# SISTEMA ROBOTIZADO DE XADREZ

**Filipe Sousa, José Gonçalves, Paulo Leitão**

Instituto Politécnico de Bragança, Quinta de St$^{a}$ Apolónia
134, 5301-857 Bragança

*filipe.gr.sousa@alunos.ipb.pt,{goncalves,pleitao}@ipb.pt*

Resumo: A utilização de sistemas robotizados na indústria apresenta vantagens em termos económicos e técnicos, essencialmente permitindo um incremento de produtividade, robustez e qualidade. Este artigo apresenta um sistema robotizado com aspectos lúdicos que permite demonstrar aos alunos e visitantes da Escola Superior de Tecnologia e Gestão (ESTiG) do Instituto Politécnico de Bragança (IPB) as capacidades associadas a estes sistemas. A aplicação computacional desenvolvida implementa um jogo de xadrez, cujo movimento fisico das peças é realizado por um *robot*.

## 1. INTRODUÇÃO

Um *robot* industrial é uma máquina multi-aplicação e reprogramável possuindo características antropomórficas. A utilização de sistemas robotizados visa extrair vantagens em termos técnicos, como sejam o incremento da qualidade, robustez, flexibilidade e suporte de ambientes hostis. Também apresentam vantagens a nível económico, como sejam o incremento da produtividade e redução do WIP (Work In Progress) (P.Groover *et al.*, 1989). As aplicações mais comuns de *robots* referem-se à soldadura, pintura, montagem e manipulação de materiais.

O laboratório de Automação da ESTiG possui um *robot* industrial ABB IRB 1400, visando a aprendizagem de competências na área da robótica industrial, quer no âmbito de actividades lectivas quer de investigação.

Tendo em vista demonstrar algumas das potencialidades da robótica, o sistema desenvolvido compreende basicamente os seguintes componentes:

- A aplicação computacional, que implementa o motor de xadrez, as interfaces com o utilizador e com o mundo físico.
- Um tabuleiro de xadrez, onde os movimentos das peças do jogo da aplicação computacional são mapeadas fisicamente.
- Um *robot* que executa fisicamente os movimentos das peças no tabuleiro, de acordo com os comandos da aplicação computacional.

Este artigo está organizado da seguinte forma: na secção 2 são descritos os módulos constituintes do sistema robotizado, a secção 3 descreve a programação realizada no *robot* e a secção 4 descreve a implementação do actuador final. Por fim são apresentadas algumas conclusões e apontando algum trabalho futuro.

## 2. SISTEMA ROBOTIZADO

Em termos académicos, foi desenvolvida uma aplicação que utiliza o *robot* existente no laboratório de Automação da ESTiG para jogar xadrez. O sistema robotizado desenvolvido compreende uma aplicação computacional, o *robot* e o tabuleiro de xadrez, como ilustrado na Figura 1.

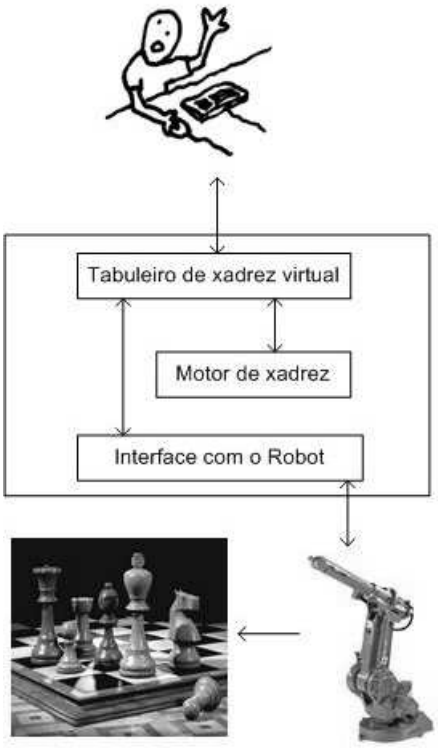


Figura 1. Módulos do sistema robotizado

A aplicação computacional, que funciona como o cérebro do sistema, compreende a interface com o utilizador, o motor de xadrez e a interface com o *robot*. Cada uma destas entidades comunica usando protocolos próprios.

### *2.1 Motor de xadrez*

Neste trabalho optou-se por integrar um motor de xadrez disponível ao invés de se desenvolver um de raiz, uma vez que o foco do trabalho é o sistema robotizado. Assim o motor de xadrez escolhido foi o motor GnuChess (*Gnuchess*, 2004), o qual é um programa executável em modo de texto e de domínio público (*General Public License*, 2004). Este motor de xadrez corre como um processo separado da interface gráfica do xadrez, comunicando com esta através de *pipes* anónimos (Johnson and Troan, 1998). O motor inicia dois *pipes*, um para o *standard input* e outro para o *standard output*. O motor de xadrez lê os comandos a partir do *standard input* e escreve as respostas no *standard output*.

O estado do jogo e outros parâmetros são controlados através de comandos que obedecem ao protocolo *xboard* (*Chess Engine Communication Protocol*, 2004), muito comum em motores de xadrez. O protocolo funciona de modo assíncrono, i.e, a entidade que envia a mensagem continua a sua actividade, não ficando bloqueada à espera da resposta.

Neste trabalho o comando mais utilizado é a ordem para mover uma peça no tabuleiro de xadrez através de coordenadas em notação algébrica. Por exemplo, a coordenada `a2a3` é um comando para mover uma peça da posição `a2` para `a3`, caso a jogada seja válida o motor de xadrez responde com as coordenadas `d7d5`, ou seja, neste caso a resposta é um movimento de uma peça preta .

Exemplo de uso directo do motor de xadrez para ilustrar o exemplo fornecido é ilustrado de seguida:

```
# gnuchess.exe -x
Chess
Adjusting HashSize to 1024 slots
a2a3
1. a2a3
1. ... d7d5
My move is: d7d5
```

Nem sempre a resposta a uma coordenada é outra coordenada, pode acontecer que a resposta seja uma indicação de que se tentou mover uma peça que viole as regras do jogo.

### *2.2 Tabuleiro de xadrez virtual*

O tabuleiro de xadrez virtual, Figura 2, é o módulo gráfico que permite a interface com o utilizador, comunicando com o motor de xadrez através de *pipes*. O tabuleiro virtual reflecte todo o estado do jogo através de uma representação gráfica de um tabuleiro de xadrez.

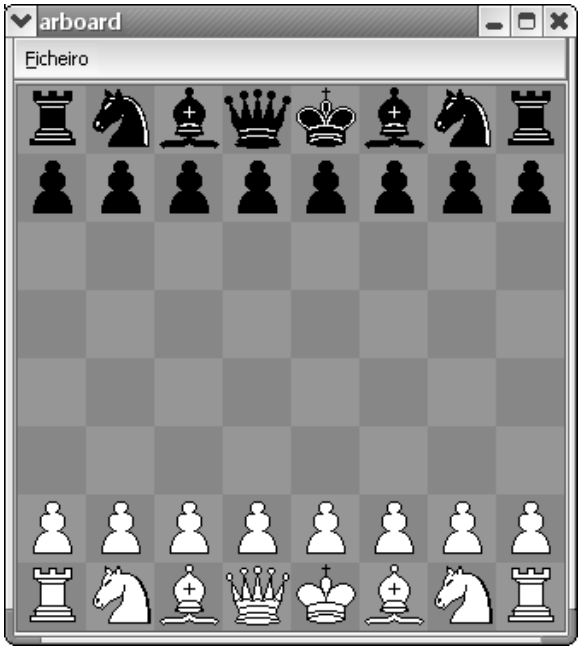


Figura 2. Tabuleiro de xadrez

A interface gráfica foi criada de raíz, ao invés de modificarmos uma existente, como por exemplo o xboard (*xboard*, 2004) ou o winboard (*Winboard*, 2004) respectivamente para a plataforma Unix e Microsoft Windows. A opção de criar uma interface gráfica de raiz deveu-se à necessidade de obter controlo absoluto sobre o jogo (por exemplo, esperar que o *robot* acabe de jogar) e a desnecessária complexidade das interfaces de xadrez existentes. A linguagem de programação de eleição usada para desenvolver todos os componentes da interface gráfica foi o C++ (Stroustrup, 2000).

A interface gráfica tem duas componentes principais:

(1) Tabuleiro de xadrez – desenvolvido usando o *toolkit* Qt (Trolltech, 2003) que permite desenvolver aplicações gráficas de uma forma amigável.
(2) Componente que implementa a comunicação com a interface de *robot* desenvolvido usando a *framework* e *Microsoft Foundation Classes (MFC)*.

A comunicação entres estas duas componentes foi conseguida através da abstracção de interfaces, usando um *design pattern* (Gamma *et al.*, 1994).

*2.2.1. Algoritmo de gestão de xadrez* A interacção do jogador com jogo é realizada com o auxílio do rato para permitir mover as peças brancas, sendo as peças pretas movimentadas pelo oponente, que neste caso é o computador.

Por cada jogada é enviado um comando que é escrito no *standard output* da interface gráfica. O motor de xadrez lê do seu *standard input* o comando enviado, processa o comando e escreve a(s) resposta(s) no *standard output*. Por sua vez a interface obtém a(s) resposta(s) lendo o seu *standard input*.

O algoritmo usado está exemplificado na Figura 3.

*2.2.2. Comunicação com a interface do robot* A comunicação das jogadas ao *robot* consiste no envio de sequências de valores reais previamente definidos.

A execução de um movimento de uma peça de uma posição inicial para uma posição final no tabuleiro real do jogo, requer o envio de uma sequência de informação, Figura 4, que compreende os campos:

- $x1, y1$ – coordenadas de origem da peça que vai ser movida.
- $x2, y2$ – coordenadas de destino para onde a peça vai ser largada.
- $flag$ – o valor da $flag$ neste caso é 1.

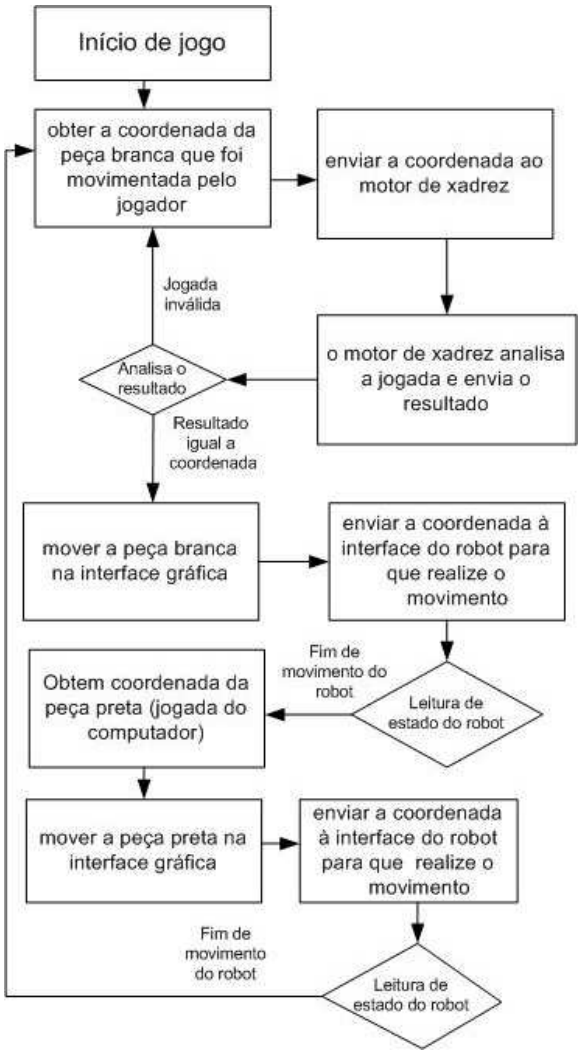


Figura 3. Algoritmo de gestão de xadrez

| x1 | y1 | x2 | y2 | flag |
|---|---|---|---|---|

Figura 4. Sequência para mover uma peça do tabuleiro

Os valores de $x$ e $y$ são calculados através de uma conversão directa de uma coordenada algébrica para valores inteiros e depois multiplicados por um factor de escala. Este factor de escala corresponde ao tamanho de cada quadrícula no tabuleiro de xadrez real.

| x | y | flag |
|---|---|---|

Figura 5. Sequência para retirar uma peça do tabuleiro

A sequência da figura 5 é enviada à interface de *robot* sempre que seja necessário retirar uma peça do tabuleiro de xadrez. Da mesma forma que a sequência descrita anteriormente, também neste caso é necessário converter as coordenadas

algébricas e aplicar o mesmo factor de escala a cada um dos seguintes valores:

- $x$ – coordenada $x$ da peça a ser retirada.
- $y$ – coordenada $y$ da peça a ser retirada.

Neste caso o valor da $flag$ é 2.

### *2.3 Interface do* robot

O *robot*, durante o jogo de xadrez, executa continuamente um programa que permite agarrar uma peça de uma posição inicial para a largar numa posição final. O programa do robot possui 5 registos de vital importância:

- `regx1` – coordenada x de origem da peça a movimentar (registo de escrita).
- `regy1` – coordenada y de origem da peça a movimentar (registo de escrita).
- `regx2` – coordenada x de destino da peça a movimentar (registo de escrita).
- `regy2` – coordenada y de destino da peça a movimentar (registo de escrita).
- `reg11` – Esta flag indica a acção a executar, podendo possuir 3 valores distintos (registo de leitura e escrita):
  - 0 – o *robot* está em movimento (leitura).
  - 1 – o *robot* deve mover uma peça de `regx1`, `regy1` para `regx2`, `regy2` (escrita).
  - 2 – o *robot* deve retirar a peça que está em `regx1`, `regy1` para fora do tabuleiro (escrita).

O acesso da aplicação computacional do *robot* para actualizar os registos previamente referidos, é realizado pela interface com o *robot*, que utiliza o *ActiveX RobComm* fornecido pela ABB (ABB, 1996). Esta interface utiliza essencialmente as seguinte primitivas disponibilizadas para manipular o *robot*:

- A primitiva `S4ProgramNumVarWrite` que permite alterar um registo usado no programa do *robot*.
- A primitiva `S4ProgramNumVarRead` que permite obter o valor de um registo usado no programa do *robot*.

## 3. PROGRAMAÇÃO DO *ROBOT*

Os movimentos dos *robots* devem ser capazes de alcançar uma posição alvo programada de forma a posicionar uma peça ou uma ferramenta e também mover-se ao longo de uma trajectória com velocidades definidas. Isto significa que um programa de um *robot* pode ser interpretado como uma sequência de localizações no espaço que o *robot* tem que percorrer com uma determinada sequência, orientação, velocidade e aceleração.

A linguagem de programação do *robot* ABB IRB1400 é a linguagem *Rapid*, efectuando o controlo do *robot* de uma forma simples e amigável para o utilizador.

O *software* desenvolvido em *Rapid*, funcionando em ciclo infinito, permite inicialmente colocar o *robot* numa posição de repouso e abrir a garra, usando respectivamente o procedimento `repouso()` e `abregarra()`.

Durante este ciclo, o programa do *robot* pode evoluir em duas direcções, de acordo com o valor existente em `reg11`. No caso de `reg11` ser 1, o *robot* move peças de acordo com os valores guardados em `regx1, regy1, regx2, regy2`, executando os procedimentos agarrar e largar. No caso de `reg11` ser igual a 2, então a peça do tabuleiro é removida, através da execução dos procedimentos `agarra()` e `remover()`.

O código usado foi o seguinte:

```
 ...

PROC routine1()
    <SMT>
  ENDPROC

  PROC main()
    IF regab=0 THEN
      repouso;
      abregarra;
      regab:=1;
    ENDIF

    IF reg11=1 THEN
      agarra;
      larga;
      reg11:=0;
    ENDIF

    IF reg11=2 THEN
      agarra;
      remover;
      reg11:=0;
    ENDIF
  ENDPROC

 ...
```

Os procedimentos usados para agarrar e largar peças seguem uma metodologia comum a todas as linguagens de programação de *robots* industriais, definindo-se uma zona livre, 10 cm acima do tabuleiro, na qual não existe limitações a nível de movimentos e velocidades, e uma zona condicionada a menos de 10 cm do tabuleiro, na qual usamos movimentos rectilíneos e velocidade condicionada.

O procedimento `agarra()` compreende a deslocação do *robot* da posição de repouso para a posição de origem, o fechar da garra e por fim a deslocação do braço para uma posição intermédia.

O procedimento `largar()` compreende a deslocação da peça para a posição de destino, o abrir das garras para largar a peça e o retorno do braço manipulador à posição de repouso.

O código usado para estes procedimentos foi o seguinte:

```
 ...

PROC agarra()
    MoveJ Offs(p30,regy1,-regx1,100)
    ,v500,fine,tool0;
    MoveL Offs(p30,regy1,-regx1,0
    ,v100,fine,tool0;
    fecha_garra;
    MoveL Offs(p30,regy1,-regx1,100)
    ,v500,fine,tool0;
  ENDPROC

  PROC largar()
    MoveJ Offs(p30,regy2,-regx2,100)
    ,v500,fine,tool0;
    MoveL Offs(p30,regy2,-regx2,0)
    ,v100,fine,tool0;
    abre_garra;
    MoveL Offs(p30,regy2,-regx2,100)
    ,v100,fine,tool0;
    repouso;

 ...
```

## 4. ACTUADOR FINAL

Os actuadores finais de um *robot* industrial permitem actuar sobre o mundo exterior com o propósito de conseguirem alcançar os seus objectivos da forma mais eficaz possível (P.Groover *et al.*, 1989).

O actuador final desta aplicação é uma garra mecânica accionada com energia pneumática, sendo capaz de abrir e fechar os seus dedos e exercer força suficiente, de maneira a que quando fechada consiga segurar as peças que foram agarradas. Para a aplicação de jogar xadrez foi projectada uma garra, representada na Figura 6, que pudesse agarrar uma peça de xadrez sem que colida com as peças que se encontram ao seu lado. Os dois buracos representados na garra são utilizados para fixar a garra ao manipulador sendo usados parafusos.

As peças de xadrez, Figura 7, encontram-se sobre uma base cilíndrica de $3,4cm$ de diâmetro, sendo

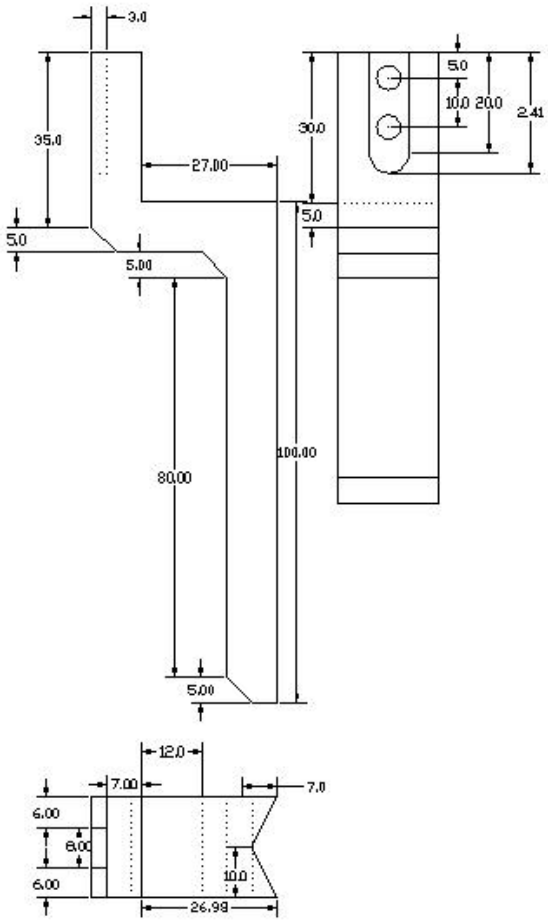


Figura 6. Garra mecânica

agarradas pela base e tendo o tabuleiro $5cm$ de lado de cada casa. O tabuleiro encontra-se colado ao chão, Figura 8, de maneira a que não sejam perdidas as referências, sendo todos os movimentos efectuados em relação a um ponto de referência do tabuleiro.

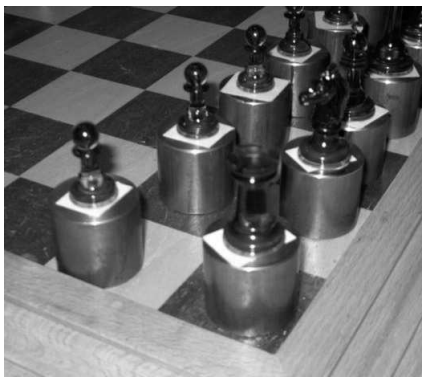

Figura 7. Peças do jogo de xadrez

Figura 8. *Robot* a jogar xadrez

## 5. CONCLUSÕES E TRABALHO FUTURO

Neste artigo pretendeu-se descrever a implementação de uma aplicação com aspectos lúdicos que permite demonstrar aos alunos e visitantes da ESTiG as capacidades associadas a sistemas robotizados, não sendo o objectivo deste trabalho o ser completamente original pois já foram realizadas coisas semelhantes sendo o exemplo: 'Eric a chess playing robot' (Pires, 1999). A aplicação desenvolvida cumpre os requisitos propostos, mas tem como limitação o facto de funcionar em malha aberta, não existindo *feedback*. O passo seguinte será usar sensores de visão para verificar se as peças estão no local correcto, para evitar colisões com a garra e evitar colocar uma peça onde já se encontra um objecto, a fim de permitir a identificação de objectos e determinação da sua geometria. O sistema de visão será constituído por uma câmara CCD (*Charged Coupled Device*), hardware de digitalização dos valores das intensidades de luz obtidas na matriz (um PC ou hardware dedicado a tratamento de imagem). A precisão do *robot* é de um milímetro e a repetibilidade depende da calibração de um ponto do tabuleiro relativamente ao qual são feitos movimentos. Com o tempo vão-se acumulando erros no posicionamento de peças, quando este erro se tornar significativo procede-se à calibração do *robot*. Da observação de vários jogos apercebemos-nos que este erro não era muito representativo pois seria necessário que os jogos fossem muito extensos para que existissem falhas na manipulação de peças.

## 6. AGRADECIMENTOS

Queremos agradecer ao Francisco Casais pelo apoio prestado, ao Laboratório de Fabricação Assistida por Computador da ESTiG pela elaboração das garras e peças do xadrez utilizadas e ao Centro de Tecnologias de Informação da ESTiG pela realização das filmagens do *robot* a jogar xadrez.

## REFERÊNCIAS

ABB (1996). *RobComm user´s guide*. ABB Flexible Automation inc.

*Chess Engine Communication Protocol* (2004). http://www.tim-mann.org/xboard/engine-intf.html.

Gamma, Erich, Richard Helm, Ralph Johnson and John Vlissides (1994). *Design Patterns*. Addison-Wesley.

*General Public License* (2004). http://www.gnu.org/copyleft/gpl.html.

*Gnuchess* (2004). http://www.gnu.org/software/chess/chess.html.

Johnson, Michael K. and Erik W. Troan (1998). *Linux Application Development*. Addison-Wesley.

P.Groover, Mikell, Mitchel Weiss, Roger N.Nagel and Nicholas G.Odrey (1989). *Robótica Tecnologia e programação*. McGraw-Hill.

Pires, Norberto (1999). Eric a chess playing robot. http://robotics.dem.uc.pt/norberto/eric/.

Stroustrup, B. (2000). *The C++ Programming Language*. Addison-Wesley.

Trolltech (2003). Qt trolltech toolkit. http://www.trolltech.com.

*Winboard* (2004). http://www.tim-mann.org/xboard.html.

*xboard* (2004). http://www.tim-mann.org/xboard.html.