

# Autochess

Fernando F. Vieira  
Gabriel A. B. Caballero  
Marco V. Busetti

Oficina de Integração 1 (ELEX20)  
Engenharia de Computação - UTFPR

21 de junho de 2025

# Roteiro da Apresentação

1 Introdução

2 Hardware

3 Mecânica

4 Software

5 Resultados

# Introdução

1 Introdução

2 Hardware

3 Mecânica

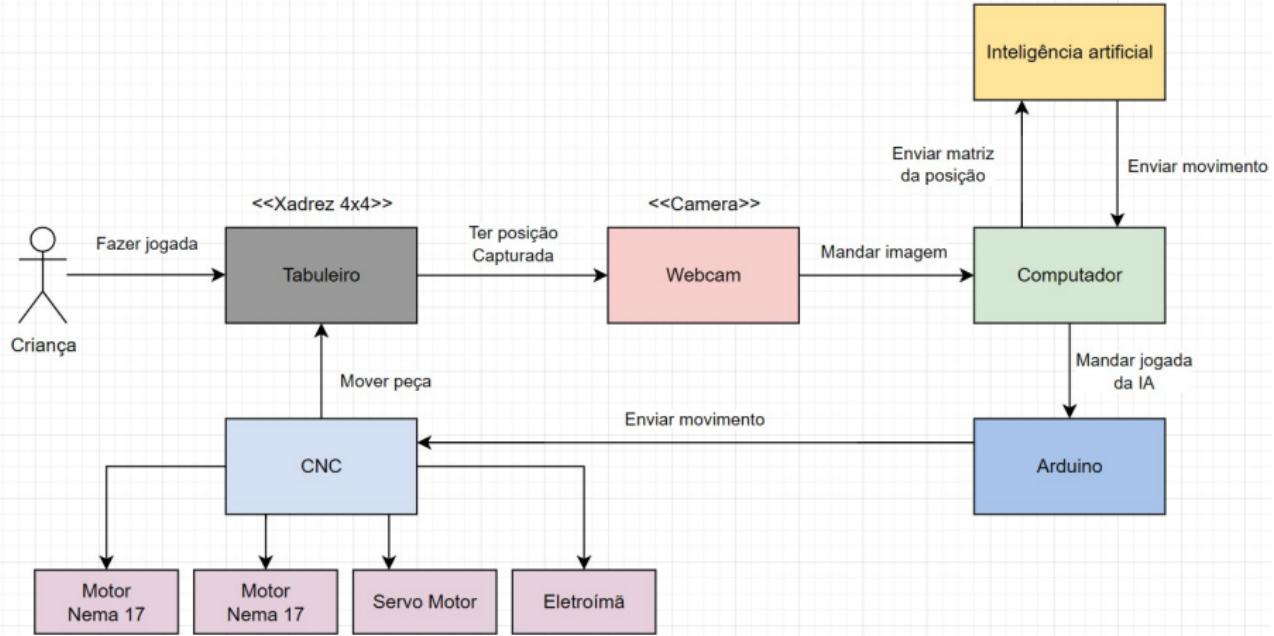
4 Software

5 Resultados

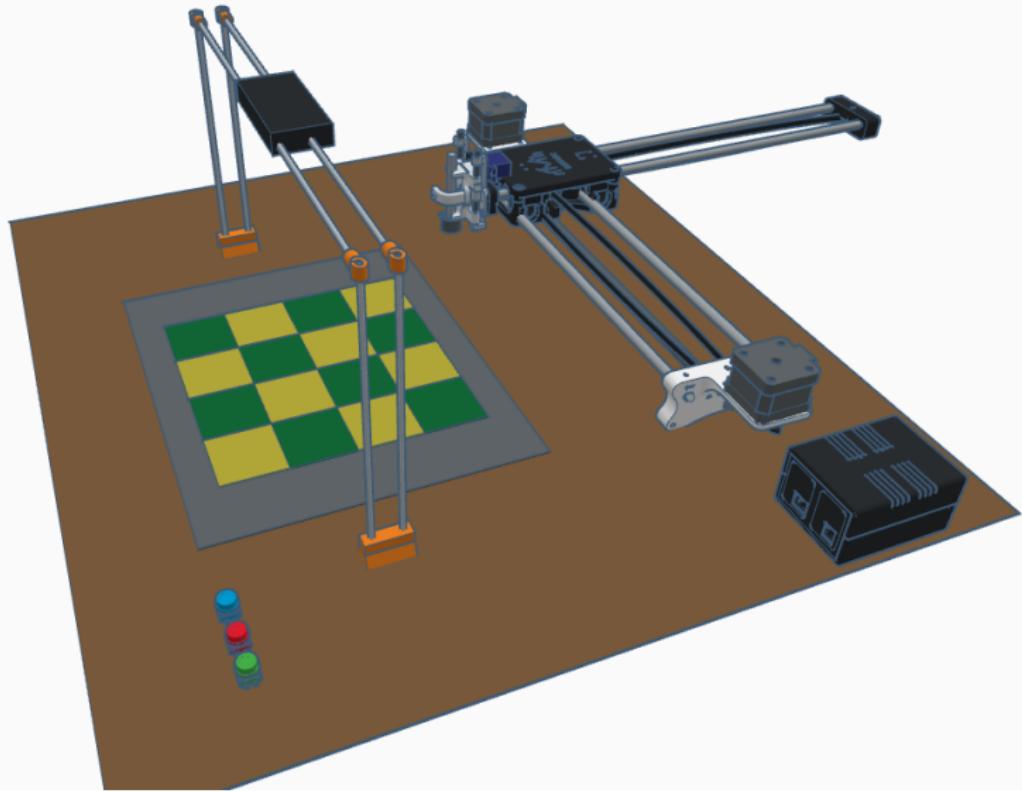
# Escopo e Objetivos

- O projeto **Autochess** foi desenvolvido com propósito pedagógico: introduzir conceitos de **Machine Learning** e **xadrez** para crianças.
- Utiliza um **tabuleiro 4x4** simplificado, composto por:
  - Módulo de **Inteligência Artificial**.
  - Módulo de **Visão Computacional**.
  - Sistema **CNC**.

# Diagrama de blocos



# Idealização do projeto



# Materiais e Orçamento

| Item                           | Quantidade | Preço (R\$)   |
|--------------------------------|------------|---------------|
| Arduino Uno                    | 2          | 40,00         |
| Shield CNC V3 + Drivers        | 1          | 15,00         |
| Eletroimã                      | 1          | 6,00          |
| Nema 17                        | 2          | 40,00         |
| Servo motor SG90               | 1          | 5,00          |
| Placa universal                | 1          | 12,00         |
| Patolas                        | 2          | 30,00         |
| Fonte de alimentação           | 2          | 50,00         |
| Webcam Full HD                 | 1          | 50,00         |
| Impressão 3D                   | -          | 100,00        |
| Base MDF                       | -          | 40,00         |
| Outros (fios, conectores, etc) | -          | 30,00         |
| <b>Total</b>                   |            | <b>418,00</b> |

*Observação: Componentes eletrônicos adquiridos via AliExpress.*

# Hardware

1 Introdução

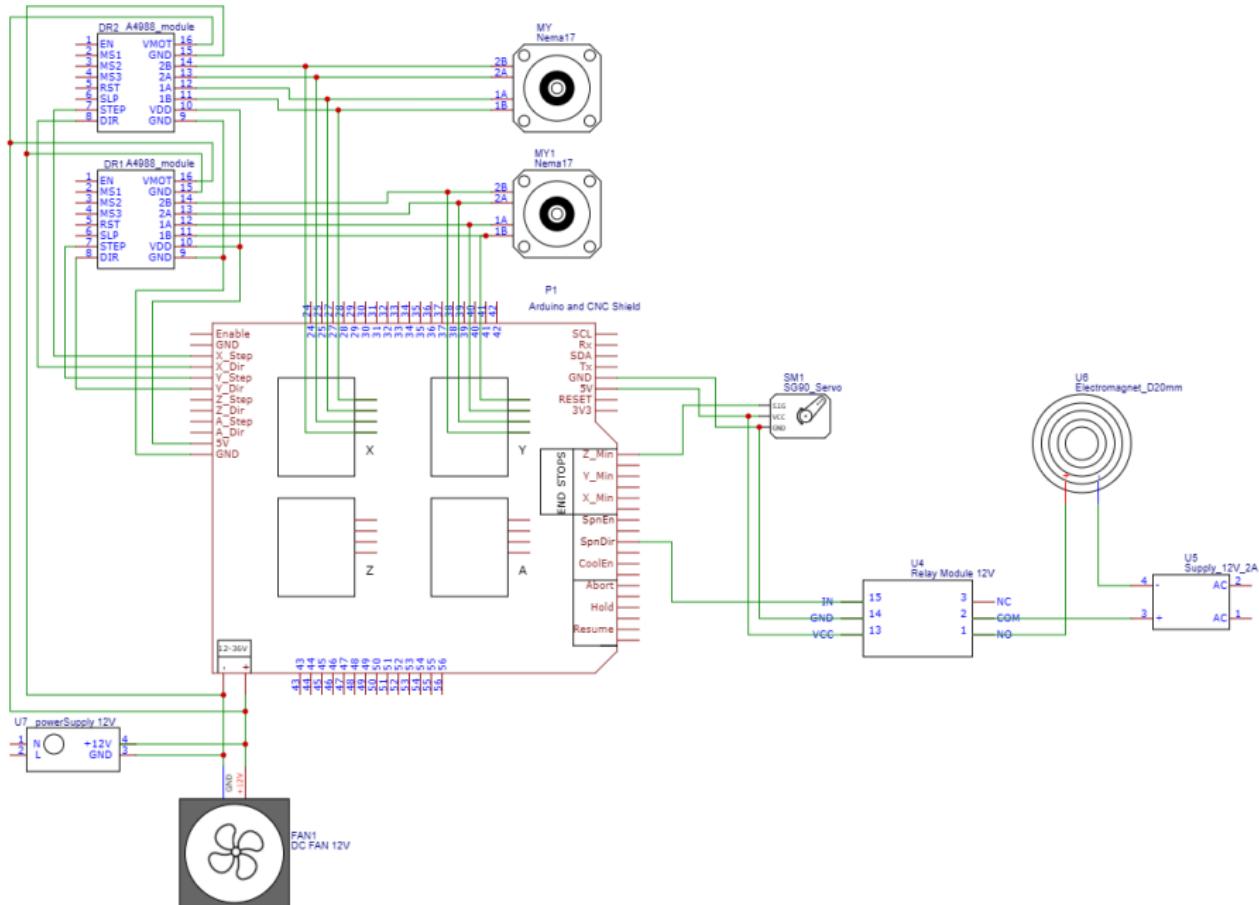
2 Hardware

3 Mecânica

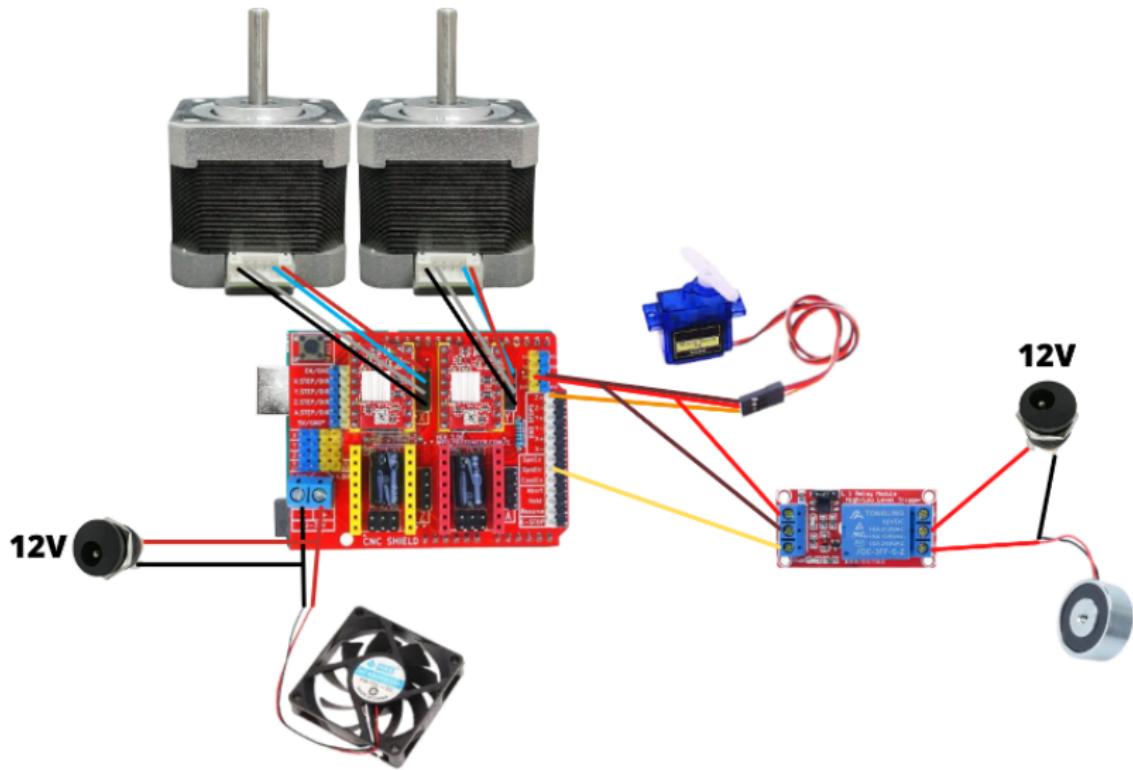
4 Software

5 Resultados

# Idealização do projeto elétrico



# Idealização do projeto elétrico



## Algumas modificações...

- Foram utilizados **2 Arduinos UNO**.
- Tarefas de cada um:
  - **Arduino 1:** Controle da CNC (*firmware GRBL-servo*);
  - **Arduino 2:** Receber input dos botões.

- Utilizado o **firmware GRBL-servo**, baseado no repositório da **Robottini**<sup>1</sup>.
- GRBL é um firmware open-source que interpreta comandos G-code para controle de CNCs.
- Permitiu o **controle dos motores de passo, eletroímã e servomotor** com precisão.
- Foram realizadas algumas modificações no código fonte do firmware para adaptar o ângulo de abertura do servomotor.

<sup>1</sup> <https://github.com/robottini/grbl-servo>

# Mecânica

1 Introdução

2 Hardware

3 Mecânica

4 Software

5 Resultados

# Estrutura CoreXY

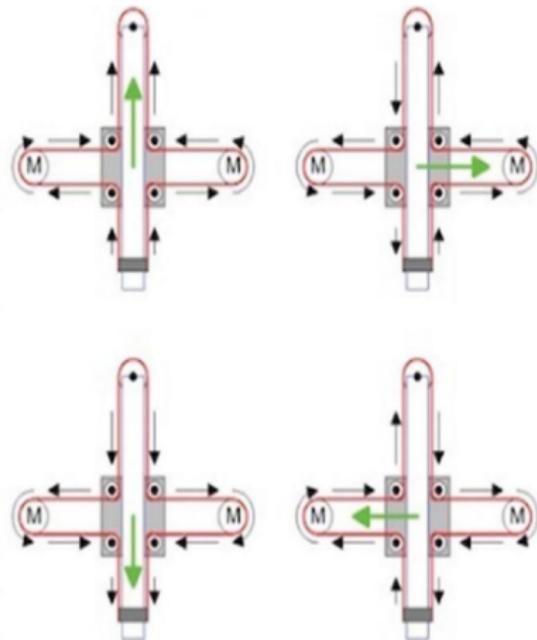


Figura: Estrutura cruzada CoreXY

Fonte: [test3dprints.com](http://test3dprints.com)

## CoreXY - Princípio de Funcionamento

- Os motores estão fixos na estrutura.
- O movimento é transmitido por apenas uma correia e polias dispostas em padrão cruzado.
- Os motores atuam juntos para gerar os deslocamentos nos eixos.

## Vantagens do CoreXY

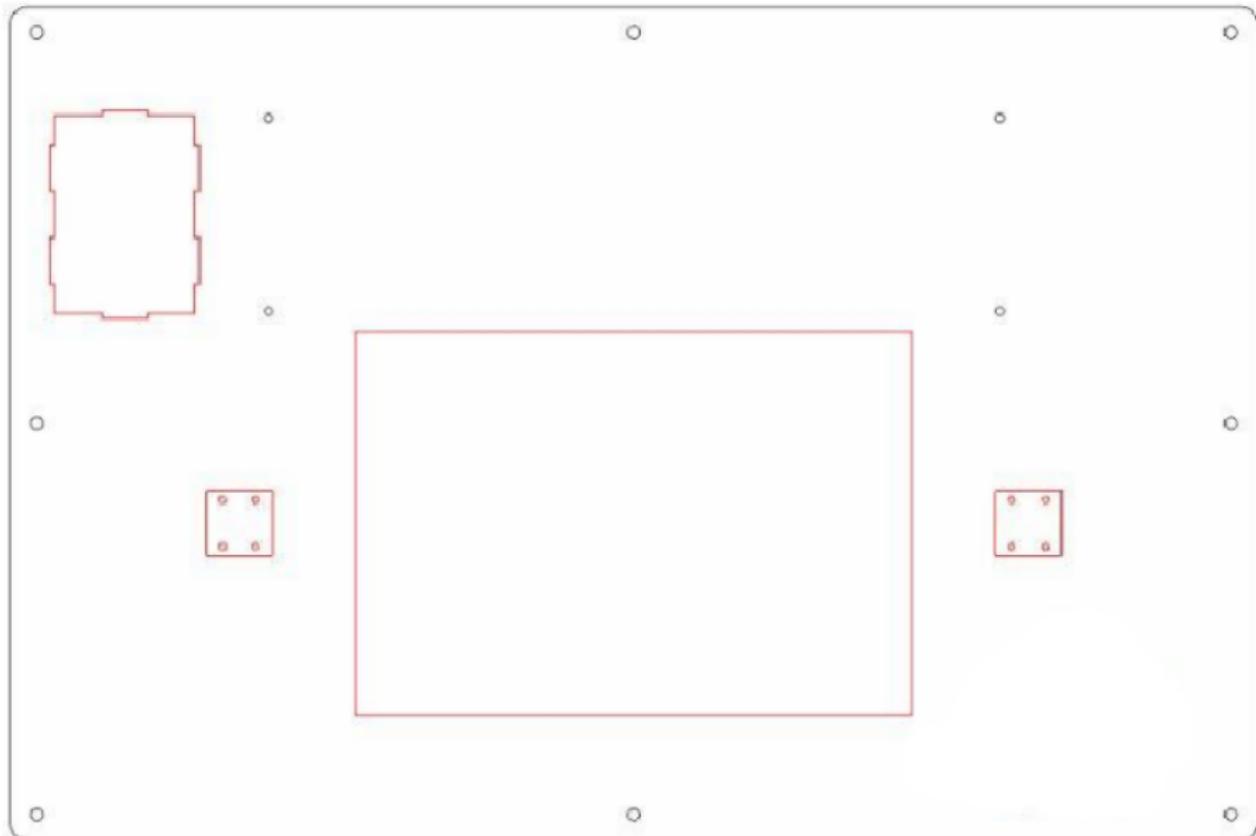
- Cabeçote leve (motores não se movem junto).
- Alta velocidade e aceleração.
- Estrutura simétrica e eficiente no uso do espaço.
- Redução de inércia no movimento.
- Uso de apenas uma correia.

## Estrutura física do projeto

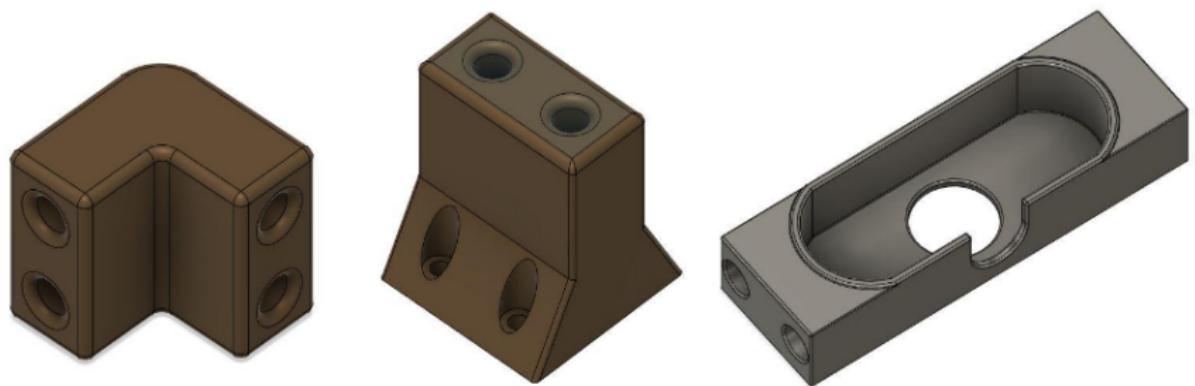
- Para o suporte da CNC foi utilizado como base o projeto **DrawingBot** da **MakerC<sup>2</sup>**.
- Arquivos para impressão 3D disponibilizados foram utilizados.
- Todos os objetos são apoiados em uma base de MDF cortada à laser.
- Fora a estrutura impressa 3D da CNC (já pronta), todos os outros objetos (peças de xadrez, suporte para câmera, etc.) foram modelados à parte.

<sup>2</sup> <https://www.thingiverse.com/thing:1517211>

# Modelo da base do MDF



# Modelos 3D - suporte para câmera



# Modelos 3D - peças do xadrez



Rainha



Base da Rainha

# Software

1 Introdução

2 Hardware

3 Mecânica

4 Software

5 Resultados

## Minichess 4×4 – Variante Silverman

- Minichess é uma **família de variantes** do xadrez tradicional, jogadas em tabuleiros menores.
- A variante **Silverman 4×4** é jogada com as seguintes peças:
  - Rei, torre, rainha e peões.
- Útil para estudo de algoritmos, resolução de jogos e IA aplicada.

## Minichess 4×4 – Variante Silverman

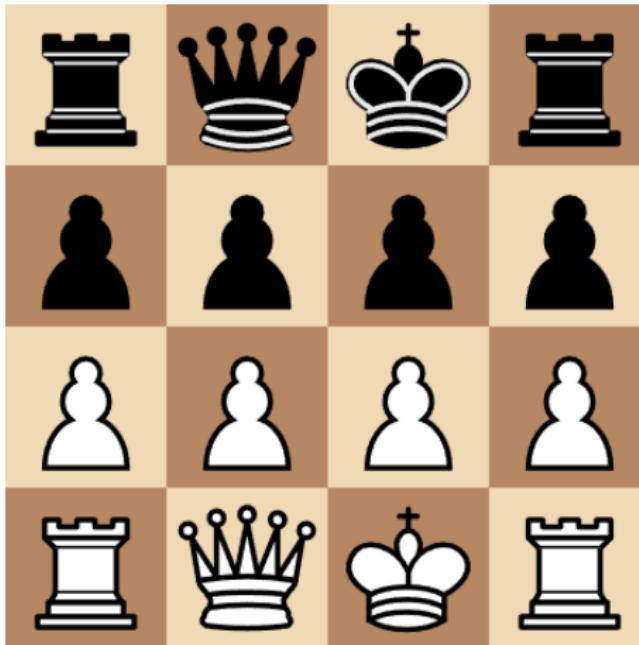


Figura: Silverman 4x4

Fonte: <https://en.wikipedia.org/wiki/Minichess>

# IA com Q-Learning no Silverman 4×4

- Modelamos o jogo como um **Processo de Decisão de Markov (MDP)**:
  - **Estados ( $S$ )**: todas as possíveis configurações do tabuleiro.
  - **Ações ( $A$ )**: todos os movimentos legais em um dado estado.
  - **Recompensa ( $R$ )**: +1 para vitória, 0 para empate, -1 para derrota.
- Utilizamos **Q-Learning**, um algoritmo off-policy de aprendizagem por reforço:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

- Estratégia  $\varepsilon$ -gulosa para equilibrar *exploration & exploitation*:

Escolhe melhor ação com probabilidade  $1 - \varepsilon$ , aleatória com  $\varepsilon$

## Parâmetros do Q-Learning e $\varepsilon$ -gulosa

- **Taxa de aprendizado  $\alpha$ :**  
→ Valor utilizado:  $\alpha = 0.5$ .
- **Fator de desconto  $\gamma$ :**  
→ Valor utilizado:  $\gamma = 0.9$ .
- **Taxa de exploração  $\varepsilon$  (estratégia  $\varepsilon$ -gulosa):**  
→ Inicialmente alta (ex: 0.9), decresce ao longo do treinamento para valores baixos (ex: 0.1 ou 0.01).

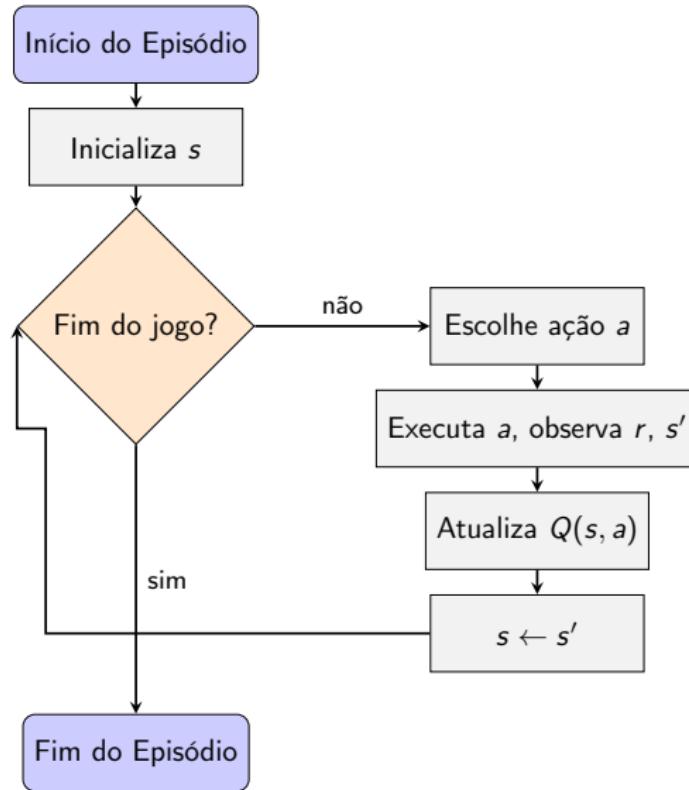
# Q-Learning

```
1 Inicializar  $Q(s, a)$  arbitrariamente
2 for cada episódio (partida) do
3     Inicializar estado  $s$ 
4     while jogo não terminou do
5         Com probabilidade  $\varepsilon$ , escolher ação aleatória  $a$ 
6         Caso contrário, escolher  $a = \arg \max_{a'} Q(s, a')$ 
7         Executar ação  $a$ , observar recompensa  $r$  e novo
        estado  $s'$ 
8         Atualizar:
9             
$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

10        
$$s \leftarrow s'$$

11    end
12 end
```

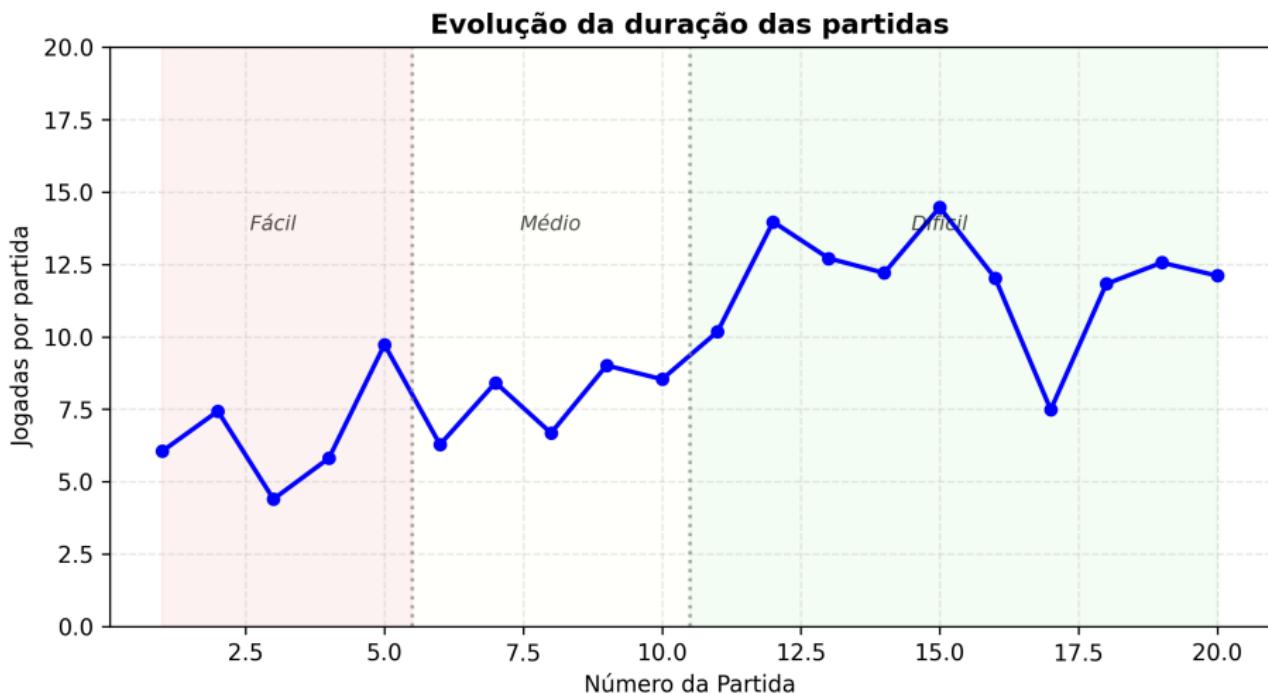
# Q-Learning



# Metodologia de Avaliação da IA

- **Número de sessões:** 30 sessões distintas, com reinicialização da IA ao início de cada sessão.
- **Partidas por sessão:** 20 partidas sequenciais por sessão, totalizando 600 partidas analisadas.
- **Métricas utilizadas:**
  - ① Média de jogadas por partida;
  - ② Número de blunders;
  - ③ Saldo de material;
  - ④ Jogadas seguras (%).
- Média de todas sessões de uma determinada partida.

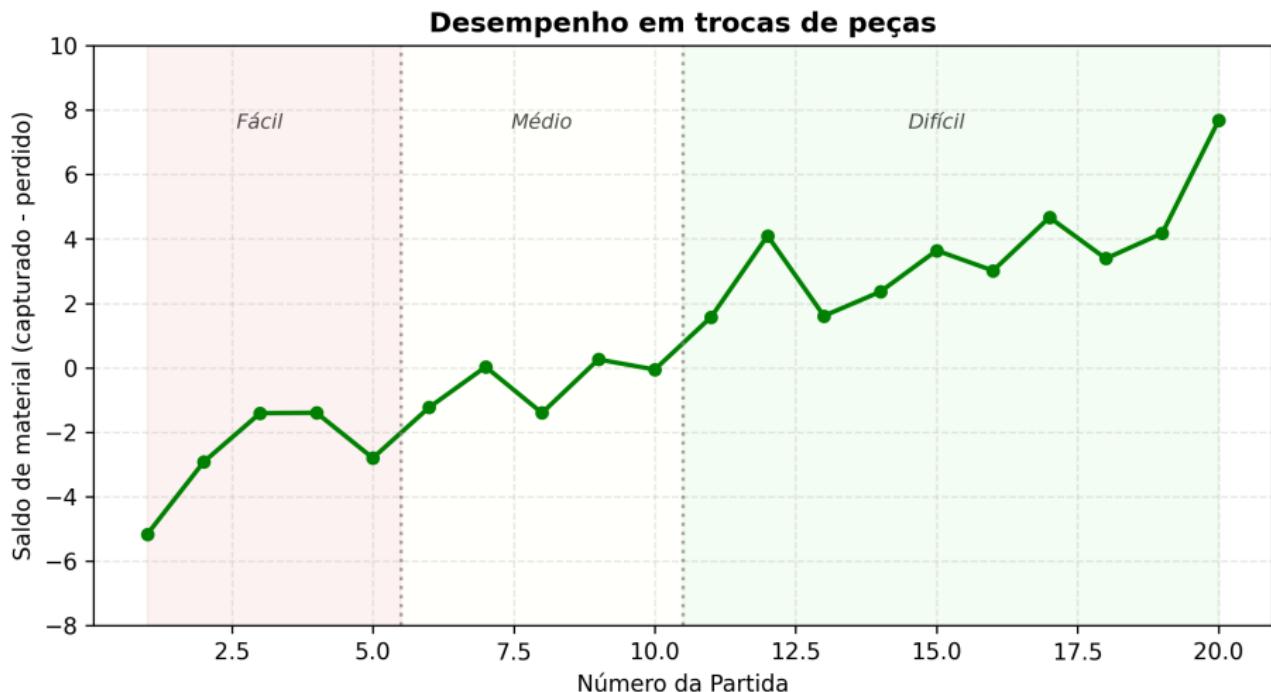
# Q-Learning - Levantamento estatístico sob os resultados



# Q-Learning - Levantamento estatístico sob os resultados

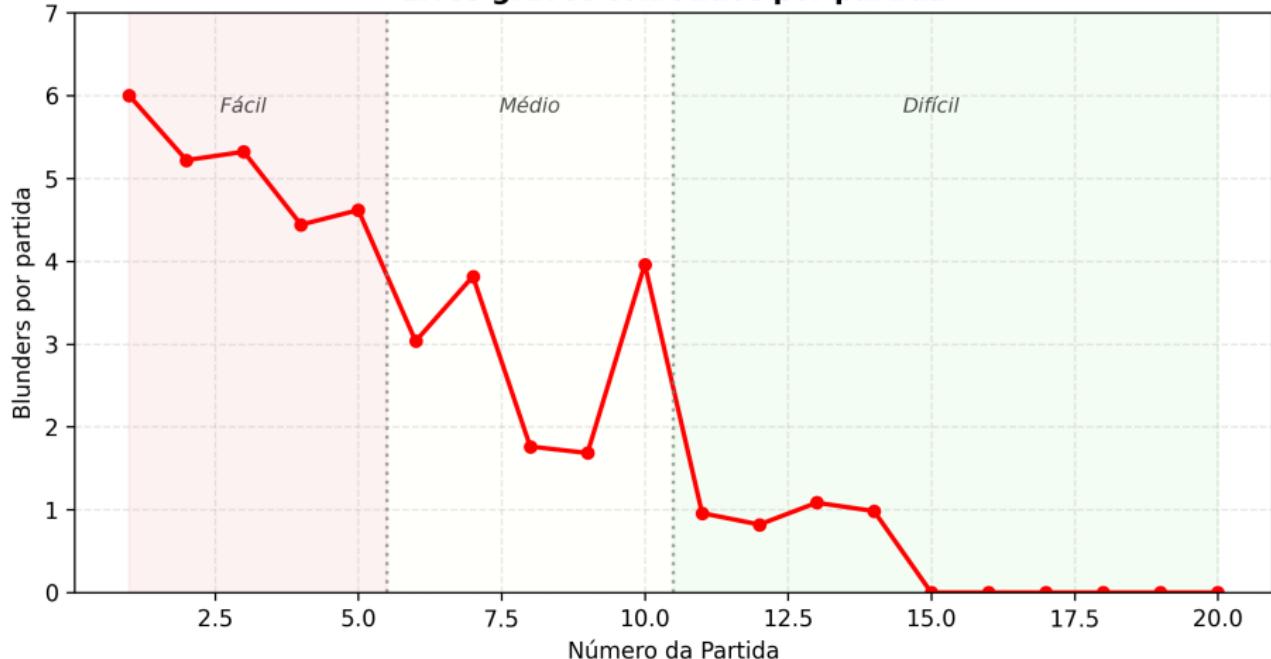


# Q-Learning - Levantamento estatístico sob os resultados



# Q-Learning - Levantamento estatístico sob os resultados

**Erros graves cometidos por partida**



# Visão Computacional

- Para identificar o estado do jogo automaticamente, foi necessário utilizar técnicas de visão computacional sobre imagens capturadas por uma câmera fixa.

## Desafios enfrentados

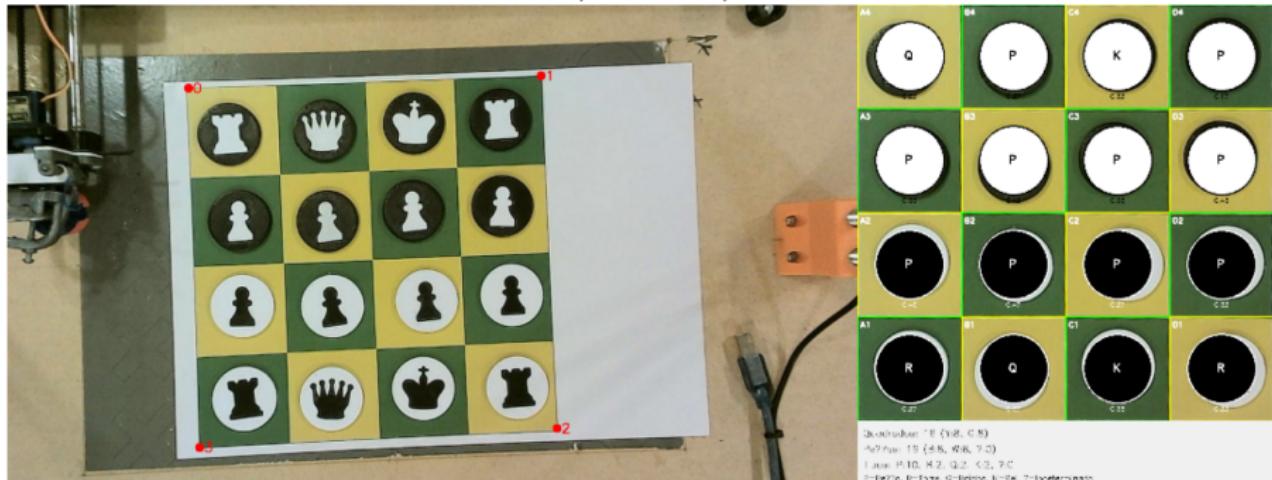
- Segmentação precisa da área do tabuleiro, mesmo com distorções da câmera.
- Diferenciação entre peças brancas e pretas sob diferentes condições de iluminação.
- Identificação das peças no tabuleiro, considerando posições variadas.

## Solução adotada

- Utilizamos o **SIFT (Scale-Invariant Feature Transform)** para detecção de pontos-chave robustos.
- Detecção de contornos das figuras das peças e extremidades do tabuleiro.
- As descrições extraídas com SIFT permitiram realizar *matching* entre peças reais e modelos de referência.

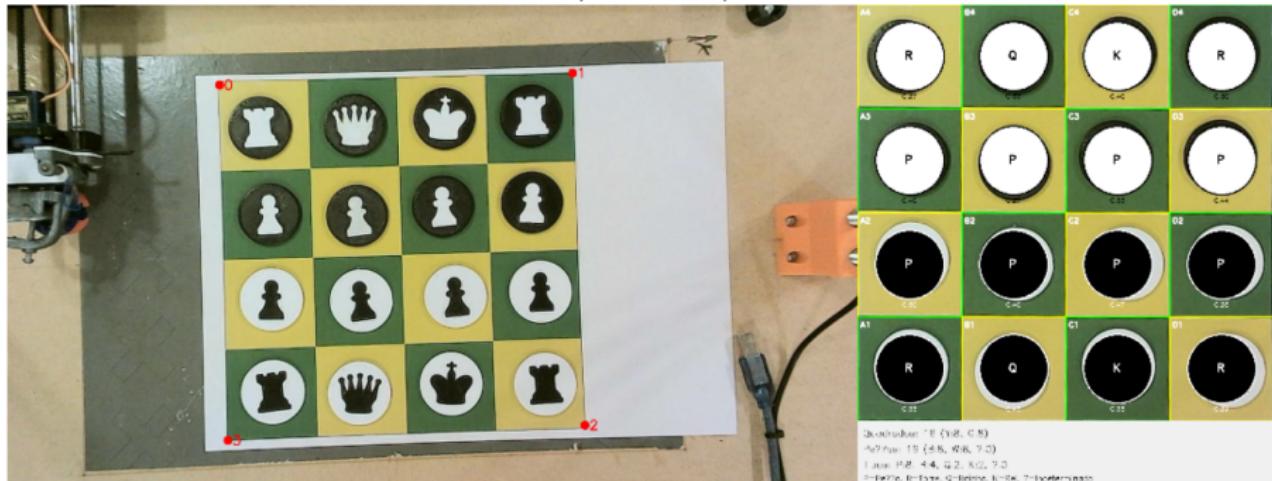
# Implementação do SIFT - Exemplos

Detecção de Tabuleiro e Peças



# Implementação do SIFT - Exemplos

Detecção de Tabuleiro e Peças



# Implementação do SIFT - Exemplos

Detectão de Tabuleiro e Peças



|    |    |    |    |    |
|----|----|----|----|----|
| A4 | P  | R  | K  | P  |
| B4 | C4 | C4 | C4 | D4 |
| A3 | P  | Q  | C3 | C3 |
| B3 | C3 | C3 | C3 | D3 |
| A2 | P  | P  | P  | P  |
| B2 | C2 | C2 | C2 | D2 |
| A1 | R  | K  | C1 | C1 |
| B1 | C1 | C1 | C1 | D1 |

Coordenadas: 0: {B8, C8}  
Peça: R  
Coordenadas: 1: {E8, F8, G8, H8}  
Peça: Q  
Coordenadas: 2: {A1, B1, C1, D1}  
Peça: P  
Coordenadas: 3: {A8, B8, C8, D8}  
Peça: K

# Implementação do SIFT - Exemplos

Detectão de Tabuleiro e Peças



|    |    |    |    |
|----|----|----|----|
| A4 | B4 | C4 | D4 |
| B3 | C3 | D3 | E3 |
| C2 | D2 | E2 | F2 |
| D1 | E1 | F1 | G1 |
| E0 | F0 | G0 | H0 |

Quadrado: E (Y=8, C=8)  
Peça: 14 (E7, W7, 7C)  
Lado: P8, 4E, 4Z, K0, 7D  
Pecas: R-Este, Q-Branco, K-Direita, 7-Inferior-Direito

# Resultados

1 Introdução

2 Hardware

3 Mecânica

4 Software

5 Resultados

# Resultados - decisões que funcionaram bem

- **Hardware:**
  - Comunicação serial bem sucedida: *firmare* GRBL servo funcionou como deveria;
- **Mecânica:**
  - Estrutura do braço CNC forte e confiável;
  - Peças de xadrez 3D respondendo perfeitamente às atrações magnéticas.
- **Software:**
  - Machine learning apresentou evolução com o decorrer das partidas;
  - Visão Computacional confiável em diversas condições de luz e posições de peças.

# Resultados - problemas a serem melhorados

- **Hardware:**
  - Utilização de dois Arduinos para comunicação serial;
- **Mecânica:**
  - Base MDF causando desníveis;
  - Um motor de passo "mais lento" que o outro.
- **Software:**
  - Visão Computacional tomando muito tempo para processamento de tabuleiro cheio.

## Conclusão dos Resultados

- O sistema funcionou bem como uma ferramenta prática e didática para apresentar aprendizado de máquina a um público infantil.
- A experiência mostrou que a escolha das ferramentas e abordagens foi acertada:
  - O uso do GRBL com Arduinos se provou uma solução estável e prática;
  - A estrutura mecânica funcionou como esperado, e as peças responderam bem ao controle magnético;
  - A IA conseguiu aprender com poucas partidas, o que validou a proposta de demonstrar aprendizado de máquina de forma acessível e dinâmica;
  - A visão computacional se manteve confiável mesmo com variações de luz e disposição das peças.

## Próximos Passos

- Melhorar a estrutura da base para evitar desníveis;
- Unificar o controle em um único microcontrolador;
- Otimizar a visão computacional para situações com muitas peças.

Muito obrigado!

Muito obrigado pela atenção!