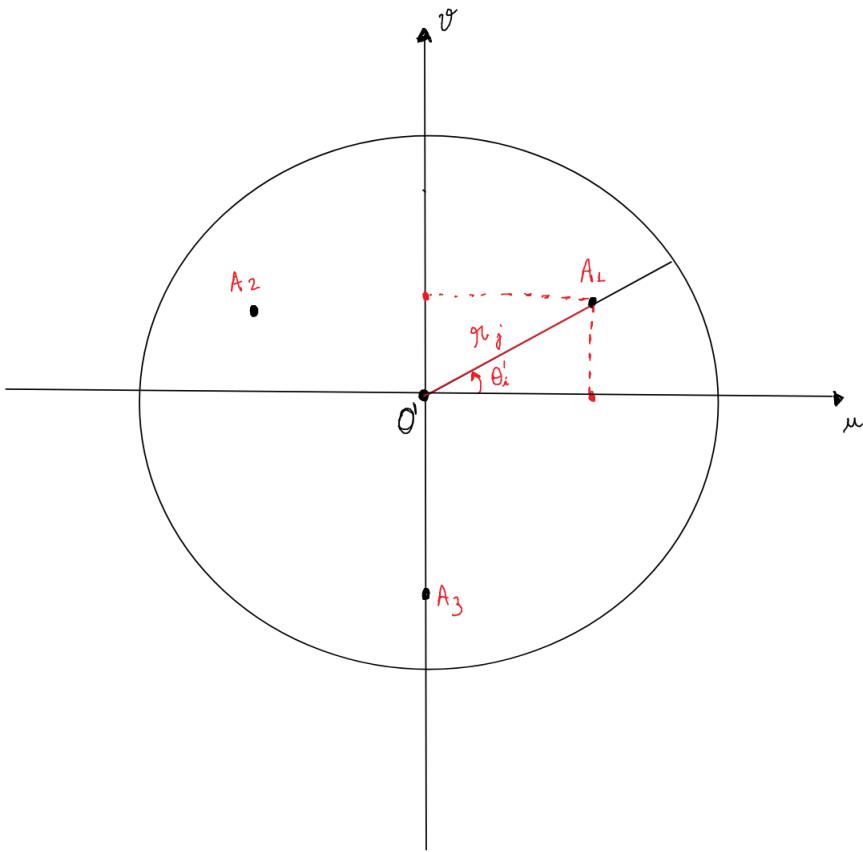
$$A_i^{O'} = \begin{bmatrix} r_j \cos \theta'_i \\ r_j \sin \theta'_i \\ 0 \end{bmatrix} \quad (2.17)$$

em que  $r_m$  é a distância da origem até o ponto  $B_i$  e  $\theta_i$  é o ângulo entre o eixo das abscissas e o vetor  $B_i$ . Os equivalentes dessas medidas para a plataforma móvel são, respectivamente,  $r_j$  e  $\theta'_i$ . Os valores dos ângulos utilizados no projeto podem ser observados na Tabela 2.3.



**Figura 2.5:** Visão da base: coordenadas do vetor  $B_i$ .  
Fonte: Elaborado pelo autor.

Como demonstrado em Zhao et al. (2015), o movimento do ponto  $C_i$  (vide Figura 2.4) em um sistema de coordenadas local pode ter sua dinâmica pensada como uma rotação  $\mathcal{R}_z$  em torno do eixo  $z$ , seguida de outra rotação  $\mathcal{R}_y$  em torno do eixo  $y$ .



**Figura 2.6:** Visão da plataforma móvel: Coordenadas do vetor  $A_i$ .  
Fonte: Elaborado pelo autor.

|             | 1º | 2º   | 3º   |
|-------------|----|------|------|
| $\theta_i$  | 0º | 120º | 240º |
| $\theta'_i$ | 0º | 120º | 240º |

**Tabela 2.3:** Valores dos ângulos utilizados na montagem.

Portanto, o vetor  $M_i$ , que parte da origem até  $C_i$  tem sua equação dada por:

$$Mi = \begin{bmatrix} x_{mi} \\ y_{mi} \\ z_{mi} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \mathcal{R}_z(\theta_i) \mathcal{R}_y(\Delta_i) \begin{bmatrix} R_m \\ 0 \\ 0 \end{bmatrix}, \quad (2.18)$$

em que  $R_m$  é o comprimento do primeiro segmento da perna e  $\Delta_i$  é o ângulo de rotação do servo. As matrizes de rotação são definidas como:

$$\mathcal{R}_y(\kappa) = \begin{bmatrix} \cos \kappa & 0 & -\sin \kappa \\ 0 & 1 & 0 \\ \sin \kappa & 0 & \cos \kappa \end{bmatrix}, \mathcal{R}_z(\kappa) = \begin{bmatrix} \cos \kappa & -\sin \kappa & 0 \\ \sin \kappa & \cos \kappa & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.19)$$

Substituindo (2.19) em (2.18),  $M_i$  pode ser expresso como:

$$M_i(\Delta_i) = \begin{bmatrix} R_m \cos \Delta_i \cos \theta_i + x_i \\ R_m \cos \Delta_i \sin \theta_i + y_i \\ R_m \sin \Delta_i + z_i \end{bmatrix} \quad (2.20)$$

Como visto na Figura 2.4, os comprimentos dos segmentos das pernas e a altura virtual da mesma,  $R_m$ ,  $D$  e  $|L_i|$ , respectivamente, podem ter suas relações vetoriais expressas das seguintes formas:

$$\begin{aligned} R_m^2 &= (M_i(\Delta_i) - B_i)^T (M_i(\Delta_i) - B_i), \\ D^2 &= (A_i - M_i(\Delta_i))^T (A_i - M_i(\Delta_i)), \\ |L_i|^2 &= (A_i - B_i)^T (A_i - B_i). \end{aligned} \quad (2.21)$$

para  $i \in \{1,2,3\}$ . Combinando as equações de (2.21), chegamos em:

$$|L_i|^2 - D^2 + R_m^2 = 2(B_i - M_i(\Delta_i))^T (B_i - A_i), \quad (2.22)$$

substituindo a equação (2.20) em (2.22):

$$\begin{aligned} |L_i|^2 - D^2 + R_m^2 &= 2R_m(z_i^{o'} - z_i) \sin \Delta_i \\ &\quad + 2R_m[(x_i^{o'} - x_i) \cos \theta_i + (y_i^{o'} - y_i)(\sin \theta_i)] \cos \Delta_i, \end{aligned} \quad (2.23)$$

que pode ser escrita de forma mais simples como:

$$c_i = a_i \sin \Delta_i + b_i \cos \Delta_i, \quad (2.24)$$

sendo:

$$\begin{aligned}
a_i &= 2R_m(z_i^{o'} - z_i), \\
b_i &= 2R_m[(x_i^{o'} - x_i) \cos \theta_i + (y_i^{o'} - y_i)(\operatorname{sen} \theta_i)], \\
c_i &= |L_i|^2 - D^2 + R_m^2.
\end{aligned}
\tag{2.25}$$

A partir da equação (2.24) e usando a manipulação mostrada em (2.26), chegamos à expressão que nos permite determinar o ângulo de rotação de cada servomotor mostrada em (2.27):

$$a \operatorname{sen} \psi + b \cos \psi = \sqrt{a^2 + b^2} \operatorname{sen} \left( \psi + \arctan \frac{b}{a} \right) \tag{2.26}$$

$$\Delta_i = \arcsin \left( \frac{c_i}{\sqrt{a_i^2 + b_i^2}} \right) - \arctan \left( \frac{b_i}{a_i} \right), \tag{2.27}$$

podendo ser reescrita sem o termo  $b_i$ , visto que na montagem os pontos  $A_i$  e  $B_i$  foram posicionados com as mesmas coordenadas  $(x, y)$ , que assumindo  $a_i > 0$ , torna-se em:

$$\Delta_i = \arcsin \left( \frac{c_i}{a_i} \right) \tag{2.28}$$

Também é necessário relacionar os ângulos de entrada  $\alpha$  e  $\beta$  com as alturas virtuais  $L_i$  que cada perna do manipulador assume para que a chapa atinja essas inclinações. Para isso, leva-se em conta a relação geométrica mostrada em (2.29) e (2.30), obtidas com base na análise das Figuras 2.7 e 2.8.

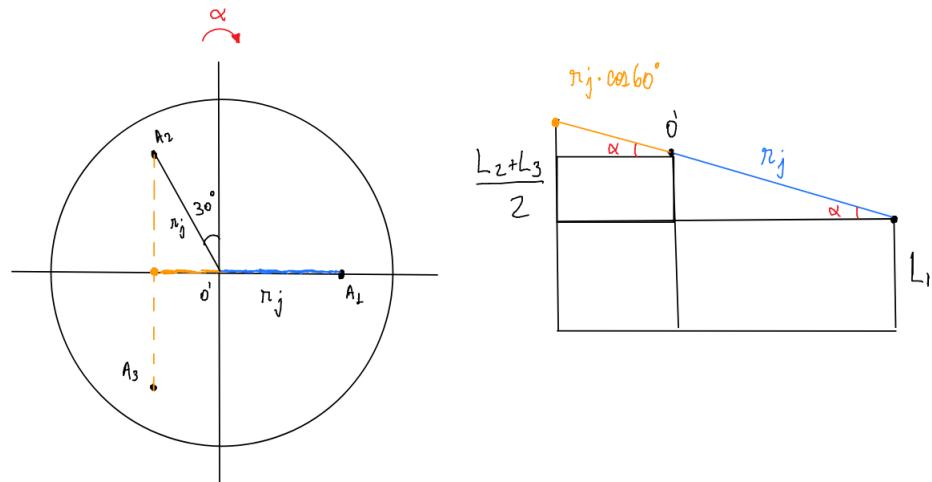
$$\alpha = \arcsin \left( \frac{L_2 + L_3 - 2L_1}{3r_j} \right) \tag{2.29}$$

$$\beta = \arcsin \left( \frac{L_3 - L_2}{\sqrt{3}r_j} \right) \tag{2.30}$$

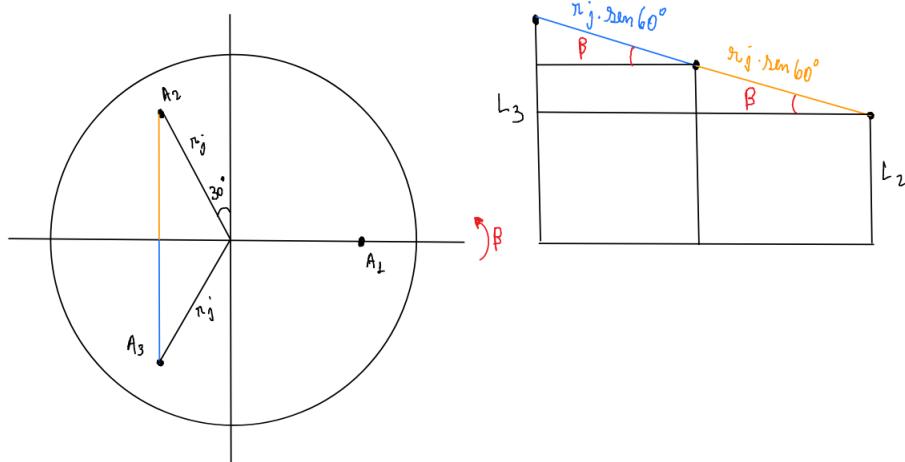
Com a obtenção dessas últimas relações, agora é possível realizar um mapeamento entre os ângulos  $\Delta_i$  e  $(\alpha, \beta)$ , concluindo assim o principal objetivo do estudo da cinemática inversa.

## 2.5 Resumo do Capítulo

Neste capítulo foi apresentada uma visão geral do funcionamento físico da planta *ball and plate* utilizada neste projeto. Inicialmente, foram apresentadas as principais

**Figura 2.7:** Obtenção do ângulo de inclinação  $\alpha$ .

Fonte: Elaborado pelo autor.

**Figura 2.8:** Obtenção do ângulo de inclinação  $\beta$ 

Fonte: Elaborado pelo autor.

fontes utilizadas para o desenvolvimento teórico e prático, sendo em sua maioria artigos que abordam de forma geral a modelagem e o controle de sistemas similares, variando graus de liberdade da plataforma, sensor utilizado, atuadores utilizados, entre outros.

Também foi apresentado um panorama de como foi o processo de montagem da planta, passando por todos os componentes que fazer parte dela, os microcontroladores utilizados, o sistema de sensoriamento e uma descrição da atuação. Dessa forma, é possível reproduzir o que foi executado e pensar em melhorias para projetos futuros.

Em seguida, foi necessário mostrar o desenvolvimento matemático para obtenção do modelo do sistema. Um sistema *ball and plate* genérico geralmente apresenta sua função de transferência similar em muitos trabalhos, mediante às simplificações feitas. O que muda bastante a física do sistema é o manipulador da plataforma, que no caso excepcional deste trabalho, tem 3 graus de liberdade e atuadores rotacionais. Por isso, foi necessário realizar um estudo da cinemática inversa dele, para que seja obtido um mapeamento do ângulo aplicado a cada servomotor e os ângulos de *pitch* e *roll* da chapa.

# Capítulo 3

## Descrição do Sistema de Visão Computacional

Neste capítulo, é discutido o desenvolvimento do sistema de visão computacional utilizado como sensor do sistema *ball and plate* montado, que consiste em uma câmera posicionada acima da plataforma, com algoritmo que realiza o *tracking* da bolinha. A implementação é realizada usando *Python* com a biblioteca *OpenCV*, que requer a biblioteca *NumPy* para correto funcionamento.

### 3.1 Sistema de Visão Computacional

Devido à alta instabilidade e não linearidade do sistema *ball and plate*, é necessário que o mecanismo de detecção da posição da bola seja rápido e eficiente. Os dois principais sistemas de sensoriamento utilizados para este tipo de planta são o *touchscreen* e sistemas de visão computacional. No primeiro, a plataforma móvel funciona como *touch panel*, e para determinados tipos de bolas conseguem manter um *tracking* constante da sua posição.

Os sistemas de visão necessitam de uma câmera e de um suporte para mantê-la paralela à chapa. Essa foi a solução escolhida neste trabalho, visto que ela flexibiliza o material, a massa e a dimensão da bola, além de proporcionar diferentes esquemas de montagem. O suporte da câmera (*JinYan 0.3 MP*) não é fixo em nenhuma superfície, permitindo assim que a planta possa ser removida do laboratório quando houver necessidade. Além disso, a distância entre a chapa e a câmera poderá ser alterada sem que o funcionamento seja prejudicado.

## 3.2 Análise do Problema

O principal trabalho utilizado como referência para o projeto deste algoritmo é Soler (2017), que traz um mecanismo mais sofisticado de *tracking* de uma bolinha de tênis de mesa, durante uma partida real. Nele, é possível mapear a trajetória da esfera dentro e fora da mesa, e consegue detectar os quiques da bola ao analisar a mudança da trajetória parabólica do objeto com certa precisão.

Distintivamente do trabalho citado, aqui o ambiente é mais controlado e a velocidade da bola é, em média, bem menor. Além disso, é necessário que haja possibilidade de reposicionamento da câmera e da plataforma, exigindo maior flexibilidade do algoritmo, uma vez que é possível que a câmera seja posicionada em diferentes alturas.

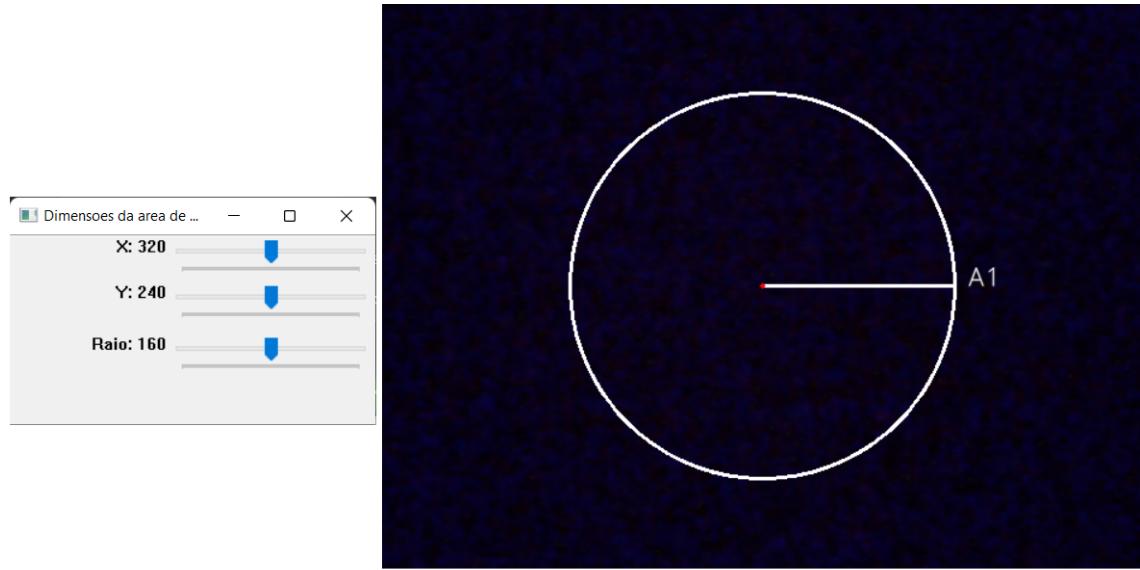
Os principais elementos necessários no algoritmo são: uma forma simples e flexível de detectar a posição da chapa, capacidade de lidar com mesas de tamanhos diferentes (já que pode haver variação na altura), e um ajuste adaptativo para detecção da cor da bola. Preferencialmente, para garantir bom funcionamento do último, a esfera deverá ter uma cor bastante distinta dos outros elementos capturados pela câmera.

## 3.3 Implementação do Algoritmo

Para detectar a mesa, optou-se por desenhar uma área circular que deverá ser ajustada manualmente de acordo com as dimensões da chapa, ao invés de detectá-la automaticamente. Essa escolha foi feita por três motivos: ela evita alterações físicas na planta, como pintar as extremidades de branco, para ter um resultado mais robusto e pela simplicidade de implementação. Com esta abordagem, pode-se alterar a posição e o raio da mesa livremente utilizando *sliders*, como pode ser visto na Figura 3.1. É importante salientar que  $A_1$  deverá ser posicionado corretamente sobre a linha marcada.

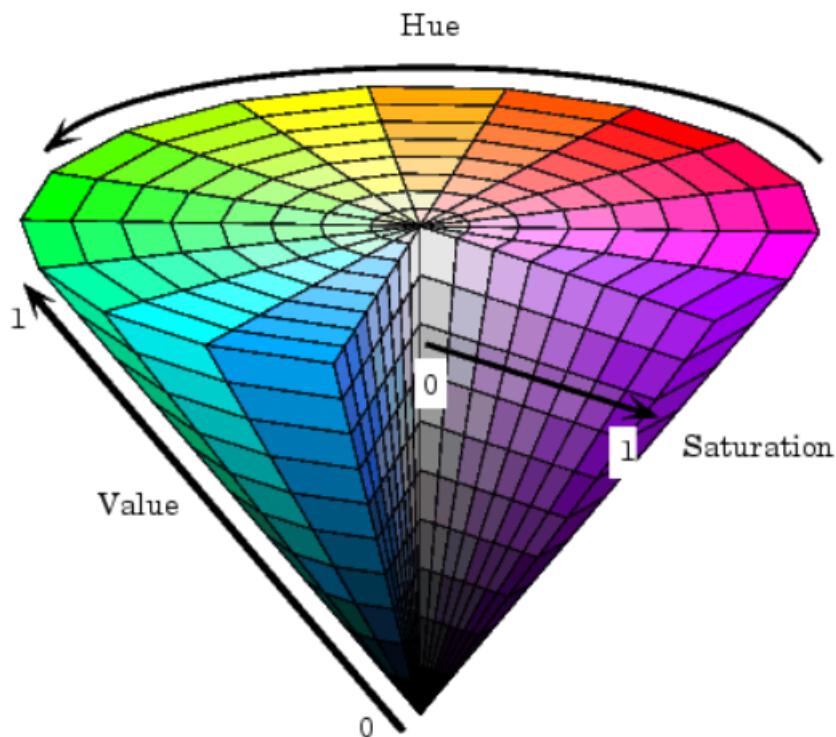
Primeiramente, para detecção da bolinha, escolheu-se uma com a cor laranja, que é bastante diferente das outras cores que serão capturadas pela câmera. A escala de cores utilizada foi a HSV (Figura 3.2, que utiliza matiz (em inglês, *hue*), saturação e valor das cores, em uma escala mais linear que o tradicional RGB, se mostrando muito mais adequado para o objetivo do projeto. Para ajustar os valores HSV também foram usados *sliders*, para definir os valores mínimos e máximos de cada uma das três variáveis, e como pode ser visto na Figura 3.3, o ajuste é melhor e mais simples para cores mais "chamativas".

A detecção e marcação da bolinha dentro da área de interesse ocorre por meio da sobreposição (utilizando a operação *AND*) de duas máscaras: *mask*, que detecta a cor selecionada pelos *sliders* em todo o espaço observado e *mask1* que faz a detecção da



**Figura 3.1:** Posição e dimensão padrão ao iniciar o algoritmo.

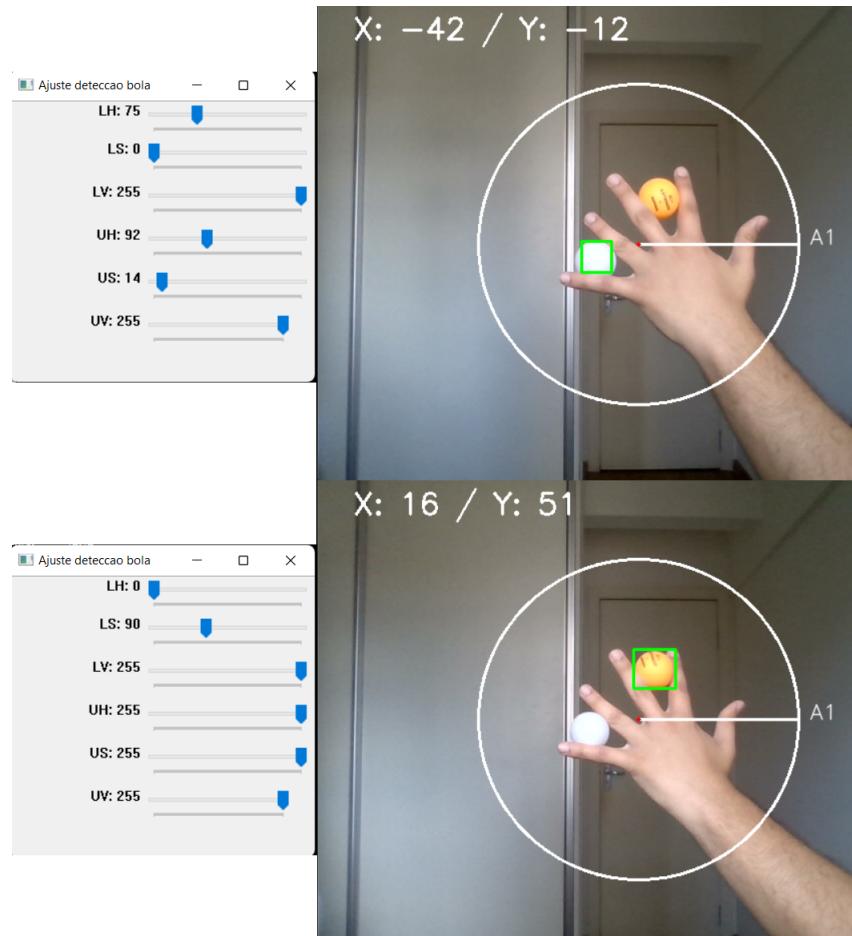
Fonte: Elaborado pelo autor.



**Figura 3.2:** Escala HSV.

Fonte: <<http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/images/color11.html>>. Acesso em: 20 out. 2022.

chapa. A operação binária das duas resulta uma terceira máscara, *mask2*, que é a principal do algoritmo e resulta na captura mostrada na Figura 3.3.



**Figura 3.3:** Detecção de duas bolas de tênis de cores diferentes.  
Fonte: Elaborado pelo autor.

### 3.4 Resumo do Capítulo

Neste capítulo foi apresentada uma concepção geral de alto nível do funcionamento do algoritmo utilizado como sensor na malha de controle. Primeiramente, o problema da detecção e a justificativa do uso da câmera foi discutido. Em seguida, o problema é destrinchado e comparado com o trabalho usado como base para o capítulo. Por último, o funcionamento do algoritmo e os motivos pela escolha dos mecanismos desenvolvidos foram mostrados, com uma abordagem de simples entendimento.

# Capítulo 4

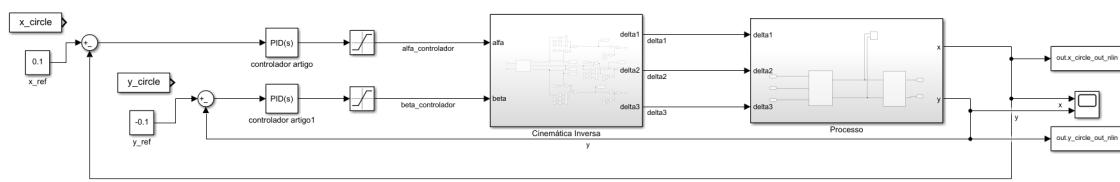
## Simulação e Controle

Neste capítulo, é discutido o processo de simulação da planta, as ferramentas utilizadas e a construção da malha de controle.

### 4.1 Malha de controle

A malha de controle é a representação gráfica do algoritmo de controle. O erro em malha fechada é calculado a partir da subtração de um *setpoint* pré-determinado com o valor real medido na planta pelo sistema de visão computacional. Ele então é enviado para o controlador, que determina a entrada a qual deve ser submetida a planta para contorná-lo (FRANCHI, 2011).

A principal malha utilizada para simulação pode ser observada na Figura 4.1. Ela é formada por duas malhas menores, visto que cada ângulo de entrada atua sobre uma das coordenadas ( $x, y$ ), como evidenciado na Tabela 2.2. Os componentes usados para realizar a simulação são detalhados a seguir.



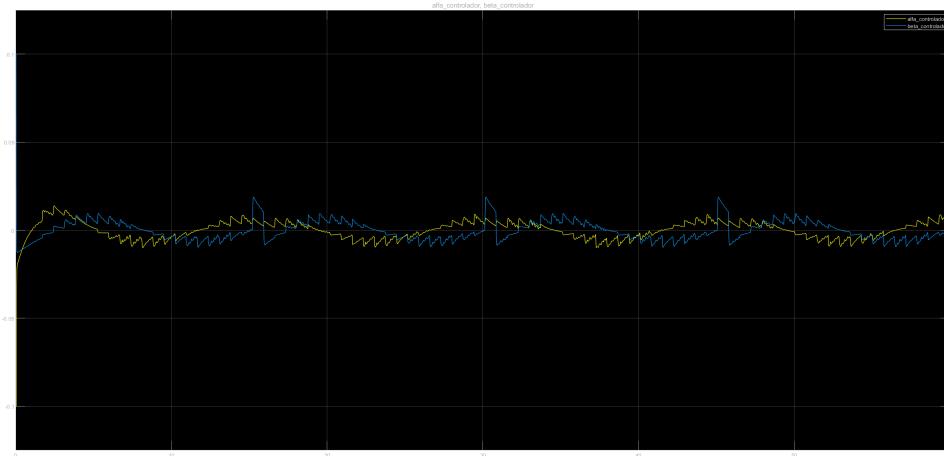
**Figura 4.1:** Malha de controle do processo.  
Fonte: Elaborado pelo autor.

## 4.2 Projeto de controlador

O principal objetivo deste trabalho é construir o processo *ball and plate* e determinar seu modelo. Por esse motivo, optou-se por obter um controlador PID simples, para validar a controlabilidade do modelo obtido e o cálculo da cinemática inversa. Utilizou-se portanto a ferramenta *PID Tuner* do *Matlab*, para obter um compensador que estabiliza o sistema, com  $K_p = -0.3$ ,  $K_i = -0.05$  e  $K_d = -0.45$ . Escolheu-se utilizar o controlador obtido pela ferramenta ao invés dos projetados pelo método de lugar das raízes pois estes últimos tiveram uma performance inferior. A função de transferência usada para obtenção do controlador pode ser observada na equação abaixo.

$$G(s) = \frac{-7.007}{s^2} \quad (4.1)$$

A simulação discutida foi realizada utilizando o modelo não linearizado do sistema, que será melhor detalhado adiante. Por esse motivo, já que o compensador foi projetado para o sistema linear, a ação de controle é submetida a uma saturação. Tal ação, usada para fazer a bolinha percorrer uma trajetória circular, pode ser observada na Figura 4.2.



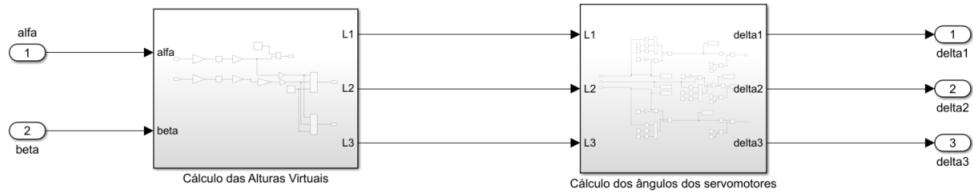
**Figura 4.2:** Ação de controle necessária para fazer com que a bolinha realize um percurso circular em torno da origem.

Fonte: Elaborado pelo autor.

## 4.3 Implementação da cinemática inversa

A maior diferença entre os diversos sistemas *ball and plate* está no manipulador paralelo que causa a inclinação da chapa. Como mencionado anteriormente, o manipulador utilizado no presente trabalho possui três graus de liberdade, e portanto faz-se

necessário obter uma expressão matemática que converta os ângulos  $\alpha$  e  $\beta$  obtidos pelo controlador para ângulos  $\Delta_i$ , que são as entradas de cada servomotor, gerando assim o *pitch* e *roll* desejado. O bloco de cinemática inversa é mostrado na Figura 4.3. Note que ele é composto por duas principais etapas: cálculo das alturas virtuais e o cálculo dos ângulos dos servomotores.



**Figura 4.3:** Blocos que realizam o cálculo dos ângulos  $\Delta_i$ .

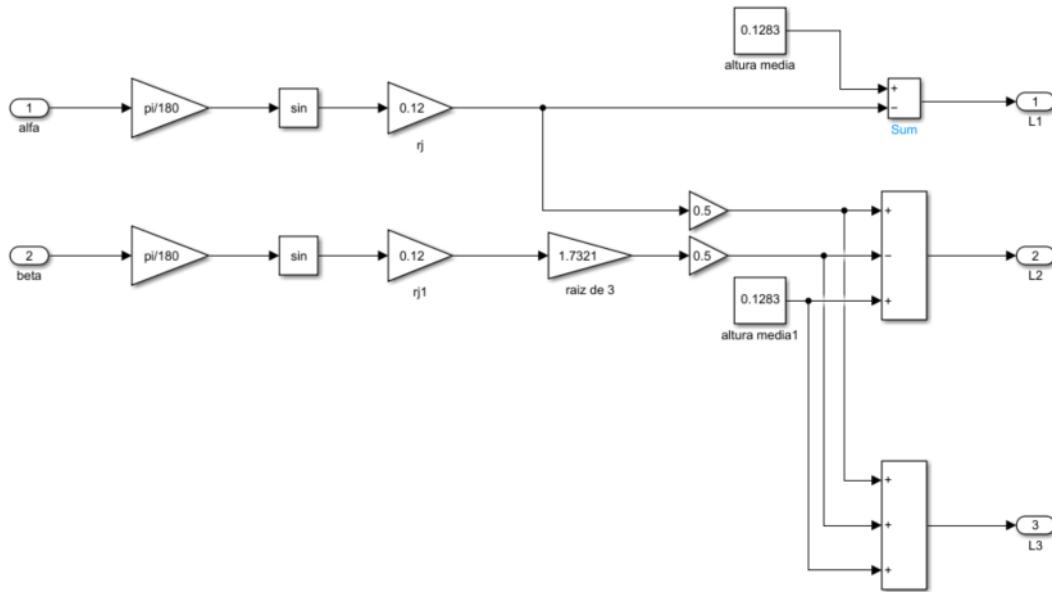
Fonte: Elaborado pelo autor.

Inicialmente, é necessário calcular as alturas virtuais  $L_i$  necessárias para causar a inclinação  $(\alpha, \beta)$ . Utilizou-se as equações (2.27) e (2.29) para montar um sistema de duas equações, com três variáveis de interesse. Isso representa um sistema com infinitas possíveis soluções, o que torna a simulação extremamente lenta devido à forma como o resultado desse sistema é obtido, o que possivelmente impossibilitaria a implementação na planta real. Para contornar o problema, foi necessário obter uma terceira equação, que tornasse o sistema próprio.

Uma boa solução seria uma equação que representasse o ângulo de *yaw* do sistema, uma vez que  $\alpha$  e  $\beta$  são os ângulos de *pitch* e *roll*. Porém, o ângulo de *yaw* é sempre zero devido à forma como as pernas foram posicionadas na montagem, impossibilitando assim obter uma relação entre as alturas virtuais e o ângulo em questão. Escolheu-se então uma altura média arbitrária menor que o comprimento total das pernas, para ser a altura fixa do ponto de origem do sistema coordenado da chapa. Com a terceira relação matemática obtida, o sistema de equações torna-se próprio e é lograda uma solução analítica para cada altura virtual, que pode ser vista na equação (4.2). O conjunto de blocos utilizado para realizar o cálculo do sistema é mostrado na Figura 4.4.

$$\begin{aligned}
 L_1 &= 0.1283 - r_j \operatorname{sen} \alpha, \\
 L_2 &= \frac{r_j \operatorname{sen} \alpha - \sqrt{3} r_j \operatorname{sen} \beta}{2} + 0.1283, \\
 L_3 &= \frac{r_j \operatorname{sen} \alpha + \sqrt{3} r_j \operatorname{sen} \beta}{2} + 0.1283.
 \end{aligned}$$

(4.2)

**Figura 4.4:** Cálculo das alturas virtuais.

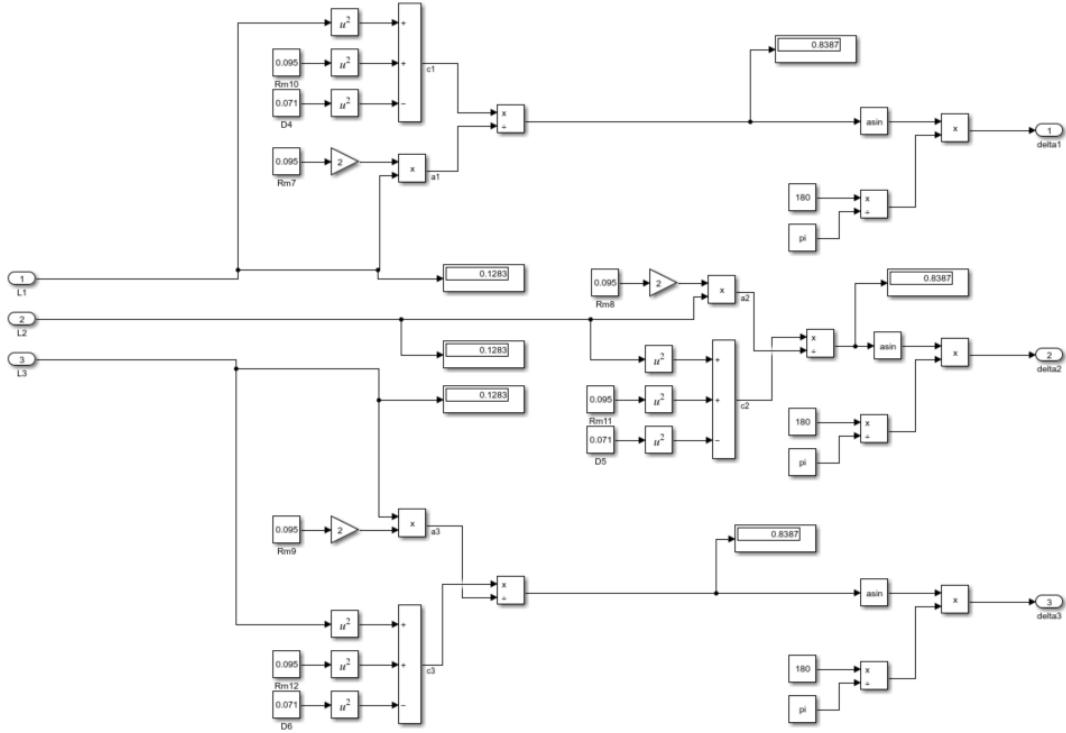
Fonte: Elaborado pelo autor.

Uma vez determinadas, utilizam-se as alturas para calcular os ângulos  $\Delta_i$  a partir da equação (2.27), obtendo primeiro  $a_i$  e  $c_i$  para cada perna. Como  $r_i$  e  $r_j$  são sempre iguais, todos os  $b_i$  sempre resultarão em zero, o que simplifica o cálculo e o diagrama usado para simulação, mostrado na Figura 4.5. É importante ressaltar que escolheu-se utilizar diagramas de blocos ao invés de *scripts* onde fosse possível, objetivando aumentar a velocidade de simulação.

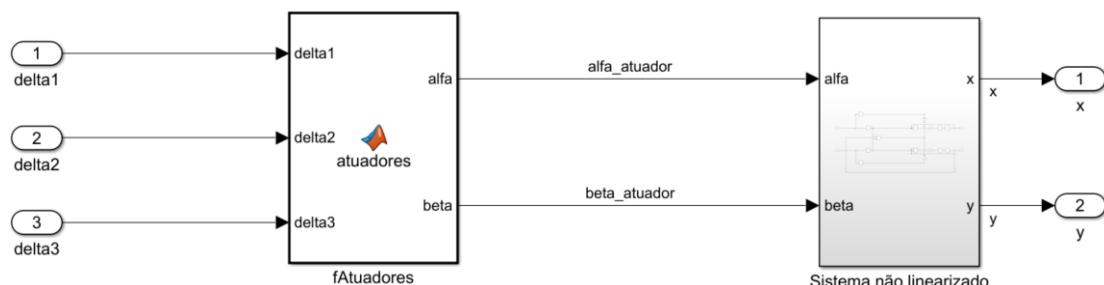
## 4.4 Processo

O bloco de processo pode ser visto na Figura 4.6. Ele é composto por dois outros blocos: um que representa a dinâmica dos atuadores, e outro que é uma representação das equações mostradas em (2.8). O primeiro difere dos outros blocos por ser uma função do *Matlab*, e isso ocorre pois é necessário resolver uma equação do segundo grau obtida a partir de (2.24), mostrada em (4.3), onde  $L_i$  são as variáveis de interesse. Como sempre existem duas soluções possíveis para essa equação, na maioria dos casos é escolhida a que resulta na menor altura. Caso ela seja negativa, a outra raiz é usada. Uma vez obtidas as alturas, os ângulos  $\alpha$  e  $\beta$  são computados a partir de (2.29) e (2.30).

$$L_i = R_m \operatorname{sen} \Delta_i \pm \sqrt{D^2 - \cos \Delta_i^2 R_m^2} \quad (4.3)$$

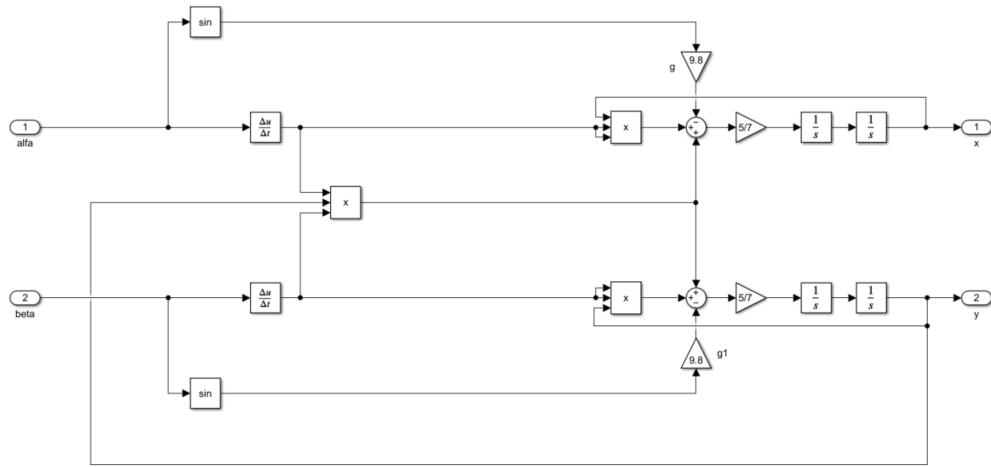


**Figura 4.5:** Cálculo dos ângulos  $\Delta_i$  a partir de  $L_i$ .  
Fonte: Elaborado pelo autor.



**Figura 4.6:** Bloco do processo, composto pelos atuadores e o sistema não linearizado.  
Fonte: Elaborado pelo autor.

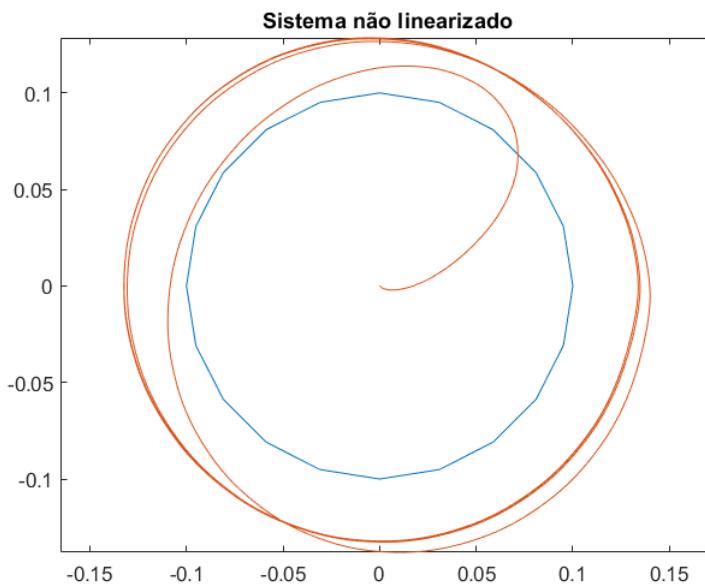
O segundo bloco é o que representa o modelo não linear do sistema, sem as simplificações discutidas no Capítulo 2. É importante realizar a simulação com ele mesmo que o controlador tenha sido obtido a partir do sistema não linearizado, uma vez que ele é uma aproximação mais fidedigna da realidade. Ele pode ser visto na Figura 4.7, e com sua análise é possível justificar o uso das saturações após o controlador, já que é necessário derivar a entrada e *spikes* da ação de controle resultam em impulsos que instabilizariam a planta.



**Figura 4.7:** Representação em blocos do modelo não linearizado.  
Fonte: Elaborado pelo autor.

## 4.5 Resultados

Para testar o sistema, foi definida uma trajetória circular de referência com a bolinha iniciando a partir da origem. A partir da análise da Figura 4.8, conclui-se que o compensador teve um comportamento satisfatório. Como pode ser visto, o sistema é controlável e com a elaboração de um controlador mais eficiente seria possível atingir tanto o erro zero quanto a trajetória circular. O resultado obtido é considerado aceitável, porém é importante ressaltar que um PID simples não é a estratégia para esse tipo de sistema, mesmo conseguindo estabilizá-lo.



**Figura 4.8:** Trajetória circular com centro na origem e raio de 10 cm.  
Fonte: Elaborado pelo autor.

## 4.6 Resumo do Capítulo

Neste capítulo foi apresentada uma visão detalhada da malha de controle utilizada na simulação, detalhando cada bloco que a compõe. O capítulo começa descrevendo a estratégia de controle utilizada, expondo em seguida a construção do bloco de cinemática inversa, que é o que varia entre os diferentes tipos de sistemas *ball and plate*. Após isso, foi demonstrado como a atuação foi simulada junto ao modelo não linear com diagrama de blocos, justificando assim o uso da saturação da ação de controle. O resultado do correto seguimento de uma trajetória circular foi obtido ao final, demonstrando que o sistema é controlável apesar de sua não linearidade e alta instabilidade.



# Capítulo 5

## Resultados

Para a execução do projeto foi necessário: montar a planta, utilizar um sistema de visão computacional, obter o modelo matemático e simular seu funcionamento utilizando software, além de ter sido necessário familiarizar-se com o funcionamento físico. Também foi necessária a leitura de diversos textos, e a elaboração de documento descrevendo todo esse processo.

### 5.1 Atividades do Projeto

O projeto teve como principais atividades:

- a) Construir uma planta *ball and plate*, com manipulador paralelo de três graus de liberdade;
- b) Realizar uma pesquisa bibliográfica sobre os principais conhecimentos utilizados no desenvolvimento do projeto;
- c) Utilizar um sistema de visão computacional que fosse capaz de retornar a posição da bola para o algoritmo de controle;
- d) Realizar a identificação do modelo matemático do processo;
- e) Demonstrar a controlabilidade do sistema por meio de simulação;
- f) Realização de testes na planta física;
- g) Redação de uma monografia relatando o desenvolvimento do projeto e um manual de uso da planta física, para facilitar e acelerar a progressão do estudo do sistema.

## 5.2 Requisitos do Sistema

Os principais requisitos do sistema são:

- a) O sistema deverá ser de baixo custo e simples de reproduzir;
- b) Deve ser possível utilizar o sistema de visão computacional em ambientes diversos, com a câmera posicionada em alturas e posições diferentes, já que a planta deve ser portátil.

## 5.3 Testes

Durante os meses de outubro e novembro de 2022, foram realizados diversos testes na planta física com os algoritmos e controladores desenvolvidos. A ideia inicial do projeto era utilizar apenas um Arduíno para realizar a alimentação e controle dos servomotores. O seu uso sozinho sozinho foi logo descartado, visto que percebeu-se que não seria possível utilizá-lo em conjunto com a câmera já que os dois atuam como escravos.

Seguiu-se então com o uso da placa *Raspberry Pi*, que é um computador pequeno e portátil com uma distribuição *Linux* como sistema operacional. Além de realizar as mesmas funções que um computador comum, a placa tem pinos de saída que podem ser utilizados para enviar os sinais de controle via *protoboard* para os servomotores. Com ela, foi possível ler a posição da bolinha em tempo real e enviar sinais de controle para o processo pela primeira vez. Também passou-se a usar uma fonte localizada no laboratório (LSI) para alimentar os três motores, já que o *Raspberry* apenas consegue alimentar um servo de cada vez.

Outra premissa do início do projeto era utilizar uma lamparina para servir de apoio para a câmera, objetivando reduzir o custo do projeto, que posicionaria a lente acima da chapa paralela a ela. No entanto, nos primeiros testes, percebeu-se que era muito difícil estabilizar a posição da lente e a lamparina não proporcionava diferentes alturas de forma simples e prática para testar a flexibilidade do algoritmo. Por esse motivo e pela urgência, foi feita a compra de um pedestal de microfone flexível com rotação de 360º, da marca *SYANG*, que atende aos requisitos melhor do que a lamparina e é relativamente barato, custando em torno de 10 dólares na data da compra.

No entanto, notou-se que havia um *twitching* nos servos quando conectados aos pinos PWM da placa, o que impossibilitava um estado de equilíbrio. O sinal PWM enviado foi analisado utilizando um osciloscópio e percebeu-se que, quando havia mais de uma *thread* sendo executada pelo processador do *Raspberry*, os pulsos enviados não

tinham período fixo. Isso provavelmente ocorre pois tal sinal é emulado por *software* na placa, diferente do Arduíno.

Na montagem final, utilizou-se as duas placas anteriormente citadas em conjunto. O *Raspberry* atua como mestre, recebendo a posição da bolinha gerada pela *thread* de visão computacional, com tempo de amostragem de 1 milissegundo. Tal dado é enviado em uma comunicação *Python* - Arduíno na *thread main* do sistema, que mantém os ângulos anteriores enquanto "escuta" sua porta USB até receber o sinal de controle enviado pelo *Raspberry*. Com tal esquema, foi possível ler corretamente a posição da bolinha e alterar os ângulos da chapa, causando deslocamento da bola sobre ela.

## 5.4 Análise dos Resultados

Nesta seção será feita uma análise dos quatro principais objetivos do projeto, além de relatar a aplicação da estratégia de controle na planta física, que está sujeita aos distúrbios e diferenças do modelo aproximado obtido.

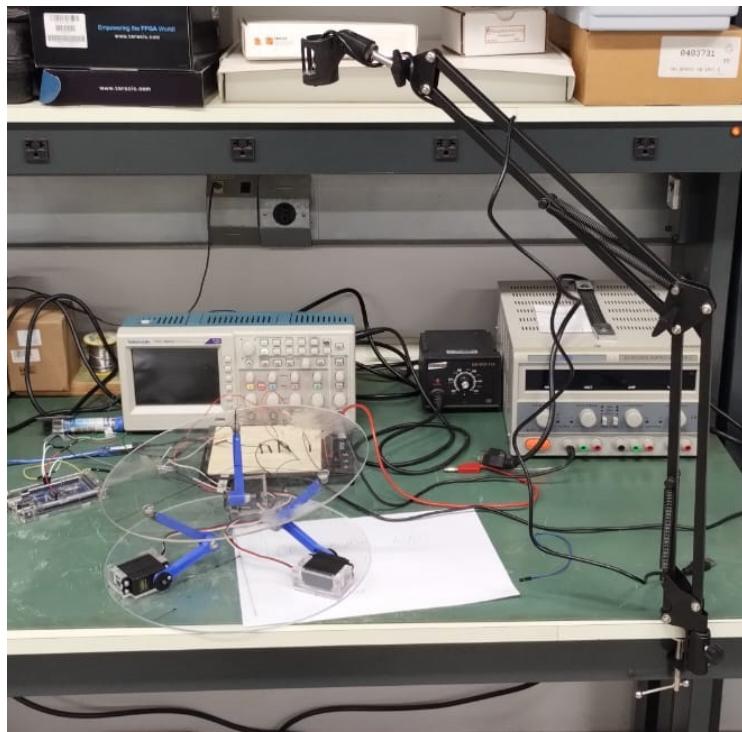
### 5.4.1 Montagem da planta

A montagem da planta estava sendo planejada desde o ano de 2019. Os professores responsáveis pelo projeto já possuíam uma planta que representa um problema bastante similar em duas dimensões, o *ball and beam*, e desejava-se obter o sistema do caso 3D aqui discutido. Devido à pandemia de COVID-19, o projeto de desenvolvimento da planta foi adiado até o ano de 2022, que é quando este trabalho está sendo escrito.

Como pode ser visto na Figura 5.1, a planta foi montada com sucesso no LSI. O esquema de montagem foi inicialmente pensado com as pernas do manipulador paralelo posicionadas em fila circular, no entanto esse tipo de montagem causava um *yaw* altíssimo no sistema. Após perceber este comportamento indesejado, elas foram posicionadas com o vetor que parte de  $B_i$  até a junta que conecta os dois segmentos da perna voltados para o centro. Dessa forma, o *yaw* no caso ideal é de zero, porém, devido à imperfeições na fabricação dos servomotores, é causado um leve *yaw* da chapa que adiciona distúrbios não modelados no sistema.

### 5.4.2 Sistema de visão computacional

Como pode ser visto na Figura 5.2, para a bolinha posicionada em cada um dos quatro quadrantes da chapa, é lido um valor de  $X$  e  $Y$  com sinal condizente com o local em que ela está colocada sobre. Além disso, é feita uma normalização entre a quantidade de *pixels* dentro do círculo de captura e o tamanho real da chapa utilizada,



**Figura 5.1:** Fotografia da planta montada no LSI.  
Fonte: Repositório de imagens do autor.

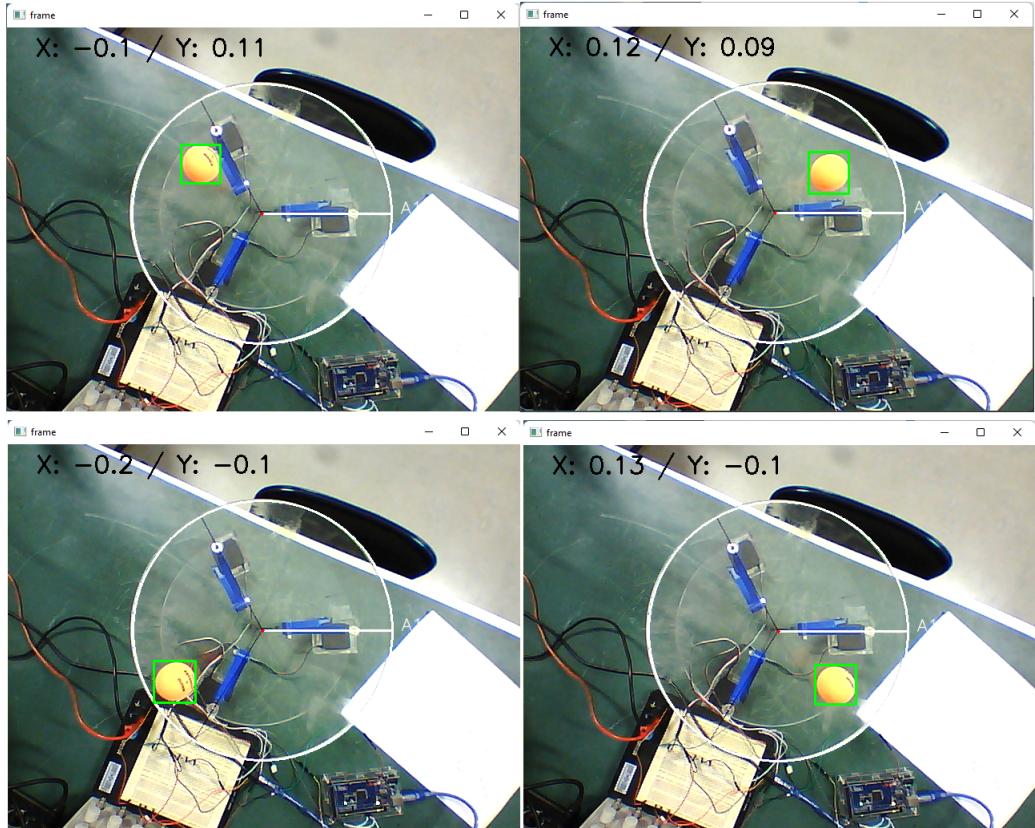
o que faz com que o valor das duas coordenadas esteja sempre entre 0 e 30 centímetros. A captura da posição da bolinha foi considerada como satisfatória.

### 5.4.3 Obtenção do modelo e controlabilidade

A dinâmica de uma planta *ball and plate* genérica é geralmente a mesma, desde que as esferas utilizadas sejam do mesmo tipo. O processo sempre poderá ser interpretado como dois sistemas não lineares, onde os ângulos de *pitch* e *roll* irão influenciar sobre a posição da bolinha. No caso mais simplificado (sistema linearizado), usado para elaboração do controlador, pode-se até considerar que cada ângulo atua sobre apenas uma coordenada.

A grande diferença entre os diversos sistemas *ball and plate* é o manipulador paralelo utilizado para causar a inclinação da chapa. No presente trabalho, foi utilizado um manipulador de três graus de liberdade, que é mais flexível que o mais tradicional com dois graus, e menos complexo que a também tradicional plataforma de *Stewart* de seis graus de liberdade. As equações (2.27) a (2.30) descrevem a dinâmica necessária para aplicação do controle, obtidas pelo estudo da cinemática inversa do manipulador utilizado.

Na Tabela 5.1, estão relacionadas as posições mostradas na Figura 5.2 com os respectivos ângulos  $\alpha$  e  $\beta$  por estas posições gerados pelo controlador e pela cinemática



**Figura 5.2:** Posição da bolinha lida em cada um dos quatro quadrantes da chapa.  
Fonte: Repositório de imagens do autor.

inversa, quando a malha de controle recebe como *setpoint* o ponto  $(0, 0)$ . Como pode ser observado, tais ângulos quando aplicados à planta, causariam a continuidade da trajetória circular em torno da origem.

**Tabela 5.1:** Ângulos  $\alpha$  e  $\beta$  necessários para mover a bolinha para a origem, resultado da simulação, quando submetido a uma trajetória circular.

| Posição atual            | Ângulos aplicados à chapa                   | Tempo      |
|--------------------------|---|------------|
| $x = 0.101, y = 0.008$   | $\alpha = 0.46^\circ, \beta = 0.29^\circ$   | $t = 2.5$  |
| $x = -0.097, y = 0.083$  | $\alpha = -0.34^\circ, \beta = 0.34^\circ$  | $t = 5.0$  |
| $x = -0.083, y = -0.106$ | $\alpha = -0.29^\circ, \beta = -0.29^\circ$ | $t = 7.5$  |
| $x = 0.112, y = -0.07$   | $\alpha = 0.51^\circ, \beta = -0.29^\circ$  | $t = 10.0$ |

#### 5.4.4 Aplicação do controlador na planta física

A estratégia de controle PID mostrou-se insuficiente para controlar a posição da bolinha na planta real. Apesar de ter funcionado tanto no modelo linear quanto no não linear nas simulações realizadas, os seguintes distúrbios provavelmente impediram o funcionamento correto do controle:

- a)  $Yaw$  não modelado causado pelas imperfeições dos servomotores usados;
- b) Apesar de terem sido reduzidos com o uso do Arduíno, ainda ocorrem eventuais *twitchings* no movimento dos servos, que modificam as alturas  $L_i$ .

Todos itens citados acima adicionam mais características não lineares ao sistema, e o controlador do tipo PID não é o mais adequado para lidar com sistemas desse tipo. Além disso, a principal desvantagem desta estratégia de controle é a sensibilidade à incertezas (KHARE; SINGH, 2010), e isso fica claro quando a planta é sujeita à distúrbios bruscos durante a execução do algoritmo de controle, como empurrar a bolinha quando ela está sobre a posição de equilíbrio. Essa mudança repentina faz com que a ação de controle "exploda" e a posição da bolinha tenda a infinito, devido também às características não lineares, e isso já podia ser observado na simulação do sistema na simulação do *Simulink* ao aplicar-se um pequeno impulso na saída.

## 5.5 Resumo do Capítulo

Neste capítulo, foram discutidos os resultados obtidos para cada um dos principais objetivos do projeto. Os três maiores objetivos, de montagem, programação do algoritmo de visão computacional e de modelagem do processo apresentaram resultado satisfatório: a planta é portátil e adaptável à diferentes ambientes, o uso da câmera é flexível e o modelo demonstrou-se controlável por meio de simulação. Devido à aplicação de técnicas de controle não tão robustas e adaptáveis em um sistema altamente não linear como o aqui estudado, o uso de um controlador PID simples demonstrou-se insuficiente. No capítulo 6, serão discutidas possíveis alternativas de controle para futuros trabalhos.

# Capítulo 6

## Conclusões

Neste capítulo, são discutidas as conclusões obtidas com a pesquisa realizada, com a construção e estudo do processo *ball and plate* com manipulador paralelo de três graus de liberdade.

### 6.1 Considerações Finais

Este trabalho visou construir e modelar uma planta *ball and plate* para ser utilizada por alunos da Universidade Federal de Minas Gerais como objeto de estudo das diversas disciplinas de Engenharia de Controle. A tarefa foi realizada no Laboratório de Sistemas Inteligentes, utilizando recursos de baixo custo, conhecimentos de visão computacional e de identificação de sistemas.

Foi necessário também realizar um estudo aprofundado do funcionamento de manipuladores paralelos com três graus de liberdade, para que fosse possível realizar o mapeamento entre os ângulos de entrada do processo e os ângulos de *pitch* e *roll* da chapa. Os trabalhos que tratam de manipuladores para esse tipo de aplicação são mais escassos que os de dois ou seis graus de liberdade, o que agrega valor à pesquisa realizada.

O processo de montagem foi um sucesso e está bem documentado neste trabalho, permitindo assim sua replicação caso haja necessidade. O algoritmo de visão computacional consegue manter o *tracking* da bola com precisão, dentro do raio de busca definido pelo usuário na interface de dimensões desse. Por fim, foi possível provar a controlabilidade do sistema por meio de simulação, tanto para o modelo linear quanto para o não linear, atingindo assim os principais requisitos e objetivos estabelecidos.

Havia expectativa, no início do projeto, de controlar a planta física utilizando um PID simples, desenvolvido a partir do modelo identificado. Apesar dessa estratégia ter funcionado para ambos os casos linear e não linear, o processo físico está submetido à

diversas incertezas, como distúrbios e não linearidades que não foram modeladas, que tal compensador não se mostrou suficiente para equilibrar a bolinha. Na seção seguinte, serão discutidas estratégias de controle mais adequadas para esse tipo de planta.

## 6.2 Propostas de Continuidade

É de interesse, para projetos futuros, determinar qual método de controle é mais eficiente para o processo *ball and plate*. Por ser não linear e altamente instável, ele está sujeito à incertezas que técnicas mais clássicas de controle, como o PID simples, não são suficientes para resolver o problema. Técnicas de controle adaptativo e robusto estavam presentes em alguns dos trabalhos aqui citados, e podem ser utilizados como referências para realizar essa comparação.

Uma possível adaptação do PID simples seria possível, utilizando um algoritmo que atualiza os ganhos do sistema em tempo real de acordo com o seu comportamento. Dessa forma, o compensador pode se comportar de diferentes formas, com diversas velocidades e intensidades, quando submetido aos variados distúrbios causados.

Em Coelho e Mariani (2006), por exemplo, técnicas de *soft computing* são aplicadas desde a modelagem até o controle da planta. Uma rede neural é usada para sintonizar os ganhos do controlador durante a execução do algoritmo, o que permite a adaptação e superação das incertezas às quais o processo está submetido.

Já em Bang e Lee (2018), é usado um método de controle chamado *sliding mode control* (SMC), que é próprio para sistemas não lineares. Tal abordagem altera a dinâmica do sistema ao aplicar um sinal de controle com descontinuidades, que permite que o sistema comporte-se de forma diferente da atuação normal. É importante ressaltar que para essa estratégia de controle específica, pode ser que haja *chattering* que leva a chaveamentos abruptos quando tende à superfície de deslizamento, que pode queimar os servomotores.

Uma estratégia de controle *fuzzy* pode vir a ser interessante e intuitiva para este tipo de planta: um controlador feito na "força bruta" teria um comportamento muito similar a um *fuzzy*. Uma vez que o sistema de visão determina a presença da bolinha em uma seção pré-definida da área de captura, o sinal de controle que será enviado para cada servo terá um peso que varia de acordo com a distância da esfera das outras seções.

É interessante estudar também a possibilidade de troca dos imãs de neodímio colados à plataforma móvel. Isso permitiria um "deslize" mais amplo do segundo segmento das pernas da plataforma móvel, permitindo assim submeter a chapa à inclinações mais amplas.

Uma abordagem de modelagem *grey box* pode vir a ser interessante para a planta. Como o modelo matemático obtido para a cinemática inversa não leva em conta alguns dos distúrbios que foram discutidos anteriormente, é importante tentar obter um modelo para esse bloco específico. Uma possibilidade seria medir os ângulos de *pitch* e *roll* da chapa, ou alternativamente as alturas virtuais das pernas, causados a partir da aplicação de sinais do tipo PRBS em cada servomotor.

### 6.2.1 Conclusão

Por meio do desenvolvimento deste projeto, foi possível aplicar um rol diverso de conhecimento adquiridos ao longo do curso para a solução de um complexo problema de engenharia. O acompanhamento próximo do professor orientador responsável foi fundamental para o andamento da pesquisa, construção e escrita, além do constante compartilhamento de conhecimento.

Por fim, entende-se que o trabalho engloba diversos aspectos conceituais estudados no curso de Engenharia de Controle e Automação, como: visão computacional, programação concorrente, identificação de sistemas, controle clássico e digital, sistemas dinâmicos lineares e não lineares, eletrônica, entre outros. A aplicação destes diversos conceitos foram fundamentais para a dimensão do projeto, e com certeza contribuirá para a formação de outros engenheiros.



# Referências Bibliográficas

- ALI, H. I.; JASSIM, H. M.; HASAN, A. F. Optimal nonlinear model reference controller design for ball and plate system. *Arabian Journal for Science and Engineering*, Springer, v. 44, n. 8, p. 6757–6768, 2019.
- BANG, H.; LEE, Y. S. Implementation of a ball and plate control system using sliding mode control. *IEEE Access*, IEEE, v. 6, p. 32401–32408, 2018.
- COELHO, L. d. S.; MARIANI, V. C. Sistema híbrido neuro-evolutivo aplicado ao controle de um processo multivariável. *SBA: Controle & Automação Sociedade Brasileira de Automatica*, SciELO Brasil, v. 17, n. 1, p. 32–48, 2006.
- FRANCHI, C. M. Controle de processos industriais. *São Paulo: Érica*, 2011.
- GUAN, J.; LIN, C.-M.; JI, G.-L.; QIAN, L.-W.; ZHENG, Y.-M. Robust adaptive tracking control for manipulators based on a TSK fuzzy cerebellar model articulation controller. *IEEE Access*, IEEE, v. 6, p. 1670–1679, 2017.
- INSTRUCTABLES. 2019. **Ball Balancing PID System**. Disponível em: <<https://www.instructables.com/id/Ball-Balancing-PID-System/>>. Acesso em 29 de ago. 2019.
- KASSEM, A.; HADDAD, H.; ALBITAR, C. Comparison between different methods of control of ball and plate system with 6DOF Stewart platform. *IFAC-PapersOnLine*, Elsevier, v. 48, n. 11, p. 47–52, 2015.
- KHARE, Y. B.; SINGH, Y. Pid control of heat exchanger system. *International Journal of Computer Applications*, International Journal of Computer Applications, 244 5 th Avenue, # 1526, New ..., v. 8, n. 6, p. 22–27, 2010.
- MORENO-ARMENDARIZ, M. A.; PÉREZ-OLVERA, C. A.; RODRÍGUEZ, F. O.; RUBIO, E. Indirect hierarchical FCMAC control for the ball and plate system. *Neurocomputing*, Elsevier, v. 73, n. 13-15, p. 2454–2463, 2010.
- NÚÑEZ, D.; ACOSTA, G.; JIMÉNEZ, J. Control of a ball-and-plate system using a state-feedback controller. *Ingeniare: Revista Chilena de Ingeniería*, Universidad de Tarapacá, v. 28, n. 1, p. 6–15, 2020.
- OKAFOR, E.; UDEKWE, D.; IBRAHIM, Y.; MU’AZU, M. B.; OKAFOR, E. G. Heuristic and deep reinforcement learning-based PID control of trajectory tracking in a ball-and-plate system. *Journal of Information and Telecommunication*, Taylor & Francis, v. 5, n. 2, p. 179–196, 2021.
- SOLER, C. J. Table tennis ball tracking and bounce calculation using opencv. 2017.

SZUFNAROWSKI, F. Stewart platform with fixed rotary actuators: a low cost design study. *Advances in medical Robotics*, n. 4, 2013.

USINAINFO. 2022. Servo Motor tipo Futaba S3003 180º 4.2Kg/cm de Posição. **USI-NAINFO Eletrônica e Robótica**. Disponível em: <<https://www.usinainfo.com.br/servo-motores/servo-motor-tipo-futaba-s3003-180-42kgcm-de-posicao-2577.html>>. Acesso em 18 de dez. de 2022.

ZHAO, Y.; QIU, K.; WANG, S.; ZHANG, Z. Inverse kinematics and rigid-body dynamics for a three rotational degrees of freedom parallel manipulator. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 31, p. 40–50, 2015.

# Apêndice A

## Manual de Uso

Este capítulo tem como objetivo mostrar de forma prática o funcionamento da planta, e como que os algoritmos que a fazem funcionar foram estruturados.

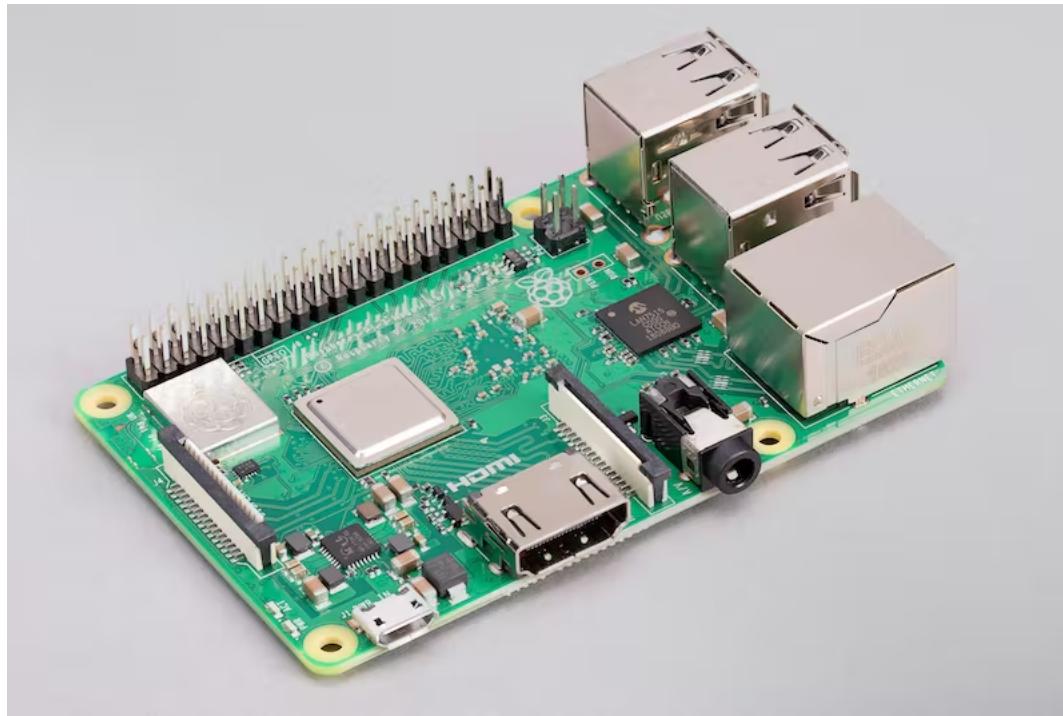
### A.1 Montando a Planta

#### A.1.1 Conectando o *Raspberry Pi* e o Arduíno

A montagem do Arduíno com os servomotores está explicitada na Figura A.2. Ele será usado apenas para enviar os sinais de controle para os servomotores, recebidos via conexão USB do *Raspberry Pi*, e deve ser aterrado no mesmo ponto que a fonte e os servos. Nesse último, deverá estar sendo executado um código em *Python*, que deverá executar duas *threads*: uma que faz os cálculos dos sinais de controle e outra que captura a imagem, cuja câmera que produz as imagens está conectada também via USB com o *Raspberry Pi*.

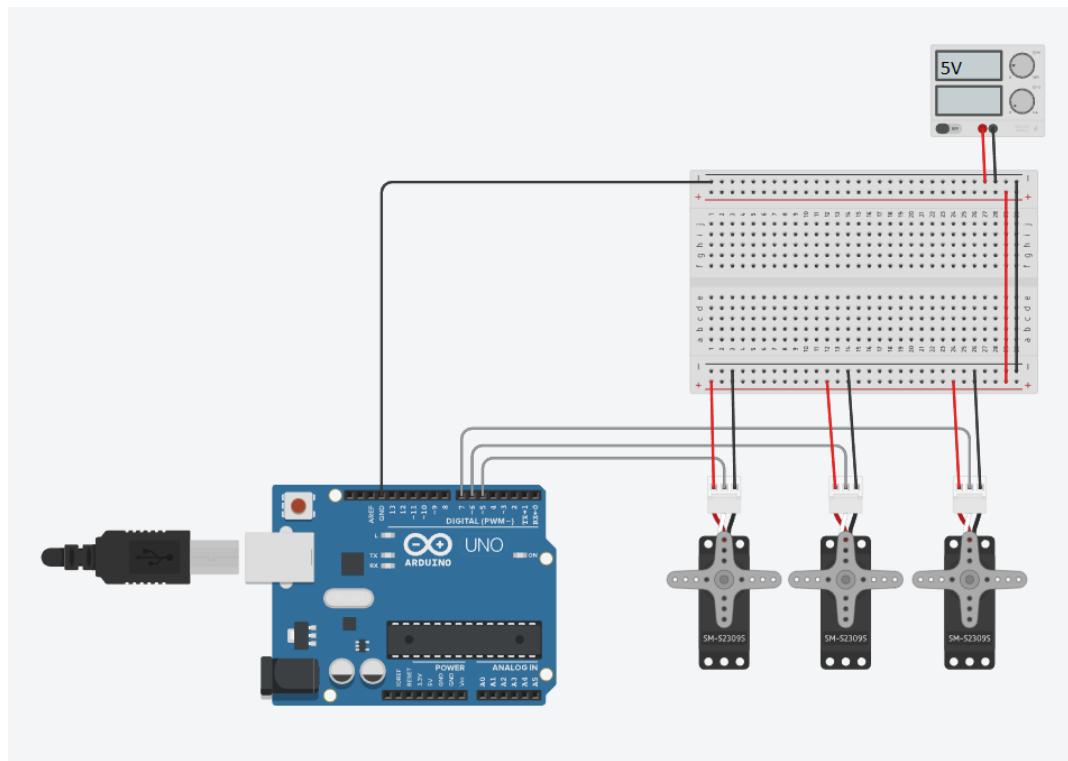
#### A.1.2 Posicionando a câmera

A câmera deverá ser posicionada acima da plataforma, com a lente aproximadamente paralela à chapa, como o exemplo na Figura A.3. O círculo mostrado na *thread* de captura de imagem deverá ter o raio e as coordenadas do seu centro alteradas de acordo com a posição real da chapa, de forma que os contornos coincidam. Além disso, é necessário alterar os valores HSV de acordo com a janela de máscara, de forma que apenas a bolinha seja detectada quando estiver dentro do círculo de captura.



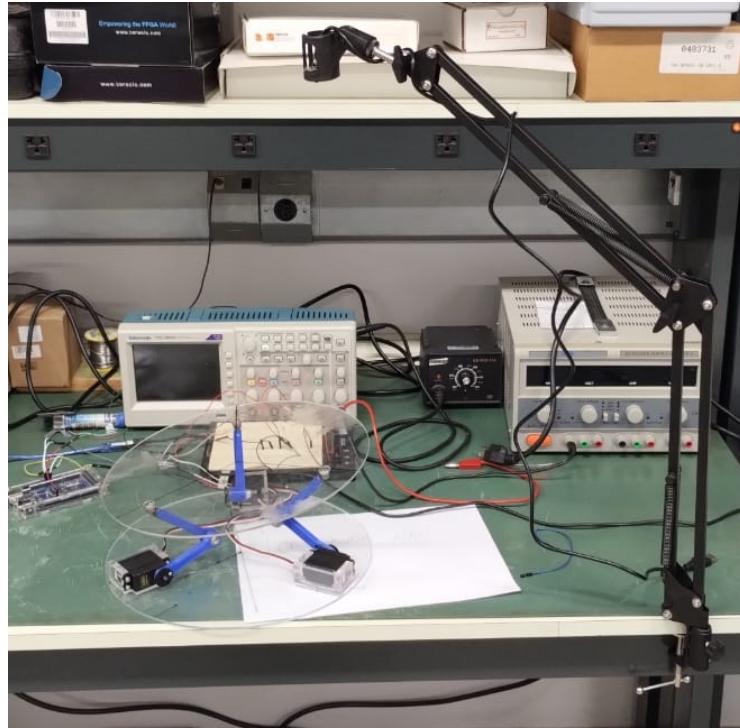
**Figura A.1:** *Raspberry Pi 3 B+*

Fonte: <<https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/>>. Acesso em: 18 dez. 2022.



**Figura A.2:** Como conectar o Arduíno aos servomotores.

Fonte: Elaborado pelo autor.



**Figura A.3:** Exemplo de posicionamento da câmera.  
Fonte: Repositório de imagens do autor.

## A.2 Detalhamento do Algoritmo de Visão Computacional

No trecho mostrado abaixo, é possível ver a inicialização do algoritmo de visão computacional usado. A função recebe como parâmetros as posições  $(x, y)$ , definidas como variáveis do tipo *Value* globais da biblioteca *multiprocessing*, já que são acessadas por *threads* diferentes. No *snippet* da Figura citada, é escolhida qual câmera deverá ser usada para captura, o *frame* principal é definido e as janelas de ajuste HSV e da posição e raio do círculo de captura são definidos.

```

1 #Usa a camera como sensor para fazer o tracking da posicao da bolinha
2 def captura_imagem(x_pos, y_pos):
3     cap = cv2.VideoCapture(0)
4     ret, frame = cap.read()
5
6     h, w, _ = frame.shape
7
8     cv2.namedWindow('Dimensoes da area de captura')
9
10    #Usar um dos conjuntos para X, Y, Raio abaixo. O segundo serve para
11    #definir valores padroes de inicializacao, e o primeiro inicia no meio
12    #do slider.

```

```

11
12     #cv2.createTrackbar('X', 'Dimensoes da area de captura', int(w/2), w,
13     nothing)
14     #cv2.createTrackbar('Y', 'Dimensoes da area de captura', int(h/2), h,
15     nothing)
16     #cv2.createTrackbar('Raio', 'Dimensoes da area de captura', int(w/4),
17     int(w/2), nothing)
18
19     cv2.createTrackbar('X', 'Dimensoes da area de captura', 318, w,
20     nothing)
21     cv2.createTrackbar('Y', 'Dimensoes da area de captura', 232, h,
22     nothing)
23     cv2.createTrackbar('Raio', 'Dimensoes da area de captura', 162, int(w
24     /2), nothing)
25
26     cv2.namedWindow('Ajuste deteccao bola')
27     cv2.createTrackbar('LH', 'Ajuste deteccao bola', 26, 255, nothing)
28     cv2.createTrackbar('LS', 'Ajuste deteccao bola', 94, 255, nothing)
29     cv2.createTrackbar('LV', 'Ajuste deteccao bola', 255, 255, nothing)
30     cv2.createTrackbar('UH', 'Ajuste deteccao bola', 255, 255, nothing)
31     cv2.createTrackbar('US', 'Ajuste deteccao bola', 255, 255, nothing)
32     cv2.createTrackbar('UV', 'Ajuste deteccao bola', 255, 255, nothing)

```

O algoritmo entra em um *loop* que funciona enquanto a câmera estiver aberta e funcionando, e as coordenadas e o raio atual do círculo de captura são salvas em três variáveis. O círculo é então desenhado com base nessas variáveis (ajustadas pelos *sliders*), e a *mask1*, que é a máscara que garante que apenas o que está dentro do círculo desenhado é escrito nas variáveis de posição globais, é definida. Em seguida, as variáveis responsáveis pelos limites inferior e superior dos valores HSV são escolhidas e agora podem ser ajustadas para detectar a cor desejada, e o trecho de código descrito pode ser visto no trecho a seguir.

```

1  while(cap.isOpened()):
2      ret, frame = cap.read()
3
4      if ret == True:
5
6          x = cv2.getTrackbarPos('X','Dimensoes da area de captura')
7          y = cv2.getTrackbarPos('Y','Dimensoes da area de captura')
8          r = cv2.getTrackbarPos('Raio','Dimensoes da area de captura')
9
10         cv2.circle(frame, (x,y), r if r >= 10 else 10, (255,255,255),
11                     2)
12         cv2.line(frame, (x,y), (x+r, y), (255, 255, 255),2)

```

```

12         cv2.putText(frame, ' A1', (x+r, y), cv2.FONT_HERSHEY_SIMPLEX,
13             0.71, (255, 255, 255), 1, cv2.LINE_AA)
14
15         mask1 = cv2.inRange(cv2.cvtColor(frame, cv2.COLOR_BGR2HSV),
16             np.array([0,0,255]), np.array([0,0,255]))
17         cv2.circle(mask1, (x,y), r if r >= 10 else 10, (255,255,255),
18             -1)
19
20         hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
21
22         l_h = cv2.getTrackbarPos('LH', 'Ajuste detecção bola')
23         l_s = cv2.getTrackbarPos('LS', 'Ajuste detecção bola')
24         l_v = cv2.getTrackbarPos('LV', 'Ajuste detecção bola')
25
26         u_h = cv2.getTrackbarPos('UH', 'Ajuste detecção bola')
27         u_s = cv2.getTrackbarPos('US', 'Ajuste detecção bola')
28         u_v = cv2.getTrackbarPos('UV', 'Ajuste detecção bola')
29
30         l_b = np.array([l_h, l_s, l_v])
31         u_b = np.array([u_h, u_s, u_v])

```

Como pode ser visto no trecho de código abaixo, a máscara maior é então definida e o que é por ela detectado sofre dilatação, que pode ser alterada aumentando ou diminuindo o número do parâmetro *iterations*, que deverá ser sempre ímpar. Faz-se então um *AND* com *mask* e *mask1*, para que *mask2* (máscara principal), detecte somente o que está dentro do círculo de captura. Portanto, nesta última máscara obtida, são detectados todos os contornos das figuras em branco que aparecem em *mask2*.

```

1         mask = cv2.inRange(hsv, l_b, u_b)
2         mask = cv2.dilate(mask, None, iterations=3)
3
4         mask2 = cv2.bitwise_and(mask1, mask)
5
6         res = cv2.bitwise_and(frame, frame, mask=mask2)
7
8         cv2.circle(frame, (x,y), 0, (0,0,255), 4)
9
10        O_x = x
11        O_y = y
12
13        contours, _ = cv2.findContours(mask2, cv2.RETR_TREE, cv2.
14            CHAIN_APPROX_SIMPLE)

```

Os contornos detectados são iterados um a um, e a área de cada um é calculada.

Caso o contorno tenha uma área maior que 300 *pixels*, um retângulo é desenhado (neste caso, em torno da bolinha), e as coordenadas do centro deste retângulo são salvas nas variáveis globais de interesse. O *frame* principal é mostrado, e a máscara *mask2* também deve ser usada como referência de ajuste do HSV. Caso a tecla *q* seja pressionada, o algoritmo sai do *loop* e a *thread* é encerrada, e o código pode ser visto no trecho a seguir.

```

1      for contour in contours:
2          (x, y, w, h) = cv2.boundingRect(contour)
3
4          if cv2.contourArea(contour) < 300:
5              continue
6          else:
7              cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0),2)
8              saida_x = (int(x+w/2)-0_x)
9              saida_y = (-int(y+h/2)+0_y)
10             posicao_str = 'X: ' + str((saida_x/r)*0.3)[:4] + ' / '
11             Y: ' + str((saida_y/r)*0.3)[:4]
12             cv2.putText(frame, posicao_str, (35, 35), cv2.
13             FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
14             #print(posicao_str)
15             x_pos.value = (saida_x/r)*0.3                      #
16             normalizacao dos valores para uma escala de 0 a 30 cm
17             y_pos.value = (saida_y/r)*0.3
18
19
20             cv2.imshow('frame', frame)
21             #cv2.imshow('mask', mask)
22             #cv2.imshow('mask1', mask1)
23
24             # DESCOMENTAR A LINHA ABAIXO (MASK2 PARA AJUSTAR DETECCAO DE
25             COR. PADRAO: LARANJA.)
26             #cv2.imshow('mask2', mask2)
27             #cv2.imshow('res', res)
28
29             if cv2.waitKey(1) & 0xFF == ord('q'):
30                 break
31             else:
32                 break

```

## A.3 Comunicação *Raspberry* - Arduíno

Como foi discutido no texto da monografia, o uso de apenas o *Raspberry Pi* não foi o suficiente para realizar a execução das duas *threads* principais e ainda enviar os sinais de controle para os servomotores. Por esse motivo, o papel de controle foi transferido para um Arduíno, e é necessário realizar a conexão entre as duas placas para que haja o correto funcionamento. O *Raspberry* recebe  $K_d$ ,  $K_p$  e  $K_i$  na sua inicialização, como pode ser visto no trecho de código abaixo, antes das duas tarefas principais serem inicializadas.

```

1 if __name__ == '__main__':
2
3     controlador_x = PID(_kp, _ki, _kd) # Definindo parametros do
4         controlador, onde PID eh a funcao do algoritmo de controle.
5         controlador_y = PID(_kp, _ki, _kd)
6
7         controlador_x.send(None) # Inicializando o controlador
8         controlador_y.send(None)
9
10        threading.Timer(1, main_loop).start()
11        threading.Thread(target=captura_imagem(x_pos, y_pos)).start()
```

A comunicação é estabelecida no início do código como pode ser visto no trecho de código a seguir, e necessita que o Arduíno já esteja conectado via cabo USB ao *Raspberry*. Primeiramente, todas as portas seriais abertas são lidas, e salvas numa lista. O usuário então escolhe qual o número da porta em que o Arduíno está conectado, e essa numeração pode ser vista ao clicar no botão *Ferramentas* na IDE do Arduíno. A *baudrate* deverá ser a mesma estabelecida no *setup* do Arduíno, e a comunicação é estabelecida em seguida.

```

1 # Estabelecendo comunicacao com o arduino
2
3 ports = serial.tools.list_ports.comports()
4 serialInst = serial.Serial()
5 portsList=[]
6
7 for onePort in ports:
8     portsList.append(str(onePort))
9     print(str(onePort))
10
11 # Escolha da porta COM do arduino
12 val = 5
13
14 for x in range(0, len(portsList)):
```

```

15     if portsList[x].startswith('COM' + str(val)):
16         portVar = 'COM' + str(val)
17         print(portVar)
18
19 serialInst.baudrate = 9600
20 serialInst.port = portVar
21 serialInst.open()

```

No *loop* principal de execução, mostrado no próximo trecho de código, o tempo atual é obtido e as variáveis de processo e manipuladas são calculadas. A primeira corresponde à posição lida naquele instante de tempo pela visão computacional, enquanto a segunda representa o sinal de controle saturado calculado naquele instante. A função *gera\_msg* recebe o valor calculado pela cálculo da cinemática inversa e adequa-o no formato que deverá ser lido pelo Arduíno, montando e enviando a mensagem em seguida. Por último, o *loop* espera o tempo de amostragem pré-definido para realizar uma nova medição.

```

1 def main_loop():
2
3     while(True):
4         t = time.time()
5         pv_x = x_pos.value
6         pv_y = y_pos.value
7
8         mv_x = satura_sinal(controlador_x.send([t, pv_x, sp_x]),
9                               saturacao, -saturacao)
10        mv_y = satura_sinal(controlador_y.send([t, pv_y, sp_y]),
11                               saturacao, -saturacao)
12
13        delta = cinematica_inversa(mv_x, mv_y)
14
15        # Enviando angulos para o arduino
16        command = gera_msg(delta)
17        command = command + ('\n')
18        #print(command)
19        print("X: " +str(x_pos.value) + " \ Y: " + str(y_pos.value))
20        serialInst.write(command.encode('utf-8'))
21
22        time.sleep(ts)

```

Abordando agora o lado do Arduíno, como pode ser visto no trecho de código a seguir, o *setup* é realizado determinando as portas PWM as quais cada servo está

conectado, escolhendo a *baudrate* e iniciando os servomotores na posição de 0º, que não assumem esta angulação na planta efetivamente, ficando em torno dos 43º. Isso ocorre devido à uma limitação do Arduíno, que não permite alterar as larguras do sinal PWM máxima e mínima por ele enviado, como pode ser feito no *Raspberry*.

```

1 void setup() {
2
3     s1.attach(SERVO1);
4     s2.attach(SERVO2);
5     s3.attach(SERVO3);
6
7     Serial.begin(9600);
8
9     s1.write(0); // Inicia motor posicao zero
10    s2.write(0);
11    s3.write(0);
12 }
```

No *loop* principal visto no próximo trecho de código, é verificado se há requisição de comunicação serial na porta do Arduíno, e todo o código que vem após é executado somente se esse *if* retornar *true*. Caso haja envio de mensagem, que deverá ser recebida em formato padrão, é lida até a quebra de linha, tem seu tamanho determinado para ser convertida de *string* para *char array* e o *strtok* é aplicado em sequência para separar os ângulos *delta1*, *delta2* e *delta3*, que chegam todos juntos para o Arduíno. Foi utilizado um artifício para enviar valores *float* de duas casas decimais, no qual os valores dos ângulos são multiplicados por 100 e convertidos para números inteiros antes de serem enviados, por isso será necessário dividi-los por 100 em seguida, antes de serem finalmente escritos em cada servomotor.

```

1 void loop() {
2
3     if(Serial.available() > 0)
4     {
5         int val[3];
6         float delta[3];
7         String msg = Serial.readStringUntil('\n');
8         char msg_array[msg.length()];
9         msg.toCharArray(msg_array, msg.length());
```

```
11 char * token = strtok(msg_array, ", ");  
12  
13 int i = 0;  
14 while(token != NULL) {  
15     val[i] = atoi(token);  
16     token = strtok(NULL, ", ");  
17     i++;  
18 }  
19  
20 for(i=0; i<3;i++) {  
21     delta[i] = val[i];  
22     delta[i] = delta[i]/100;  
23 }  
24  
25 s1.write(delta[0]);  
26 s2.write(delta[1]);  
27 s3.write(delta[2]);  
28 }  
29 }
```