Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

X.A.P.R.O
(Xadrez Autômato Profissional)

Victor Propheta Erbano

Dr. Dalton Pedroso de Queiroz (Orientador)

Dourados - MS 2017

X.A.P.R.O
(Xadrez Autômato Profissional)

Victor Propheta Erbano

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por Victor Propheta Erbano e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Dourados, 27 de outubro de 2017.

Prof. Dr. Dalton Pedroso de Queiroz (Orientador)

Curso de Ciência da Computação

Universidade Estadual de Mato Grosso do Sul

X.A.P.R.O
(Xadrez Autômato Profissional)

Victor Propheta Erbano

Outubro de 2017

Banca Examinadora:

Prof. Dr. Dalton Pedroso de Queiroz (Orientador) /Área de Computação – UEMS

Prof. Dr. Evandro Cesar Bracht /Área de Computação – UEMS

Profª MSc. Jéssica Bassani de Oliveira / Área de Computação – UEMS

AGRADECIMENTOS

Agradeço a todos aqueles que me ajudaram, de uma forma ou outra, a realizar uma trajetória acadêmica nessa instituição.

São pessoas especiais que não preciso indicar seus nomes pois elas sabem quem são. Aos meus familiares.

RESUMO

Neste trabalho apresenta-se um manipulador robótico para aplicação em um jogo de xadrez por meio do X.A.P.R.O (Xadrez Autômato Profissional). O braço robótico desenvolvido foi baseado no manipulador 6 DoF (6 Degrees of Freedom - seis graus de liberdade), o qual ele se refere a liberdade de movimento de um corpo rígido em um espaço tridimensional, 3 de translação e 3 de rotação, fornecendo uma relação direta entre a posição do atuador e a configuração do manipulador definido pela sua cinemática direta e inversa. Ele possui seis servo-motores e o Raspberry Pi 3 b+, como controlador, além de uma estrutura mecânica desenvolvida para acoplamento de uma garra. O programa inicial para dar movimentação ao manipulador teve como linguagem adotada o python, dentro do SO (Sistema Operacional) do Raspbian, que tem como base o Debian do Linux. Esse projeto pretende iniciar uma linha de desenvolvimentos subsequentes que poderão culminar futuramente em um sistema completo e inteligente para manipulação de peças, táticas e competições de xadrez.

Palavras-chaves: Mecânica, Programação, X.A.P.R.O

ABSTRACT

This work presents a robotic manipulator for application in a game of chess by means of the X.A.P.R.O (Chess Professional Automata). The developed robotic arm was based on the 6 DoF (6 Degrees of Freedom) 6 manipulator, which refers to the freedom of movement of a rigid body in a three-dimensional space, 3 of translation and 3 of rotation, providing a direct relationship between the position of the actuator and the configuration of the manipulator defined by its direct and inverse kinematics. It has six servo motors and the Raspberry Pi 3 b +, as a controller, plus a mechanical structure developed for coupling a claw. The initial program for handling the manipulator was adopted as python, within the Raspbian Operating System (OS), which is based on Linux Debian. This project aims to initiate a line of subsequent developments that could culminate in the future in a complete and intelligent system for manipulation of chess pieces, tactics and competitions.

Key-words: Mechanics, Programming, X.A.P.R.O

SUMÁRIO

Lista de Siglas

COG – Center of Gravity (Centro de Gravidade)

6 DoF – 6 Degrees of Freedom (6 Graus de Liberdade)

X.A.P.R.O - (Xadrez Autômato Profissional)

I A – Inteligência Artificial

RIA – Robotic Institute of America

RAD Tools – Rapid Apllication Development Tools

ENIAC - Electronic Numerical Integrator and Computer

PEAS - Performance, Environment, Actuators, Sensors

Lista de Figuras

INTRODUÇÃO

Em todo mundo desponta uma variação de natureza econômica, que acontece tanto geograficamente quanto culturalmente. Devido à "nova ordem de reestruturação" em vários setores, principalmente os interesses das empresas de modernizar seus sistemas, e de potencializar suas atividades, a "revolução tecnológica" surge como uma ferramenta no período de "reestruturação do capitalismo". Entretanto, a transformação social e a evolução histórica não são determinadas pelas tecnologias, mas estrategicamente neste período histórico possuem influências e a capacidade de transformação definida pela sociedade, com características específicas de acordo com a instituição, cultura, e a relação ao capitalismo global. (CASTELLS, 1999).

Assim, a robótica é uma dessas revoluções tecnológicas que têm proporcionado transformações no cotidiano da sociedade em geral, sendo de fundamental importância que a Educação possa desenvolver ações que visem contextualizar os educandos nesse novo mundo que se apresenta, principalmente naquelas profissões ligadas diretamente as novas tecnologias.

Nesse sentido, o trabalho proposto acaba também por demonstrar aspectos de todas essas transformações, pois em tempos remotos não se conceberia que um elemento não humano pudesse ser capaz de manipular inteligentemente qualquer tipo de peça, principalmente aquelas que seriam objetos de um jogo envolvendo capacidade motora e raciocínio.

Então, construiu-se um manipulador robótico que possa ter aplicações em jogos de xadrez. Para o desenvolvimento deste projeto, que é intrinsicamente multidisciplinar, trabalha-se a mecânica, com perspectivas à inteligência artificial, para mover uma peça do jogo de xadrez, aqui nomeado de X.A.P.R.O, além da eletrônica, matemática, probabilidade, comunicações, entre tantas outras áreas correlatas. Por ser um braço robótico, sua versatilidade permite também aplicações multiusos.

Seguindo a concepção de Morton, o jogo de xadrez como um autômato não é recente, desde o séc. XVII havia sido projetado uma máquina com o nome de *The Turk* que movia automaticamente as peças sem a interferência humana, desafiando os jogadores a tentarem vencê-lo. Sendo sempre questionado de ser ou não autômato. Quando em 1819 *The Turk* foi levado para a Inglaterra, Charles Babbage viu pela primeira vez, e após três anos iniciou a projeção de uma máquina para calcular e tabular função matemáticas automaticamente

(MORTON, 2015). Como nos escritos de outro autor Tom Standage em *The Turk*:

> "Ao contrário das novas máquinas da revolução industrial, que substitui a atividade física humana, este fragmento do motor de diferença, como *The Turk*, levantou a possibilidade de que as máquinas poderiam, eventualmente, ser capaz de substituir a atividade mental também" (STANDAGE, 2002, tradução nossa).

Percebe-se que naquela era iniciava indícios de avanços para o desenvolvimento da tecnologia com uma máquina analítica capaz de executar as quatro operações (somar, dividir, subtrair, multiplicar), armazenar dados em uma memória e imprimir resultados, o computador.

Surgem também outros projetos ambiciosos do homem combinando as faculdades mentais por meio do uso de modelos computacionais. O reconhecimento da IA (Inteligência Artificial) deu início com o seu primeiro trabalho realizado por Warren McCulloch e Walter Pitts. Os mesmos autores usaram 3 fontes: primeiro o conhecimento de filosofia básica e a função dos neurônios no cérebro; segundo uma análise formal da lógica proposicional devido a Russel e Whitehead; e terceiro a teoria da computação de Turing. Eles propuseram um modelo de neurônios artificiais em que cada neurônio se caracteriza na condição de "ligado" ou "desligado". Para mudança de "ligado" ocorre em resposta a estimulação por um suficiente número de neurônios vizinhos. Admite-se que o estado do neurônio "equivalente em termos concretos a uma proposição que definia seu estimulo adequado". (MCCULLOCH e PITTS, 1943)

Seguindo com o mesmo modelo dois alunos de graduação no departamento de matemática de Princeton, Marvin Minsky e Dean Edmonds, construíram a primeira rede neural de computador em 1951. O SNARC, como chamada, usou 3000 tubos de vácuo para simular uma rede de 40 neurônios. (RUSSEL e NORVIG, 2004).

Nesse contexto, o interesse pelo desenvolvimento de robôs cresce amplamente no meio industrial em estatísticas, os desafios parecem ser diversos para atingir a forma ideal de cada robô, resta ainda, inúmeros questionamentos em torno da criação da utilidade. A insuficiência de informações e pesquisas nesse vasto campo deixa uma gama ilimitada de criatividade para ser desenvolvido e aplicado na sociedade, sobretudo na região do MS.

Entendemos necessário esclarecer os termos que servirão para compor o X.A.P.R.O, para a ideia que expressa à palavra robô, compreendido como máquina que faz o serviço de um ser humano, podendo ou não, assemelhar-se ao um ser vivo. Seguindo a definição da palavra robótica que trata da criação de tudo o que compõem um robô, desde os desenhos básicos para as suas funcionalidades até a programação e o teste de seus funcionamentos.

O termo “robot” usado pela primeira vez por Karel Capek, e mencionado em seus escritos na época de 1920. Na ocasião o mesmo autor tentou definir o que vem a ser a expressão robô, seguindo suas ideias descreve com características principais: um robô não precisa se alimentar, não possui sentimento e faz exatamente o que foi programado sem precisar descansar. O robô não tem uma única definição, segundo *Joseph Engelberger,* o pai da robótica moderna, tenta definir um robô como diz “Eu não posso definir o que é um robô, mas eu reconheço um quando eu vejo um”. (TZAFESTAS, 2014).

Em outro conceito defendido pelo Instituto de Robótica da América (Robotic Institute of America (RIA) define a robótica industrial como sendo um manipulador multifuncional reprogramável e desenhado para mover materiais, peças, partes ou dispositivos específicos por meio de várias programações de movimentos para performances de múltiplas tarefas pela qual adquire as informações a cada momento e responde com movimentos inteligentes. (TZAFESTAS, 2014).

1. Tema - O braço mecânico e suas etapas: mecânica, lógica.
2. Problematização
    - Qual a probabilidade do ensino do jogo xadrez por meio da programação min/máx.
    - Seria possível passar o conhecimento de programação de inteligência artificial através do X.A.P.R.O.
    - A programação min/máx. com poda alpha/beta tem como o pensamento de todas as probabilidades de movimento possíveis ou jogadas possíveis dentro do xadrez, o que acaba envolvendo o raciocínio rápido e o pensamento de menos perda e mais ganho, sempre pensando alguns passos a frente e parte da programação faz com que o pensamento evolua para uma programação de forma gulosa.
    - A possibilidade de transferir o conhecimento de programação de IA vem com o conhecimento básico da matemática e o raciocínio logico e o X.A.P.R.O para motivação.
3. Objetivo Geral
    - Compreender o desenvolvimento das fases da mecânica e programação no X.A.P.R.O.

    Específicos:
    - Conhecer a abrangência da robótica na era contemporânea
    - Experimentar a utilidade da robótica no X.A.P.R.O

- Descobrir a forma de jogar xadrez por meio do manipulador
- Desenvolver a mecânica e a programação para o X.A.P.R.O
- Montar peças de um manipulador compatível ao mecanismo da programação eletrônica, para implantação do jogo do xadrez

4. O trabalho será composto de 5 Capítulos, como segue: Introdução. Capítulo II, o panorama mundial dos robôs. Aponta a maior participação por área de atuação e utilidades dos robôs na atualidade e o desenvolvimento tecnológico nas indústrias: no uso pessoal, entretenimento, doméstico e entre outros. A composição do manipulador. Tornam-se indispensáveis, o cálculo e formula para fazer todo tipo de movimento necessário para se operar em um ambiente de trabalho, sendo utilizado o conhecimento humano para adequar os seus movimentos naquele local e forma. Capítulo III, A origem do xadrez. Desde a sua criação constata que não é somente um jogo de entretenimento, e que não pode faltar o uso do raciocínio para se vencer. Com o avanço da sociedade o xadrez tem nos mostrado elementos importantes tanto para demonstração estratégica como também na parte de inteligência artificial que requer uma fundamentação do que o robô tem que se orientar pelos padrões lógicos. Fundamentos básicos e técnicas competitivas da IA. Fundamentada em várias áreas do conhecimento, entre elas esta teoria do controle e cibernética, engenharia da computação entre outras. A IA pode ser voltada para suprir problemas entre eles alguns dos quais vai ser utilizado no X.A.P.R.O para viabilizar a solução do problema de jogo, fazendo com que a inteligência artificial complemente o que deve ser feito. E finalizando o Capítulo IV, Procedimento metodológico: programação. Será utilizada uma das linguagens de programação de alto nível. Que possui a compatibilidade que se tem com o Raspberry Pi. A linguagem de programação acompanha o desenvolvimento da computação desde o início, mas com o passar do tempo foi ampliando uma grande variedade de linguagem para os seus desenvolvedores. Desenvolvimento do X.A.P.R.O. E o Capítulo V. Consideração Final.

Capítulo II

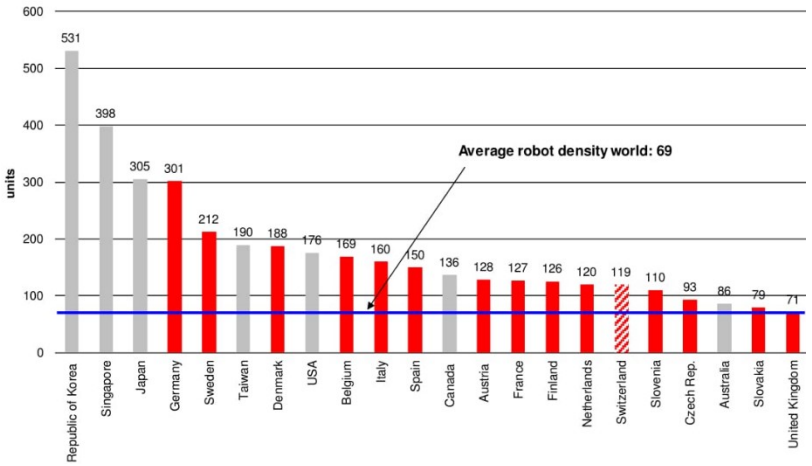
Este capítulo estrutura-se da seguinte forma: com levantamento de informações globais que se utilizam da robótica industrial demonstrando o desempenho da manipulação altamente sofisticada e mobilidade. A visibilidade se dá por meio de gráficos estatísticos pontuando fatores supostamente importantes como os tipos de robôs existentes no mercado. E demonstra a forma de entendimento básico de um manipulador, com toda a movimentação possível, explicando como ele consegue atuar sobre objetos tridimensionais desde a posição e rotação do manipulador até a programação do mesmo.

2.1. - Panorama mundial dos robôs

Supõe-se que na década de 2010 houve uma grande expansão de robôs para todos os tipos de atividades para a sociedade. Os gráficos a seguir representam os tipos de robôs com as suas classificações industrial, profissional e doméstico/entretenimento.

Nas três primeiras barras do gráfico 1 é mostrado a alta densidade de robôs entre os países classificados pelo avanço tecnológico no âmbito mundial, colocando a Coreia do Norte como um dos países que mais possui robôs, seguido da Singapura, o Japão, entre outros países que possuem bastante robôs em várias áreas.

Gráfico 1- Densidade global de robôs



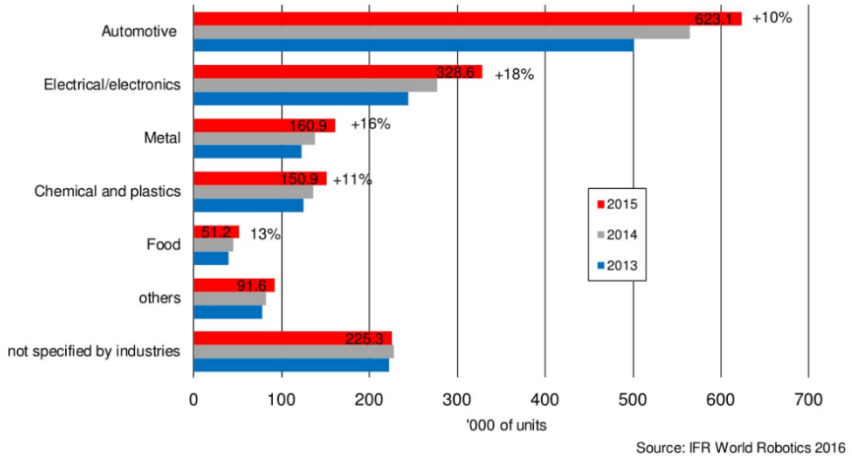
Fonte: (ISIC rev.4: C (International Standard Industrial Classification of All Economic Activities, Rev.4)) (2015)

Analisando o gráfico 1, temos o número de robôs industriais de multipropósito (todos os tipos), por 10.000 empregados na indústria de manufatura. Conforme a linha horizontal

azul onde se pode notar a média de densidade de 69 robôs. Mostra ainda, os países que tem a maior densidade de robôs no mundo entre os primeiros temos a República da Coreia, Singapura e o Japão.

Na primeira barra do gráfico 2, nota-se que na indústria automotiva é uma das áreas em que se utiliza bastante robôs e que é uma das áreas com um crescimento continuo em sua utilização. Essas indústrias são as que mais utilizam robôs em suas linhas de produções, indicando o grande nível de compras de unidades de robôs, o que ajudam a alavancar as produções para ampliar o comercio destes produtos.

Gráfico 2 – Industrias diversas de robôs



Fonte: (IFR(International Federation of Robotics)) (2017)

No gráfico 2, apresenta-se a venda de robôs para indústria automotiva, que aponta um aumento considerável de investimento pelo mundo todo. Entre 2010 e 2015, as vendas de robôs para a indústria automotiva aumentaram em média 10% ao ano (CAGR). Com os investimentos em novas capacidades de produção, bem como os investimentos em modernização da produção nos principais países produtores de automóveis, aumentou o número de instalação de robôs.

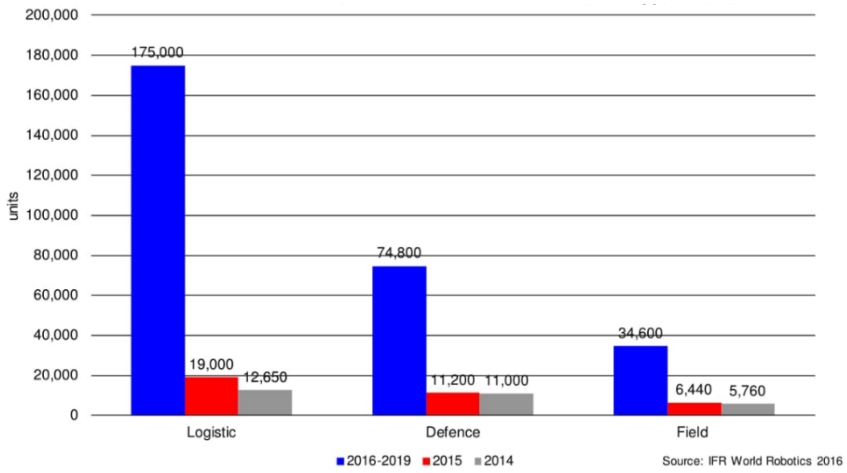
A venda de robôs para a indústria elétrica/eletrônica também expõe uma crescente demanda por eletrônicos e novos produtos, expressando necessidade de automatizar a produção, sendo um dos fatores que impulsionam o aumento de vendas nesta área. Já na área de metais e máquinas apontaram que as vendas em todos os subsetores (metais básicos, produtos metálicos, maquinas indústrias) seguiu uma tendência crescente. Dentro da indústria da borracha e plástico teve um aumento continuo no número de instalação de robôs.

Um robô de serviço ou para uso profissional, esse usado para uma tarefa comercial,

geralmente operado por um operador devidamente treinado. Exemplos de robôs: de limpeza para locais públicos, para entrega em escritórios ou hospitais, de combate a incêndio, de reabilitação e para realizar cirurgia em hospitais. Neste contexto, um operador é uma pessoa designada para iniciar, monitorar e parar a operação pretendida ou sistema de um robô.

Os robôs com as principais aplicações de utilidade profissional, em algumas áreas vêm demonstrando um alto crescimento da automatização, como visto no gráfico 3, na área de logística, defesa, entre outros. Com o grande desenvolvimento e a portabilidade dos robôs, há um aumento no nível de vendas em outras áreas, além das linhas de produção. Conforme o gráfico 3 e 4.

Gráfico 3 - Robôs de serviço para uso profissional

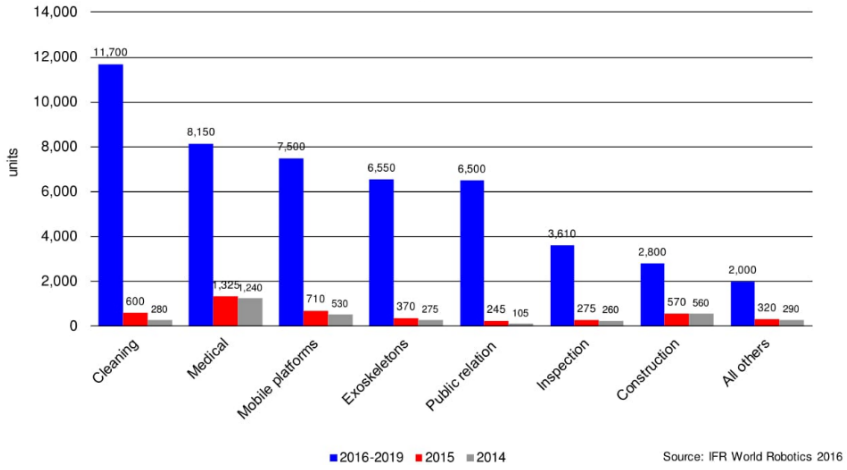


Fonte: (IFR(International Federation of Robotics)) (2017)

No gráfico 3 temos as projeções para o período de 2016-2019 e a quantidade de unidades que já foram vendidas nos anos de 2014-2015. As vendas de sistemas logísticos tiveram um aumento considerável, sendo que são veículos com guia automatizados. Encontra-se ainda, os robôs para a área de defesa e agrícola, mostrando o avanço no desenvolvimento de IA.

O gráfico 4 é uma continuação do gráfico 3, onde aponta a aplicabilidade dos robôs profissionais em diversas áreas, sinalizando que não é somente nas indústrias que vem automatizando, pois existem outras áreas que também ocorre aumento de demanda na agilidade e automatização.

Gráfico 4: Crescimento de venda em vários campos profissionais



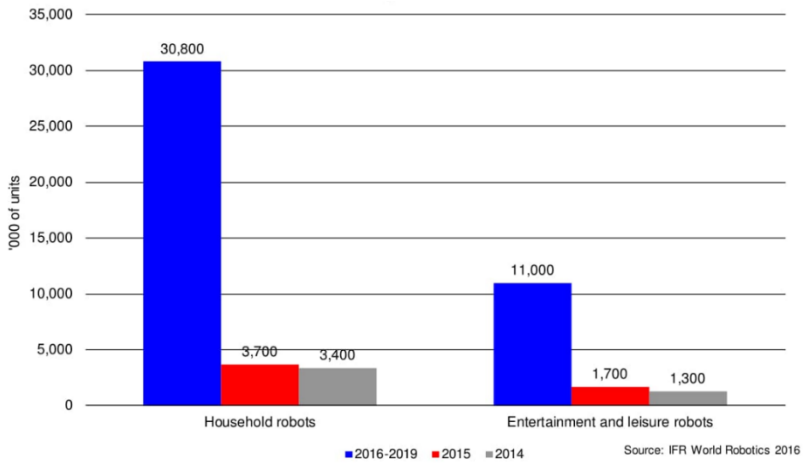
Fonte: (IFR(International Federation of Robotics)) (2016)

No gráfico 4, o robô de serviço para uso profissional teve um grande crescimento, que são os robôs de limpeza profissionais, principalmente os sistemas de limpeza de piso; no setor médico, com os robôs de cirurgia; e no setor de plataformas móveis, com o uso customizável de plataformas de multiuso.

Um robô de serviço pessoal, ou um robô de serviço para uso pessoal ou aquele que faz serviço de uma tarefa não comercial geralmente por pessoas leigas pode ser exemplificado como a cadeira de rodas automática e de assistência de mobilidade pessoal.

Com a grande facilidade na compra de um robô pessoal e o baixo custo, podemos ver no gráfico 5 o crescimento de dados significativos e o desenvolvimento de inteligência artificial que implementam as atividades e suprem as necessidades.

Gráfico 5 - robôs domésticos e entretenimento



Fonte: (IFR(International Federation of Robotics)) (2015)

No gráfico 5 os robôs de serviço para uso pessoal/doméstico como os robôs de limpeza de piso terrão um grande aumento em suas vendas com a previsão de aproximadamente 30 milhões, e em relação aos robôs que cortam a grama com mais 910.000 unidades para os anos de 2016-2019.

Os fornecedores de robô de serviço já estimaram em 2010 um forte aumento de vendas de companheiros de robôs/assistentes/humanoides. No entanto, até agora, não houve vendas significativas de humanoides, como companheiros humanos para realizar tarefas diárias típicas em ambientes de produção, escritório ou doméstico. Algumas empresas japonesas (Honda, Kawada, Toyota e outras) e também as empresas americanas, coreanas e europeias estão no processo de desenvolvimento desses assistentes de robôs de propósito geral, além da faixa de brinquedos e lazer. As primeiras produções de humanoides começaram em 2004, em laboratórios de universidades internacionais como plataformas de pesquisa e desenvolvimento de robótica high-end.

O tamanho do mercado de robôs de brinquedo e sistemas de hobby é previsto em cerca de 8 milhões de unidades, a maioria das quais por razões óbvias, são de baixo preço. Cerca de 3 milhões de robôs para educação e pesquisa deverão ser vendidos no período 2016-2019.

Em suma, a estatística contribui amplamente para compreender o mercado dessa natureza. A densidade de robôs existentes mundialmente nos principais países acentuou a grande expansão, e investimento com relação à tecnologia voltada para as áreas de produção, com utilização principalmente de manipuladores. As indústrias são as que mais se beneficiam dos robôs em suas linhas de fabricação, o que demonstrou o aumento no nível de aceitação de robôs para atividades comerciais. Com o avanço e a facilidade da portabilidade comprova a possibilidade na compra de robô pessoal com baixo custo. Comprova ainda o crescimento de dados significativos e melhoramento do auxílio ao ser humano, às pesquisas, inclusive nas diversas atividades, entretenimentos, enfim onde quer que seja para uso de interesses específicos como no desenvolvimento do X.A.P.R.O.

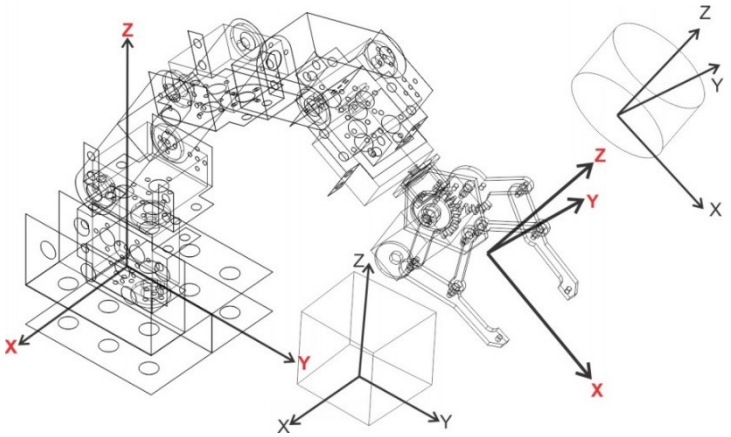
2.2. A composição do manipulador

2.2.1. Posição e orientação

Onde exatamente move as ferramentas com as quais ele lida com outros objetos. Para isso necessita saber a posição e a orientação sendo demonstrado na Figura 1, onde é feito

cálculos matemáticos, que descreve o sistema de coordenadas tanto do manipulador quanto do objeto que se deseja mover.

Figura 1 - Braço robótico com a demonstração do sistema de referências



Fonte: Erbano J. A (2017)

O sistema de coordenadas está estabelecido em um vetor de posição de 3x1, onde contém as informações que lhe definem em um espaço. Os componentes de P^a possuem valores numéricos que mostram a distância no eixo de (a), P^a é considerado uma posição no espaço e pode ser escrito na forma de um vetor como no exemplo da formula 1 a seguir:

$$\mathbf{P^a} = \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix} \qquad \textbf{(1)}$$

Conforme a formula 1. Deve ser escrito no sistema de referência, para a orientação de um corpo no espaço, utilizando o vetor P^a como referência de um ponto fixo do manipulador. As localizações das juntas não estão especificadas no manipulador, até que seja passado a orientação onde será fixado um sistema de coordenadas ao manipulador.

$$R_b^a = [{}^a\mathbf{X}_b \; {}^a\mathbf{Y}_b \; {}^a\mathbf{Z}_b] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad \textbf{(2)}$$

Na formula 2. Deverá ser apresentado um conjunto de três vetores para especificar uma orientação, formando assim uma matriz de 3x3. A demonstração de um ponto de um vetor é transferir uma orientação ao manipulador que deve existir uma matriz.

A posição em que se encontra a base do manipulador pode ser aleatório, no entanto,

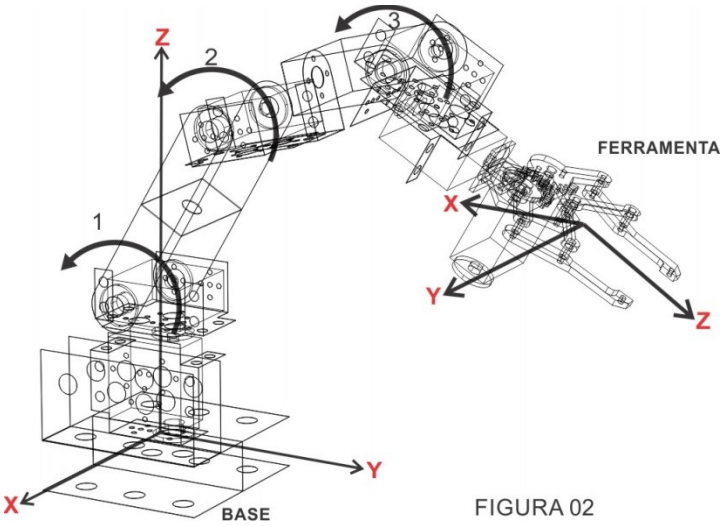
esse ponto escolhido serve como origem do sistema de referência para as outras partes do manipulador. O sistema de referência é composto por quatro vetores que fornece as informações de posição e orientação, e com as informações de coordenadas poderá integrar a um outro sistema de referência existente dentro do manipulador.

2.2.2. Cinemática dos manipuladores

A cinemática dos manipuladores demonstra as propriedades de como deve se mover, geometricamente e quanto tempo ele deve se mover. Os manipuladores possuem elos que estão conectados por juntas, com os quais permitem o movimento dos elos.

O número de graus de liberdade que um manipulador possui abrange o número de variáveis de posição independentes que se tem para se localizar todo o mecanismo. Na ponta livre da cadeia de elos que forma o manipulador fica o efetuador, que varia conforme a aplicação que se pretende com o robô como demonstrado na Figura 2.

Figura 2 – referência da ferramenta em relação a base



Fonte: Erbano J. A (2017)

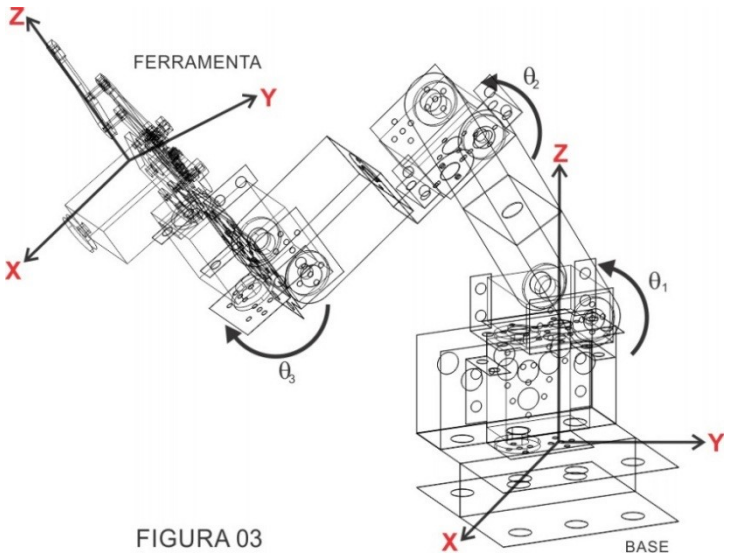
O braço robótico 6 DoF na Figura 2 possuirá o mesmo embasamento das equações cinemáticas que são descritas para o sistema de referência da ferramenta em relação a sua base, sendo esta uma função de variáveis das juntas.

Porém, poderá parecer algumas dificuldades de manipulação mecânica que se refere a cinemática direta, que se trata da geometria estática de computar a posição e o sistema de coordenadas da ferramenta em relação a base do manipulador.

Consente ser a cinemática inversa o cálculo de todos os possíveis ângulos que as

juntas conseguem se mover para se obter a posição e a orientação desejada como na Figura 3. Sendo este um problema não tão simples quanto a cinemática direta, já que as equações cinemáticas não são lineares. Essa solução nem sempre é fácil em uma forma fechada, pois poderá existir ou não uma solução e ao mesmo tempo ter mais de uma solução para se obter o mesmo resultado.

Figura 3 – posição e orientação do sistema de referência



Fonte: Erbano J. A (2017)

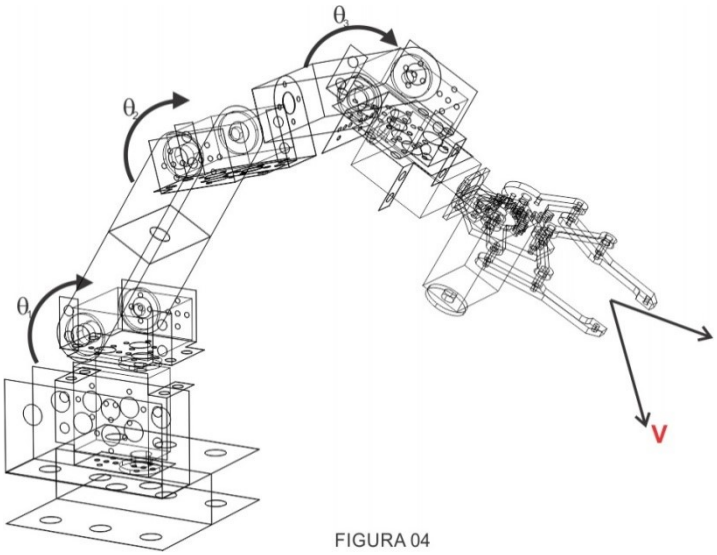
O braço robótico na Figura. 3 demonstra a posição e a orientação do sistema de referência para a ferramenta do manipulador, sendo estes os cálculos para as juntas que serão feitos pela cinemática inversa em relação a base do manipulador.

A existência ou inexistência de uma solução cinemática define o espaço de trabalho de um manipulador. A ausência de uma solução significa que o manipulador não conseguirá chegar à posição e à orientação desejada porque está fora do espaço em que ele alcança. Isso significa que a cinemática direta e inversa faz os cálculos da movimentação que são limitados pela pré definição do braço robótico.

2.2.3. Velocidades e forças estáticas

Para fazer o mapeamento das velocidades de um manipulador deve fazer uma matriz conhecida como jacobiano do manipulador. O jacobiano do manipulador especifica o deslocamento das velocidades nas articulações e as velocidades cartesianas como na Figura 4. Com o mapeamento visualiza a variação da configuração do manipulador. Em alguns pontos, a atribuição não é invertida ou chamados de singularidades.

Figura 4 – relação geométrica entre as articulações e a velocidade



Fonte: Erbano J. A (2017)

Acima a Figura 4 demonstra a relação geométrica entre as proporções das articulações e a velocidade do efetuador final onde pode ser descrito em uma matriz conhecida como o Jacobiana.

Caso o mecanismo funcione bem na maior parte da escala de operação, torna-se menos ideal do que o braço quando se move quase, ou totalmente para cima. Quando necessita localizar um ponto acima do seu eixo provoca um movimento muito brusco sobre o seu eixo do azimute.

Manipuladores nem sempre estão em movimento ás vezes precisa tocar a superfície da peça ou do trabalho e manter uma força estática sobre ele. Com a matriz jacobiana do manipulador torna-se natural solucionar os cálculos que precisa para manter essa força que está sendo aplicado no objeto.

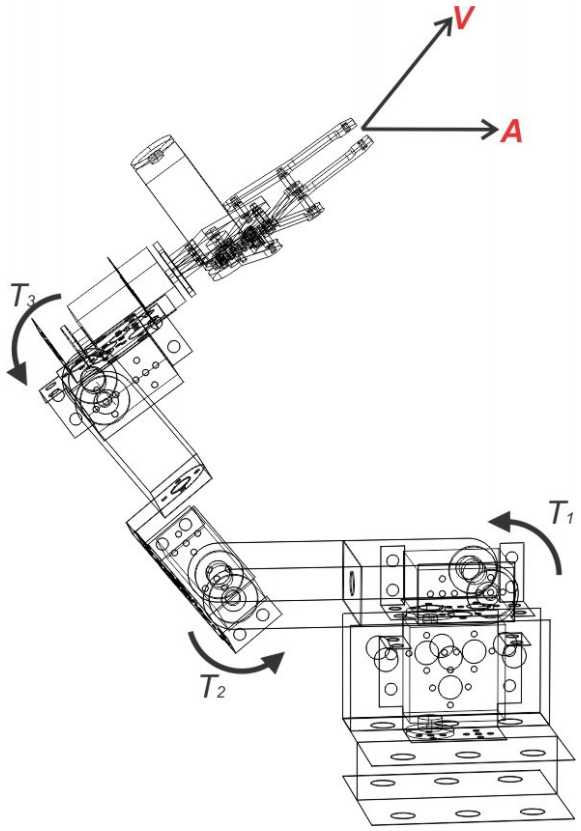
2.2.4. Dinâmica de movimento

Com o propósito de acelerar um manipulador desde o repouso, manter o efetuador com a sua velocidade e desacelerar até parar, é importante um complexo conjunto de funções de torque que se deve aplicar nos atuadores das juntas. Para que um manipulador siga um curso desejado, precisa fazer os cálculos das funções de torque do atuador utilizando as equações das dinâmicas do movimento do manipulador.

A simulação é uma alternativa para a utilização dos cálculos da dinâmica de movimento. E, para que as equações sejam computadas e utilizadas como uma função do

torque do atuador ele possibilita simular como o manipulador poderá se movimentar. Como na Figura 5

Figura 5 – torques aplicados pelos atuadores e o movimento resultante



Fonte: Erbano J. A (2017)

Na Figura 5 mostra a relação entre os torques aplicados pelos atuadores e o movimento resultante do manipulador que pode ser incorporado nas equações de dinâmica de movimento.

A modelagem dinâmica de robôs (manipulador) trata da derivação das equações dinâmicas do movimento do robô. Isso pode ser feito usando duas metodologias a seguir:

- Método de Newton-Euler
- Método Lagrange

A complexidade do método de Newton-Euler é O (n), enquanto a complexidade do método Lagrange só pode ser reduzida até O (n^3), onde "n" é o número de graus de liberdade.

Como a cinemática, dinâmica pode ser distinguida em:

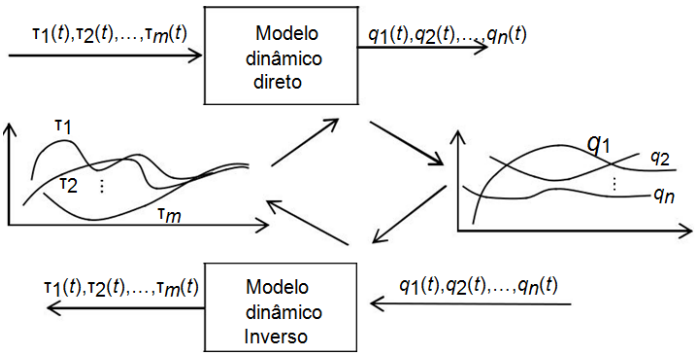
- Dinâmica direta
- Dinâmica inversa

A dinâmica direta fornece as equações dinâmicas que descrevem as respostas

dinâmicas do robô às forças/torques dados τ1; τ2; ...; τm que são exercidos pelos motores.

A dinâmica inversa fornece as forças/torques que são necessárias para obter as trajetórias desejadas dos links do robô. A modelagem dinâmica direta e inversa como ilustrada na Figura 6.

Figura 6 - modelagem dinâmica direta e inversa.



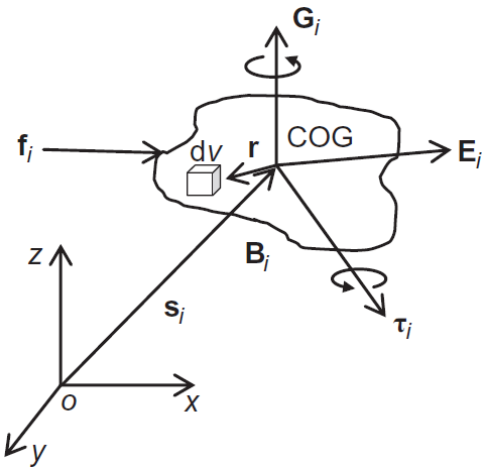
Fonte: Tzafestas S. G. (2014)

No modelo dinâmico inverso, as entradas são as trajetórias desejadas das variáveis de link e emite os torques do motor.

2.2.4.1.Newton-Euler

O modelo deriva-se pela aplicação direta das equações de Newton-Euler para o movimento de translação e rotação. Considerando o objeto B_i pelo qual uma força total se aplica no centro de gravidade (COG). Como na Figura 7

Figura 7 - um corpo sólido B_i e o quadro de coordenadas inerciais O_{xyz}.



Fonte: Tzafestas S. G. (2014)

Então, o movimento de transladação como descrito esse movimento na, formula 3:

$$\frac{dE_i}{dt} = f_i \quad \textbf{(3)}$$

Em, E_i no momento linear dado por, formula 4:

$$E_i = m_i s_i \quad \textbf{(4)}$$

Onde m_i representa a massa do corpo e s_i a posição do COG em relação ao quadro de coordenadas do mundo (inercial) O_{xyz}. Supondo que m_i seja constante, portanto na, formula 5:

$$m_i s_i = F_i \quad \textbf{(5)}$$

Sendo o modelo dinâmico translacional geral.

O movimento rotacional de B_i se descreve por, formula 6:

$$\frac{dG_i}{dt} = \tau_i \quad \textbf{(6)}$$

Em que G_i procede no momento angular total de B_i em relação ao COG, e τ_i o torque externo total que produz o movimento rotacional do corpo. O impulso total G_i é dado por, como na formula 7:

$$G_i = I_i \omega_i \quad \textbf{(7)}$$

Aqui, I_i significa que o tensor de inércia seja dado pela integral do volume, conforme a formula 8:

$$I_i = \int_{V_i} [r^t r I_3 - r r^t] \rho_i dV \quad \textbf{(8)}$$

Onde ρ_i é a densidade de massa de B_i, dV consiste do volume de um elemento infinitesimal de Bi deitado na posição r em relação a COG, ω_i é o vetor de velocidade angular sobre o eixo inercial que passa de COG, I_3 sendo essa matriz de 3x3, e V_i é o volume de B_i.

2.2.4.2. Lagrange

O modelo dinâmico Lagrange geral de um corpo sólido pode ser descrito pela formula 9:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial q}\right) - \frac{\partial L}{\partial q} = \boldsymbol{\tau}, \quad \boldsymbol{q} = [\boldsymbol{q_1}, \boldsymbol{q_2}, \dots, \boldsymbol{q_n}]^T \qquad \textbf{(9)}$$

Onde q_i compreende a variável *i*-ésimo grau de liberdade, sendo τ o vetor de força generalizada externamente aplicado ao corpo (isto é, a força para o movimento, translacional e torque para movimento rotacional), o L representa a função Lagrange definida pela formula 10:

$$\boldsymbol{L = K - P} \qquad \textbf{(10)}$$

Assim o K simboliza a energia cinética total e P a energia potencial total do corpo dada por:

$$\boldsymbol{K = K_1 + K_2 + \cdots + K_n} \qquad \textbf{(11)}$$

$$\boldsymbol{P = P_1 + P_2 + \cdots + P_n} \qquad \textbf{(12)}$$

Onde K_i desempenha a energia cinética do link (grau de liberdade) *i* e P_i retrata a energia potencial. A energia cinética K do corpo, conforme a formula 13:

$$\boldsymbol{K = \frac{1}{2} m s^T s + \frac{1}{2} \omega^T I \omega} \qquad \textbf{(13)}$$

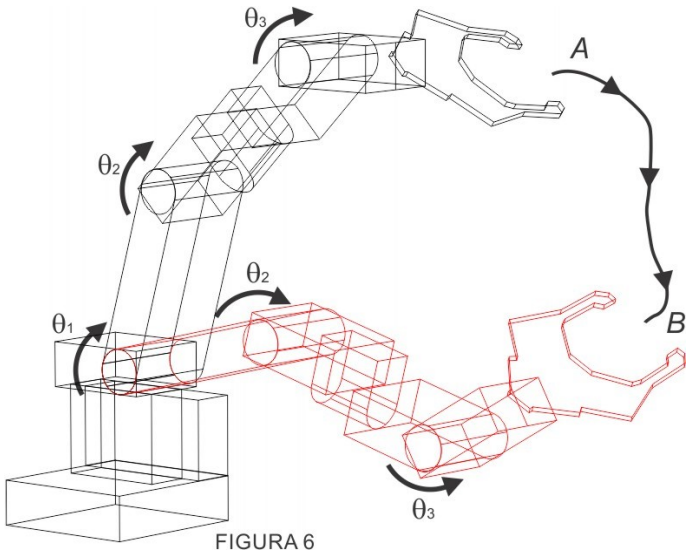
Onde *s* significa a velocidade linear do COG, ω é a velocidade angular da rotação, m representa a massa e I o tensor de inércia do corpo.

2.2.5. Planejamento de trajetórias

Afim de saber o destino desejado e também os pontos intermediários do manipulador, requer computar a trajetória que cada junta deve seguir, após ser computado deve prever os caminhos pelo qual o manipulador deve se mover e assim flexibilizar o movimento das juntas,

como demonstra na Figura 8.

Figura 8 – movimento das juntas através da trajetória



Fonte: Erbano J. A (2017)

Na Figura 8 o planejamento do caminho subentende que é uma capacidade geral incorporada para todos os tipos de robôs inclusive (manipuladores como será aplicado no X.A.P.R.O). Em um sentido amplo e mais detalhado, o planejamento do caminho do robô está preocupado com a determinação de como um robô se moverá e ira manobrar em um espaço de trabalho ou ambiente para atingir seus objetivos.

O problema do planejamento do caminho envolve a computação de um caminho livre de colisão entre uma posição de início e uma posição de objetivo. Muitas vezes, além da evitação de obstáculos, o robô também deve satisfazer alguns outros requisitos ou otimizar determinados critérios de desempenho.

O planejamento do caminho se distingue de acordo com o conhecimento disponível sobre o meio ambiente (ambiente totalmente conhecido/estruturado). Nos casos práticos, o ambiente é apenas parcialmente conhecido, onde o robô, antes do planejamento e navegação do caminho, já possui conhecimento de algumas áreas dentro do espaço de trabalho (áreas que provavelmente colocam problemas de mínimos locais).

A natureza de um obstáculo constata que se descreve através da sua configuração que pode ter uma forma convexa, ou forma côncava, ou ambas. O status de um obstáculo pode ser estático (quando sua posição e orientação em relação a um quadro de coordenadas fixo conhecido é invariante no tempo), ou dinâmicas (quando a sua posição), ou a orientação ou ambas as mudanças em relação à moldura da coordenada fixa mudam.

O planejamento do caminho pode ser local ou global:

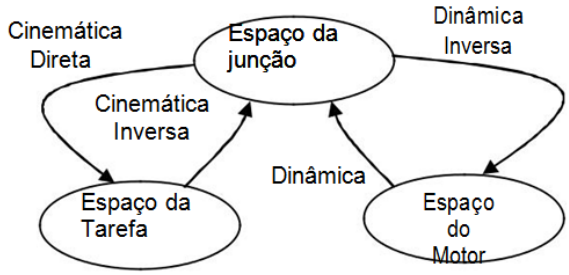
- O planejamento do caminho local é aquele realizado enquanto o robô está em movimento, levando dados de sensores locais. Neste caso, o robô tem a capacidade de gerar um novo caminho em resposta às mudanças do ambiente.
- O planejamento do caminho global pode ser realizado somente se o ambiente é estático e perfeitamente conhecido pelo robô. Nesse caso, o algoritmo de planejamento do caminho produz um caminho completo desde o ponto de partida até o ponto de objetivo antes do robô iniciar seu movimento.

O planejamento do movimento: tem como o processo de seleção de um movimento e as entradas correspondentes, de modo a assegurar que todas as restrições (evitação de obstáculos, prevenção de riscos, etc.) sejam satisfeitas. Ele pode ser considerado como um conjunto de cálculos que fornecem sub-finais ou pontos de ajuste para o controle do robô. Esses cálculos e os planos resultantes são baseados em um modelo adequado do robô e do ambiente em que faz o movimento.

O movimento de um robô pode ser descrito em três espaços diferentes como demonstrado na Figura 9:

1. Tarefa ou espaço cartesiano
2. Espaço das articulações (motores)
3. Espaço dos atuadores (junção)

Figura 9 – relações dos espaços de tarefas



Fonte: Tzafestas S. G. (2014)

A ilustração da Figura 9 expõe a relação dos espaços. Seguindo com a descrição do movimento no espaço da tarefa, na especificação de um ponto de referência, em que

geralmente se faz no quadro de coordenadas globais cartesianas.

A especificação final do efetuador ou a posição do robô móvel nem sempre é suficiente para determinar as posições de todos os links. Por esta razão, usamos o espaço das juntas, que é o produto cartesiano da distância permitida de todos os graus de liberdade (geralmente um subconjunto de n).

Finalmente, para cada movimento do robô que seja compatível com as restrições cinemáticas e dinâmicas, deve existir pelo menos um conjunto de forças/torques que produzem esse movimento. As forças/torques dos atuadores (motores) que geram todos os movimentos permitidos definem o espaço dos motores.

Um problema básico no planejamento do movimento do robô é a presença de obstáculos e o requisito resultante para encontrar um caminho livre de obstáculos. Este é o problema do planejamento do caminho. A necessidade de planejamento de movimento vem do fato de que há um número muito grande de movimentos através dos quais o robô pode atingir uma meta e executar uma tarefa desejada. Além disso, para um determinado movimento, pode haver mais de uma entrada (forças/torques) dos motores que produzem o movimento desejado.

O planejamento de tarefas envolve três fases:

a) Modelagem mundial: um modelo mundial deve envolver uma descrição geométrica de robôs, e objetos no ambiente (que geralmente é incorporado em um sistema de CAD), uma descrição física de objetos (por exemplo, massa e inércia de peças), uma descrição cinemática do corpo e ligações do robô e uma descrição dos recursos do robô (por exemplo, limites de juntas, aceleração máxima ou permitida e características do sensor).

b) Especificação da tarefa: o planejador de tarefas geralmente recebe as tarefas como uma sequência de modelos do estado mundial em várias etapas da execução da tarefa. Na verdade, uma especificação de tarefa é um modelo do mundo acompanhado de uma sequência de mudanças nas posições dos componentes do modelo.

c) Síntese do programa de robôs: esta é a fase mais importante para que o planejador de tarefas seja bem-sucedido. O programa sintetizado deve incluir comandos de

compreensão, propriedades de movimento, comandos de sensores e testes de erro. Isso implica que o programa deve estar na linguagem de programação no nível do robô.

2.2.6. Controle da força

O controle de posição auxiliado pelo controle de força ajuda a calcular a força para pegar a peça sem danificá-la, para isso é importante saber o limite de força no momento em que for fazer o contato com as peças, objetos ou superfícies, como na Figura 10.

Figura 10 – Controle de posição auxiliado pela força

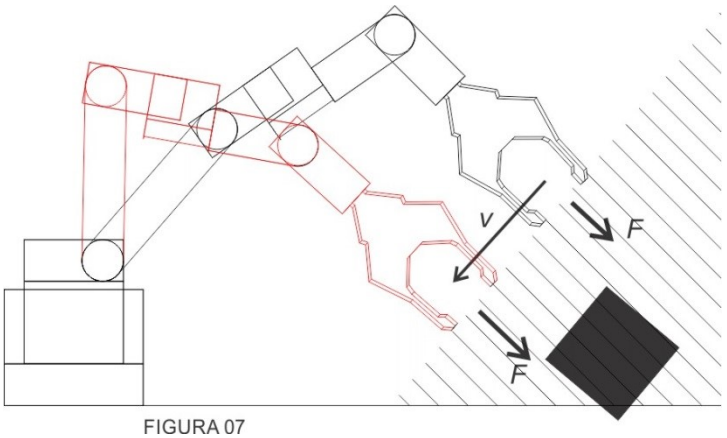


FIGURA 07

Fonte: Erbano J. A (2017)

Na Figura 10 as tarefas que o manipulador possui se dividem em sub-rotinas, que acontece desde mover o efetuador e pegar o objeto dentro da sua área de trabalho. Estas sub-rotinas é associado a restrições naturais, que resultam na mecânica e na geometria especifica das tarefas. Quando as situações são muito complexas as tarefas podem ser divididas em sub-rotinas com as quais podem ser identificados como restrições naturais e restrições adicionais.

Cada tarefa possui uma referência de restrição que é extremamente relevante para a execução, o objeto de trabalho pode ser fixo ou móvel, também deve ser indicado as restrições de posição do objeto assim evitando que ocorra colisões com outros objetos.

O hibrido de força e posição são essenciais para solucionar três problemas como consente o (CRAIG, 2012):

a) A posição do manipulador ao longo de direções nas quais existe uma restrição natural de força.
b) Controle de força do manipulador ao longo de direções nas quais existe uma restrição natural de posição.

c) Um esquema para aplicar a combinação arbitrária desses modos ao longo dos graus ortogonais de liberdade de um sistema de referência arbitrário.

A detecção de força permite que um manipulador detecte o contato com uma superfície, usando a sensação do toque, realizando alguma ação após o mesmo.

2.2.7. Programação

Por meio da linguagem de programação interage a comunicação entre o usuário e o robô. Os manipuladores robóticos são diferentes dos autômatos fixos pois, é possível não somente programar os movimentos que se faz, como utiliza-se dos sensores e a comunicação para se adaptar as tarefas que lhe são passadas.

O usuário deverá propor o movimento, orientando o deslocamento do qual deverá seguir os pontos de passagem definidos por um sistema de coordenada. E, as coordenadas da base do robô são utilizadas como referência, obtendo assim um ponto operacional e um limite de área de atuação.

Para o desenvolvimento do estudo em questão foi de suma importância seguir os passos que foram mencionados nos itens de 1 ao 7 para saber a real fundamentação de um manipulador com todas as ferramentas com as quais ele lida com objetos necessitando saber a posição e a orientação. E de como é feito os cálculos matemáticos, da cinemática dos manipuladores demonstrando as propriedades de como deve se mover, geometricamente e quanto tempo ele deve se mover. Com a utilização da matriz jacobiana é possível fazer o mapeamento da velocidade do X.A.P.R.O.

Entendendo, que a complexidade do conjunto de funções do torque que se deve aplicar nos atuadores das juntas, tendo em mente que o X.A.P.R.O siga o curso desejado e também os pontos intermediários. Necessita fazer os cálculos de posição de cada junta para consegui-lo mover com o auxílio do controle de força, e delimitar a movimentação para não danificar as peças do xadrez.

Capítulo III

Este capítulo estrutura-se da seguinte forma: na primeira parte é feito uma busca no plano da inteligência artificial, trazendo pontos em que se assemelham no uso do raciocínio durante uma partida de xadrez, utilizando a busca aleatória, profundidade e outros elementos. E na segunda parte pontua-se brevemente sobre o sistema de inteligência artificial e as áreas de conhecimento para as futuras complementações ao X.A.P.R.O, tendo uma noção precisa para a compreensão da racionalidade e suas funcionalidades, voltado para inteligência artificial de jogos.

3.1. Habilidade e estratégia no jogo de xadrez

O jogo de Xadrez demonstrou que desde o seu início nada mais é que uma partida de julgamento de poder mental e habilidade intelectual. Com as nossas atualizações modernas, o jogo tendeu a encurtar o tempo de duração sendo, “uma consumação”, às vezes, “devendo ser devidamente desejado”, para pôr em prática os vários tipos de estratégias que a mente humana consegue pensar em uma partida.

Afirmando o processo para a busca de uma solução visualiza-se de imediato trajetos no espaço do problema, procurando o estado objetivo ou o estado desejado. Os pesquisadores, (CHI; GLASSER, 1992) relatam estratégias em seus estudos para a realização da busca como:

Busca aleatória: sendo bom para problemas pequenos, com ineficiência para a maioria dos problemas, Exemplo: o jogo do xadrez, que possui uma busca exponencial exigindo uma busca mais profunda para encontrar a solução.

Busca sistemática: na primeira busca a profundidade procura até o fim de um trajeto. Caso não seja o estado objetivo, sobe um nível e reinicia novamente por um trajeto não explorado. Esse método aplicável para problemas simples, e não os complexos.

Análise de meios e fins: busca por boas alternativas onde levem ao estado objetivo. Possuindo a ideia de achar a diferença que existe entre o estado atual e o desejado e então procurar pelo caminho que reduz. Poderá haver mais de um caminho, naquele que possui a diferença mais ampla aplicado em primeiro lugar para encontrar o melhor meio de chegar ao objetivo. O método de análise de meio e fins, são conhecidos também como métodos heurísticos, onde se reduz o número de alternativas, não garantindo sucesso sempre. Sub-objetivos: faz a escolha de um estado intermediário em direção a solução, alcançando assim

um objetivo temporário. A estratégia de busca divide o problema em dois ou mais subproblemas, fazendo com que todo o espaço de busca se torne de menor profundidade.

A forma como se calcula as possibilidades é útil para a solução de um problema qualquer. Dentro do xadrez, a quantidade é baseada em todos os movimentos das peças dentro do tabuleiro, contando os que são favoráveis ou não.

O plano é consistido em uma forma de hierarquia com operações de *feedback*, onde a cada etapa se testa para ver se está no estado desejado. Onde é definido por um conjunto de planejamento, com as atividades que antecipam e regulam todo o comportamento. Para o planejamento depende do ambiente, das possíveis soluções, e as estratégias que consigam solucionar os problemas. Seguindo com o que foi planejado, considerando como uma árvore de possibilidades e possíveis soluções.

O planejamento descrito por (HAYES, 1989) consiste em três etapas:

1- Representação da tarefa real em um ambiente de planejamento.
2- Exploração do ambiente de planejamento para tentar solucionar o problema.
3- Selecionar um caminho para a solução.

Em (DE LISI,1987) conceitua o plano como duas conotações principais:

1- Os aspectos representacionais onde podem ser comunicados de uma pessoa a outra.
2- Os aspectos funcionais e comportamentais de um plano, como a organização e o controle do comportamento durante o mesmo como o mesmo autor menciona:

> *"*Os comportamentos estratégicos, como os necessários para resolver problemas que surgem em jogos como o xadrez ou a ponte de contrato, são exemplos de planos de tipo 3. Exceto para novatos que não estão familiarizados com as regras e o propósito do jogo, o xadrez, por exemplo, é jogado antecipadamente antecipando e evocando mentalmente uma série de movimentos projetados por cada lado por sua vez. As várias sequências são comparadas e avaliadas para determinar o melhor movimento seguinte. Um movimento real é feito somente após esse procedimento mental ser conduzido. Se o oponente fizer uma das várias respostas antecipadas, o plano pode continuar. Se o oponente fizer uma resposta inesperada, um planejamento adicional pode ocorrer antes de um segundo movimento real ser feito. As limitações nas fases de geração e monitoramento do planejamento de xadrez, ou seja, uma hierarquia sem fim de movimentos possíveis preconcebidos, vem não apenas das limitações da memória humana, mas também das regras do próprio jogo*. " (DE LISI, 1987).*

Percebe-se que em tempos atuais além do jogo do xadrez ser um entretenimento de concentração mental, ele testa constantemente as possibilidades de buscas de estratégias. Um pouco diferente de quando surgiu o jogo, lá as intenções testavam as possibilidades de

combates nas guerras de poderes entre os estados como o próprio jogo demonstra. Podemos verificar através dos tempos algumas modificações e avanços com a tecnologia.

Na origem do xadrez, em estudos de (FORBES, 1860) tudo indica com evidência histórica que foi inventado na Índia, e logo introduzido na Pérsia em outras regiões asiáticas durante o século VI. Mas não tendo uma única fonte confiável que tende a provar que ela já foi inventada ou conhecida em qualquer outro país anteriormente.

Se investigarmos calmamente em fatos tão simples como chegar ao nosso alcance, reservando todos os preconceitos ingênuos e parciais, acharemos que a história do xadrez cai sob três períodos distintos:

O primeiro é o antigo jogo hindu chamado Chaturanga, no qual os movimentos e poderes de todas as peças empregadas foram o mesmo que prevaleceu na Ásia e na Europa, até o final do século XV. A origem desta forma de jogo foi perdida nas profundezas da antiguidade. Mas não pode haver dúvida, que foi inventado na Índia. O tabuleiro consistia em sessenta e quatro quadrados. O jogo era desempenhado por quatro pessoas, cada um com um Rei, uma Torre, um Cavaleiro e, um Bispo (onde era representado por um Navio) junto com quatro Peões. Os dois jogadores opostos se tornaram aliados contra os outros dois, e os movimentos eram decididos pela virada de um dado oblongo com quatro faces marcadas com os números dois, três, quatro e cinco.

A única peculiaridade neste jogo primordial era que o rei poderia ser capturado, bem como qualquer outra peça. Isso, devemos lembrar, era simplesmente o xadrez em sua infância, e a captura do rei estava certamente em conformidade com os usos da guerra real, embora possamos considerá-lo como tendo uma tendência a estragar o jogo. A simplicidade e a imperfeição deste Xadrez primitivo fornecem as melhores provas possível de ser o original. Sua criação pode ter sido de três a quatro mil anos antes do século VI.

O segundo, ou período mediano, na história do Xadrez, ocupa um espaço de cerca de mil anos, do século VI ao século XVI. No início deste período, a melhoria feita no jogo é muito decidida. O tabuleiro e os poderes das peças ainda permanecem os mesmos, mas as duas forças aliadas estão unidas em um lado do quadro, enquanto os adversários fizeram o mesmo no outro. Um dos reis aliados se torna uma peça subordinada, chamado pelos persas e pelos árabes Farzin ou, conselheiro ou ministro, com apenas metade do poder que já possuía anteriormente como soberano independente. Ao mesmo tempo, a Torre e o Bispo mudam de lugar, o primeiro foi transferido para o canto do conselho e o Bispo para o lugar que ele agora ocupa. Finalmente, o dado é descartado.

O terceiro, ou período moderno, começa no século XVI. A mudança feita aqui consiste, primeiro, em estender o poder do Bispo, permitindo que ele comande toda a diagonal, em vez de cada terceiro quadrado, como antigamente; Em segundo lugar, ao dar à Rainha ou Ministro o enorme poder da Torre e do Bispo combinados; E, por último, ao permitir que os Peões avancem um ou dois quadrados, no primeiro movimento. A essas melhorias, podemos acrescentar o de lançar o Rei, de acordo com o método italiano ou o da escola anglo-francesa. É provável que nossa posteridade em andamento introduz algumas modificações adicionais, por exemplo, dar à Rainha o poder adicional do Cavaleiro.

Com o levantamento das informações obtidas do jogo do xadrez e partes da inteligência artificial obteve-se o embasamento necessário para se utilizar a mesma forma de raciocínio lógico que ocorrem em uma decisão na inteligência artificial, e explicação de como as peças do jogo do xadrez se movem, para o X.A.P.R.O, do qual poderá ganhar alguns princípios de processamento e regras para fazer os seus movimentos.

3.2. Fundamentos básicos, técnicas competitivas da IA e aplicações a jogos

3.2.1. Compreendendo o processo da Inteligência Artificial

De acordo com os especialistas da área as definições variam ao longo de duas principais dimensões. De modo geral preocupados com processos de pensamento e raciocínio e outros ao comportamento. Algumas definições que serão citadas logo a seguir, pautada no sucesso em termos de fidelidade para a performance humana, enquanto outras medem contra um conceito ideal de inteligência, que foram chamados de racionalidade. Ou seja, entende-se por racional se ele faz a "coisa certa", que se baseia no raciocínio.

Tabela 1. Propostas de sistemas voltados para o lado humano e racionalidade

Sistemas que raciocinam e atuam como humanos:	**Sistemas que raciocinam e atuam com racionalidade**
"O excitante novo esforço para fazer computadores pensarem … máquinas com mentes, no sentido literal e completo." (Haugeland, 1985)	"O estudo de faculdades mentais por meio do uso de modelos computacionais. " (Charniak e McDermott, 1985)

"[A automação de] atividades que nós associamos com pensamento humano, atividades tais como tomada de decisão, resolução de problemas, aprendizagem..." (Bellman, 1978)	"O estudo de computações que fazem possível perceber, raciocinar e agir. " (Winston, 1992)
"A arte de criar máquinas que realizam funções que requerem inteligência quando realizadas por pessoas. " (Kurzweil, 1990)	"Inteligência Computacional é o estudo do projeto de agentes inteligentes. " (Poole et al., 1998)
"O estudo de como fazer computadores realizarem coisas em que, no momento, pessoas são melhores. " (Rick e Knight, 1991)	"IA ...é preocupada comportamento inteligente em artefatos. " (Nilssom, 1998)

Fonte: Russell e Norvig (2004)

Na tabela 1 aparenta ter uma tensão entre as propostas que são voltadas na área para humanos que é uma ciência empírica, que envolve hipóteses e experimentações para a comprovação, e as de racionalidade requer uma combinação da matemática e da engenharia para solucionar os problemas. (RUSSELL; NORVIG, 2004).

3.2.2. Fundamentação da Inteligência Artificial

A IA se fundamenta em diversas áreas do conhecimento, mas nesse trabalho será demonstrado a seguir algumas áreas que supõe ser de interesse para o desenvolvimento do X.A.P.R.O. ao que se refere engenharia de computação, teoria do controle e cibernética e linguística da programação apoiado no especialista da área como RUSSELL; NORVIG, (2004).

a- Engenharia de Computação

O primeiro computador eletrônico, o ABC, foi montado por volta de 1940-1942 na UNIVERSIDADE ESTADUAL DE IOWA, sem muito sucesso. Outros projetos que promoveram ser o precursor do computador moderno com mais influência sendo o ENIAC (Electronic Numerical Integrator and Computer), que foi desenvolvido como parte de um projeto militar secreto na universidade de Pennsylvania. (RUSSELL; NORVIG, 2004).

No sec. XIX, desde então, cada geração de hardware de computador tem trazido um

aumento na velocidade e capacidade e preço reduzido. “A performance vem dobrando a cada 18 meses ou mais”. (RUSSELL; NORVIG, 2004).

A máquina analítica foi de longe a mais ambiciosa: pois ela incluía memória endereçável, programas armazenados e pulos condicionais (mover valores), e foi a primeira capaz de realizar computação universal criada por Charles Babbage. A colega de Babbage, Ada Lovelace, foi talvez a primeira programadora do mundo (a linguagem de programação Ada foi nomeada depois dela). (RUSSELL; NORVIG, 2004).

A IA deve à área de software da ciência da computação, a qual lhe forneceu os sistemas operacionais, as linguagens de programação e as ferramentas necessárias para escrever programas modernos. Mas, essa área de IA também veio favorecer mais tarde para o trabalho de precursora em muitas opiniões que foram utilizadas na ciência da computação em geral. (RUSSELL; NORVIG, 2004).

Citando algumas como “time sharing, interpretadores interativos, RAD Tools” (Rapid Application Development Tools), tipos de dados de listas encadeadas, gerenciamento automático de armazenamento e conceitos chaves de programação simbólica, funcional, dinâmica e orientada a objetos. (RUSSELL; NORVIG, 2004).

b- Teoria do controle e cibernética

A figura central na criação daquilo que se conhece hoje como teoria de controle destacando Norbert Wiener um matemático brilhante que trabalhou junto com Bertrand Russell, entre outros. Antes de se interessar por sistemas de controle biológico e mecânico e sua conexão com a cognição. Eles viram o comportamento consciente como o resultado de um mecanismo regulador tentando minimizar o erro da diferença entre o estado atual e o estado do objetivo. (RUSSELL; NORVIG, 2004).

A teoria de controle, em especial o ramo conhecido como controle estocástico ótimo, tem o objetivo do projeto de sistemas que maximizam uma função sobre o tempo. Correspondendo aproximadamente a nossa visão da IA: projetando sistema que comportem de maneira ótima. Sendo a IA e a teoria do controle campos diferentes residindo no acoplamento estrito entre as técnicas matemáticas familiares aos participantes e os conjuntos de problemas correspondentes, que foram incluídos em cada visão do mundo. (RUSSELL; NORVIG, 2004).

O cálculo e a álgebra de matrizes, tendo como ferramenta a teoria de controle, sendo adequado para sistemas que podem ser descritos por conjuntos fixos de variáveis continuas. A IA supostamente criada como um meio de escapar das limitações da matemática da teoria de controle. As ferramentas de inferência lógica e computação permitiram que os pesquisadores da IA considerassem alguns problemas como linguagem, visão e planejamento que ficavam completamente fora do campo de ação da teoria de controle. (RUSSELL; NORVIG, 2004).

c- Linguística da programação

A linguística moderna e IA, entende-se uma intersecção em um campo híbrido chamado linguística computacional ou processamento de linguagem natural ambas da mesma época. Entender linguagem requer uma compreensão do assunto e do contexto, não apenas tentar entender da estrutura das sentenças. Muito dos trabalhos anteriores em representação do conhecimento foi vinculado à linguagem e informado pesquisas em linguística, ligado a décadas de trabalho de análise filosófica de linguagem. A questão de se entender linguagem até a década de 1960 foi pouco apreciado. (RUSSELL; NORVIG, 2004).

3.2.3. **Agentes Inteligentes:** agentes e ambientes

Compreende-se por um agente aquele que pode perceber seu ambiente através de sensores e operar no ambiente através dos atuadores. Um agente humano tem olhos, orelhas como sensores, e mãos, pernas, boca como atuadores. Um agente robô pode ter câmeras e localizador infravermelho como sensores, e vários motores como atuadores. (RUSSELL; NORVIG, 2004).

Uma vez que um agente de software recebe as entradas pelo teclado, ou o conteúdo de arquivos, pacotes de rede como sensores de entrada e atua no ambiente mostrando resultados em uma tela, gravando arquivos e enviando pacotes pela rede. (RUSSELL; NORVIG, 2004).

Quando um agente escolhe uma ação ele depende do histórico de percepções. Matematicamente fala-se que um comportamento de um agente se descreve por uma função que mapeia qualquer sequência de percepções para cada ação. (RUSSELL; NORVIG, 2004).

Pode-se tabular uma função que descreve um agente; sendo esta uma tabela muito grande, a não ser que se coloque um limite no tamanho de sequências de percepções que se deseja considerar. Para construir uma tabela de um agente tenta-se todas as sequências de

percepções possíveis e grava as ações que o agente responde. A tabela é uma característica externa do agente, enquanto que internamente a função é implementada por um programa. A distinção de ambas as ideias é que o agente pode ser descrito pela matemática abstrata e o programa é uma implementação concreta, que roda na arquitetura. (RUSSELL; NORVIG, 2004).

A performance incorpora o critério de sucesso do procedimento do agente, gerando uma sequência de ações de acordo com as percepções que recebe, se for a sequência desejável ele desempenha um bom trabalho, claro não existe uma medida fixa para todos os agentes. (RUSSELL; NORVIG, 2004).

A natureza dos ambientes

Especificando o ambiente de tarefas

Necessita especificar a medida de performance, o ambiente, os atuadores e os sensores do agente, onde agrupa-se sobre o título de ambiente de tarefas, que se define como a sigla PEAS (Performance, Environment, Actuators, Sensors) sendo a descrição do ambiente, onde o primeiro passo precisa especificar o ambiente tão completo quanto possível. (RUSSELL; NORVIG, 2004).

Propriedades dos ambientes de tarefa

As faixas de ambientes que podem surgir em IA são vastas. Podendo ser identificados um número de dimensões em que são categorizados:

- **Totalmente observável x parcialmente observável (acessível x inacessível)**: o sensor do agente tem acesso completo ao estado do ambiente o tempo todo, observando todos os aspectos relevantes para escolher uma ação e executar, o ambiente deverá ser completamente observável, se importando com as medidas de performance. Um ambiente pode ser parcialmente observável devido a ruído ou sensores não aperfeiçoados ou parte do estado do ambiente que esteja faltando. (RUSSELL; NORVIG, 2004).

- **Determinístico x estocástico (determinístico x não determinístico)**: Se o próximo estado do ambiente for completamente determinado pelo estado atual e suas ações executadas pelo agente, o ambiente se considera como determinístico, ou seja, previsível. O ambiente considerado determinístico, quando não conta com as ações de outros agentes, chamado como ambiente estratégico. Em princípio, um agente não precisa se preocupar sobre incertezas em um ambiente determinístico por ser completamente observável. Por outro lado, se o ambiente for parcialmente observável, ele pode ser estocástico. Portanto, ele pode conter variações imprevisíveis que incluem elementos que define como estocásticos. (RUSSELL; NORVIG, 2004).
- **Episódico x sequencial (episódico x não episódico)**: um ambiente episódico, a experiência do agente se divide em episódios atômicos. Cada episódio consiste na percepção e realização de uma única ação. Crucialmente, o próximo episódio não depende das ações que foram feitas em episódios anteriores, sendo que a escolha de uma ação depende somente do próprio episódio. Exemplo: um agente para apontar peças defeituosas em uma linha de montagem baseia cada decisão somente na peça atual, não se preocupando com as decisões anteriores, o que não afeta nas decisões futuras. (RUSSELL; NORVIG, 2004).

  Em ambientes sequenciais, a decisão atual pode afetar todas as decisões futuras. Exemplo: o xadrez e um motorista de táxi são sequenciais: em ambos os casos, as ações de curto prazo podem ter consequências de longo prazo. Ambientes episódicos são mais simples porque o agente não tem que pensar a frente, não tem que pensar nas consequências futuras de suas ações, apenas nas consequências da ação atual. (RUSSELL; NORVIG, 2004).
- **Estático x dinâmico**: o ambiente pode ser alterado enquanto o agente está deliberando, sendo dito como dinâmico, e estão continuamente perguntando ao agente o que ele quer fazer; se ele não decidiu ainda, isto conta como se tivesse não decidido fazer. (RUSSELL; NORVIG, 2004).

  Ambientes estáticos são fáceis de lidar pois o agente não necessita observar o mundo enquanto toma uma decisão para uma ação, que vai executar, e nem precisa se preocupar com a passagem do tempo. (RUSSELL; NORVIG, 2004).

  Se o ambiente não muda com a passagem do tempo, mas a performance do agente sim, então se diz como semi-dinâmico. Exemplo: Dirigir um táxi se considera como dinâmico. O xadrez, quando se utiliza um relógio em uma partida, se vê como

semi-dinâmico. Jogos de palavras cruzadas são estáticos. (RUSSELL; NORVIG, 2004).

- **Discreto x continuo**: Um ambiente discreto, possui um conjunto finito de estados, percepções e ações. Exemplo: o xadrez. O ambiente continuo não possui um problema da incerteza da quantidade de estados, ações e tempo. Exemplo: Dirigir um táxi. (RUSSELL; NORVIG, 2004).
- **Agente único x multi agente**: Um agente único possui somente apenas um agente e nenhum outro para competir. Exemplo: jogo de palavras cruzadas. Multi agente possui mais de um agente ou mais no ambiente, criando uma competição. Exemplo: o xadrez. (RUSSELL; NORVIG, 2004).

  Existe o ambiente de multi agente parcialmente cooperativo, parcialmente competitivo. Exemplo: dirigir um táxi, onde se evita colisões maximizando a performance de todos os agentes, e competindo por passageiros. (RUSSELL; NORVIG, 2004).

  Demonstrativo de propriedades e tarefas de um agente ou ambiente:

Tabela 2. Ambiente de atuações

Ambiente de tarefa	**Observável**	**Determinístico**	**Episódico**	**Estático**	**Discreto**	**Agentes**
Palavras cruzadas	Completamente	Determinístico	Sequencial	Estático	Discreto	Único
Xadrez com um relógio	Completamente	Estratégico	Sequencial	Semi-Dinamico	Discreto	Multi
Pôquer	Parcialmente	Estratégico	Sequencial	Estático	Continuo	Multi

Fonte: Russell e Norvig (2004)

A Tabela 2 exemplifica em alguns ambientes de atuações de agentes resumindo as explicações acima colocada pelo autor.

A estrutura de agentes

A inteligência artificial projeta o programa do agente, que logo implementa a função de mapear as percepções para as ações. Assumindo que o programa rodará em um tipo de

dispositivo computacional com sensores físicos e atuadores, sendo chamado de arquitetura: (RUSSELL; NORVIG, 2004).

agente = arquitetura + programa.

O programa que for escolhido deve ser apropriado para a arquitetura, que pode ser um PC, câmeras e outros sensores. A arquitetura faz as percepções com o programa utilizando os sensores, onde executa o programa e passam para as escolhas das ações, que depois para os atuadores. (RUSSELL; NORVIG, 2004).

Programas de agente

Os tipos básicos de programas de agentes, que incorporam os princípios subjacentes de quase todos os sistemas inteligentes mencionadas pelo autor RUSSELL; NORVIG, (2004).

- **Agentes reativos simples**: O tipo mais simples de agente, que seleciona suas ações com base na percepção atual, ignorando o resto do histórico de percepções. Exemplo, dirigir um carro. Se a luz de freio do carro da frente acender, será dado início a frenagem. Nós chamamos tal conexão de regra de condição-ação. (RUSSELL; NORVIG, 2004).
- **Agentes reativos baseados em modelos**: A forma mais efetiva de gerenciar ambientes parcialmente observáveis está em manter estados internos que dependem do histórico das percepções, tendo informações da área parcialmente observável. Atualizando informação com o tempo requer dois tipos de conhecimento: (RUSSELL; NORVIG, 2004).

  Primeiro, precisa-se de informações de como o mundo progride, independentemente do agente. O segundo, precisa de algumas informações sobre como as ações do agente alteram o mundo. Este conhecimento chama-se de modelo do mundo. Um agente que usa tal modelo é chamado de agente baseado em modelo. (RUSSELL; NORVIG, 2004).
- **Agentes baseados em objetivo**: Saber o estado atual do ambiente nem sempre será o suficiente para decidir o que fazer. Exemplo: uma junção de ruas, o táxi pode virar à esquerda, à direita ou ir em frente. A decisão depende do destino final, além da

descrição do estado atual, o agente precisa de algum tipo de informação com objetivo que descreva o destino do passageiro. (RUSSELL; NORVIG, 2004).

O agente baseado em objetivo parece menos eficiente, sendo mais flexível pois o conhecimento que suporta suas decisões é representado explicitamente e pode ser alterado. Para um agente reflexivo, nós podemos ter que reescrever muitas regras de condição-ação. (RUSSELL; NORVIG, 2004).

- **Agentes baseados em utilidade**: Objetivos sozinhos não são realmente suficientes para gerar comportamento de alta qualidade na maioria dos ambientes. Exemplo: existem muitas sequências de ações que levariam o táxi ao seu objetivo final mais, rápidos, seguros e confiáveis. Objetivos apenas proveem uma distinção binária crua, que permite uma comparação de estados de mundo preferível do que outro, tendo maior utilidade para o agente. (RUSSELL; NORVIG, 2004).

3.2.4. Agentes de resolução de problemas

Os agentes inteligentes devem maximizar sua medida de desempenho com o objetivo às vezes simplificado, adotando uma meta de alcançá-lo. (RUSSELL; NORVIG, 2004).

Exemplo: imagine um agente na cidade de Arad, na Roménia, aproveitando uma viagem de férias. A medida de desempenho do agente contém muitos fatores um deles melhorar seu bronzeado e o outro conhecimento do idioma romeno. Os objetivos ajudam a organizar o comportamento do agente, pondo um limite nas ações para tentar alcançar um fim. A formulação de problemas se baseia na situação da circunstancia e na medida de desempenho, sendo o primeiro passo para a resolução. (RUSSELL; NORVIG, 2004).

Esse processo de procurar por tal sequência chama-se busca. Um algoritmo de busca recebe um problema como entrada e retorna uma solução sobre a forma de uma sequência de ações. Depois que uma solução é encontrada, as ações que ela recomenda podem ser executadas, isso se chama fase de execução. Desse modo, temos um simples projeto de "formular, buscar, executar" para o agente. (RUSSELL; NORVIG, 2004).

O projeto do agente também pressupõe que o estado inicial seja conhecido, sendo mais fácil conhecê-lo se o ambiente for observável. A ideia de enumerar "cursos alternativos de ação" pressupõe que o ambiente pode ser visto como discreto. Por fim, e mais importante, o projeto de agente pressupõe que o ambiente seja determinístico. As soluções para problemas são sequências de ações únicas, e assim elas não podem lidar com quaisquer eventos inesperados, as soluções são executadas sem atenção às percepções.

(RUSSELL; NORVIG, 2004).

Problemas e soluções bem definidos

Um problema pode ser definido formalmente por quatro componentes:

- O estado inicial em que o agente começa. Exemplo: o estado inicial do nosso agente na Romênia poderia ser descrito como Em (Arad).
- Uma descrição das ações possíveis que estão disponíveis para o agente.
- O teste de objetivo, que determina se um dado estado é um estado objetivo.
- Uma função de custo de caminho que atribui um custo numérico a cada caminho. (RUSSELL; NORVIG, 2004).

Exemplos de problemas

A abordagem de resolução de problemas é aplicada a uma ampla série de ambientes de tarefas. Um mini problema se destina a ilustrar ou exercitar diversos métodos de resolução de problemas. Ele pode ter uma descrição concisa e exata. Isso significa que pode ser usado com facilidade por diferentes buscadores, com a finalidade de comparar o desempenho de algoritmos. Um problema do mundo real será aquele cujas soluções de fato preocupam as pessoas. (RUSSELL; NORVIG, 2004).

Em busca de soluções

Depois de formular alguns problemas, agora precisamos resolvê-los. Isso pode ser feito por meio de uma busca em todos os estados. Existem técnicas de busca que utilizam uma árvore de busca explícita, gerada pelo estado inicial e pela função sucessor que, onde definem os estados. Em geral, podemos ter um grafo de busca em lugar de uma árvore de busca, quando o mesmo estado pode ser alcançado a partir de vários caminhos. (RUSSELL; NORVIG, 2004).

Como o estado inicial não pode ser o estado objetivo, precisa considerar alguns outros estados. Sendo feito pela expansão do estado atual, que se aplica a função de sucessor a esse, gerando um novo conjunto de estados. A essência da busca é seguir uma única opção, testar se aquele estado é o objetivo e após não ter achado uma solução, continuará a procurar nos demais estados até encontrar a solução ou até não ter outros estados. A estratégia de busca determina por qual estado iniciar. (RUSSELL; NORVIG, 2004).

Existem muitas maneiras de representar os *nós*, mas partindo do princípio que um *nó* será uma estrutura de dados com cinco componentes tomando por base em Russell e Norvig :

- *ESTADO*: O estado no espaço que o *nó* corresponde.
- *NÓ-PAI*: O *nó* na árvore de busca que o gerou.
- *AÇÃO*: A *ação* que foi aplicada ao pai para gerar o *nó*.
- *CUSTO-DO-CAMINHO*: O custo, tradicionalmente denotado por *g(n),* do caminho desde o estado inicial até o *nó*, indicado pelos ponteiros do pai.
- *PROFUNDIDADE*: O número de passos ao longo do caminho, desde o estado inicial até o final. (RUSSELL; NORVIG, 2004).

Há relevância de distinguir entre *nós* e estados. Um *nó* será uma estrutura de dados de anotação usada para representar a árvore de busca. Um estado corresponde a uma configuração do mundo. Desse modo, os *nós* estão em caminhos específicos, definidos por ponteiros *Nó-PAI*. (RUSSELL; NORVIG, 2004).

Também precisamos representar a coleção de nós que foram gerados, mas ainda não expandidos, essa coleção chama-se borda. Cada elemento da borda será um *nó* folha, um *nó* sem sucessores na árvore. Supondo que a coleção de nós seja implementada como uma fila**.** As operações sobre uma fila concebido por Russell e Norvig (2004):

- ***CRIAR-FIM*** *(elemento, ...)* cria uma fila com o(s) elemento (s) dado (s).
- ***VAZIA*** *(Fila)* retorna verdadeiro somente se não existir mais nenhum elemento na fila.
- ***PRIMEIRO*** (*fila*) retorna o primeiro elemento da fila.
- ***REMOVER-PRIMEIRO*** (fila) retorna *PRIMEIR0* (*fila*) e o remove da fila.
- ***INSERIR*** *(elemento, fila)* insere um elemento na fila e retorna a fila resultante. (RUSSELL; NORVIG, 2004).

Medição do desempenho de resolução de problemas

A saída de um algoritmo de resolução de problemas consiste em *uma* solução. (Alguns algoritmos podem ficar paralisados em um laço de repetição infinito e nunca retornar uma saída.) Avalia-se o desempenho do algoritmo em quatro aspectos seguindo ainda Russell e Norvig (2004):

- **Completude:** O algoritmo oferece a garantia de encontrar uma solução caso ela exista, senão não considera como completo.
- **Otimização**: A estratégia para encontrar a solução é ótima, sendo esta a melhor forma de chegar ao objetivo.
- **Complexidade de tempo:** Quanto tempo é levado para encontrar uma solução.
- **Complexidade de espaço:** Quanta memória necessita para executar a busca toda. (RUSSELL; NORVIG, 2004).

A complexidade de tempo e a complexidade de espaço são sempre consideradas em relação a alguma medida da dificuldade do problema. Em ciência da computação, a medida típica será o tamanho do grafo do espaço de estados, porque o grafo visualiza-se como uma estrutura de dados explícita inserida no programa de busca. (RUSSELL; NORVIG, 2004).

Em IA, na qual o grafo será representado pelo estado inicial pela função sucessor e com frequência pode ser infinito, a complexidade se expressa em termos de três quantidades:

- *b:* o número de máximo de sucessores de qualquer nó.
- *d:* a profundidade do nó objetivo menos profundo.
- *n:* o comprimento máximo de qualquer caminho no espaço de estados.

Para avaliar a efetividade de um algoritmo de busca, se considera apenas o custo de busca que em geral depende da complexidade de tempo, onde pode incluir um termo para uso da memória. (RUSSELL; NORVIG, 2004).

Estratégias de busca sem informação

Algumas estratégias de busca reunidas sob o título de busca sem informação. A expressão significa que elas não têm nenhuma informação adicional sobre estados, além daqueles fornecidos na definição do problema. Tudo o que eles podem fazer é gerar sucessores e distinguir um estado objetivo de um estado não-objetivo. (RUSSELL; NORVIG, 2004).

- **Busca em extensão:** A busca em extensão é uma estratégia simples em que se expande primeiro no *nó* raiz, após isso expande todos os sucessores do mesmo, e assim sucessivamente. Em geral, todos os *nós* em uma dada profundidade na árvore de busca são expandidos, antes que todos os *nós* no nível seguinte sejam expandidos.

A busca em extensão pode ser implementada chamando-se BUSCA-EM-ÁRVORE com uma borda vazia que seja uma fila do tipo first-in-first-out (FIFO), assegurando-se que os *nós* visitados sejam expandidos primeiro.

- **Busca em profundidade:** A busca em profundidade sempre expande o *nó* mais profundo na borda atual da árvore de busca. Prosseguindo até o nível mais profundo da árvore de busca. Quando esses *nós* são expandidos, eles são retirados da borda, e então a busca "retorna" ao *nó* seguinte mais raso que ainda possui sucessores que não foram explorados.
  Essa estratégia pode ser implementada por BUSCA-EM-ÁRVORE com uma fila last-in-first-out (LIFO). Como uma alternativa para à implementação de BUSCA-EMÁRVORE, o mais comum a ser feito é utilizar uma função recursiva.
- **Busca em profundidade limitada:** O problema de árvores ilimitadas pode ser atenuado pela busca em profundidade com um limite de profundidade pré-determinando um valor para a parada. O limite de profundidade resolve o problema de caminhos infinitos, por outro lado, ele introduz uma fonte adicional incompleto.
- **Busca bidirecional:** A ideia que rege a busca bidirecional é executar duas buscas simultâneas uma a partir do estado inicial, e a outra a partir do objetivo, parando somente quando as duas buscas se encontrarem em um ponto intermediário.
  A busca bidirecional pode ser implementada fazendo-se uma ou ambas as buscas verificarem cada *nó* antes dele ser expandido, para ver se o *nó* está na borda da outra árvore de busca, encontrando assim uma solução. (RUSSELL; NORVIG, 2004).

Busca com informações parciais

O agente pode calcular exatamente o estado que resulta de qualquer sequência de ações e sempre sabe em que estado se encontra. Suas percepções não fornecem nenhuma informação nova depois de cada ação. Diferentes tipos de incompleto levam a três tipos distintos de problemas: (RUSSELL; NORVIG, 2004).

1) **Problemas sem sensores**: Se o agente não tem nenhum sensor, então ele poderia estar no meio de vários estados iniciais possíveis, e cada ação poderia, portanto, ir para vários estados sucessores prováveis.
2) **Problemas de contingência:** Se o ambiente for parcialmente observável ou se as ações forem incertas, as percepções do agente fornecerão *novas* informações depois de cada

ação. Cada percepção possível define uma contingência que deve ser planejada, sendo um problema chamado adverso se a incerteza for causada pelas ações de outro agente.

3) **Problemas de exploração:** Quando os estados e as ações do ambiente são desconhecidos, o agente deve atuar para descobri-los. Os problemas de exploração podem ser visualizados como um caso extremo de problemas de contingência. (RUSSELL; NORVIG, 2004).

3.2.5. Jogos

A teoria dos jogos matemáticos, é um ramo da economia, onde vê qualquer ambiente multi-agente como um jogo, desde que o impacto de cada agente nos outros seja "significativo", independentemente dos agentes serem cooperativos ou competitivos. (RUSSELL; NORVIG, 2009).

Na IA, os jogos mais comuns supõem ser do tipo específicos, os teóricos do jogo chamam de jogos deterministas, *turn-taking*, *two-player*, zero-sum de informações perfeitas (como o xadrez). Em nossa terminologia, isso significa ambientes determinísticos e totalmente observáveis em que dois agentes agem alternadamente e em que os valores de utilidade no final do jogo são sempre iguais e opostos. (RUSSELL; NORVIG, 2009).

Os jogos envolvem as faculdades intelectuais dos seres humanos. Para pesquisadores de IA, a natureza abstrata dos jogos torna um assunto atraente para o estudo. Sendo que o estado de um jogo é fácil de representar, e os agentes geralmente se restringem a um pequeno número de ações cujos resultados são definidos por regras precisas. (RUSSELL; NORVIG, 2009).

Os jogos físicos, como o croquete e o hóquei no gelo, apresentam descrições muito mais complicadas, uma gama muito maior de ações possíveis e regras bastante imprecisas que definem a legalidade das ações. Com exceção do futebol de robô, esses jogos físicos não atraíram muito interesse na comunidade da AI. (RUSSELL; NORVIG, 2009).

Os jogos, ao contrário da maioria dos problemas de brinquedos, são interessantes por serem muito difíceis de solucionar. Exemplo: o xadrez tem um fator de ramificação média de cerca de 35, e os jogos frequentemente vão para 50 movimentos por cada jogador, então a árvore de busca tem cerca de 35.100 ou 10.154 *nós*. (RUSSELL; NORVIG, 2009).

Com uma definição do movimento ideal e um algoritmo para encontrá-lo. Analisando

as técnicas para escolher um bom movimento quando o tempo é limitado. A poda nos permite ignorar partes da árvore de busca que não faz diferença para a escolha final, e as funções de avaliação heurística permite aproximar a verdadeira utilidade de um estado sem fazer uma busca completa. (RUSSELL; NORVIG, 2009).

Considerando jogos com dois jogadores, com o qual chamamos de MAX e MIN. MAX se move primeiro, e depois trocam os movimentos entre eles até o fim do jogo. No final do jogo, os pontos são concedidos ao jogador vencedor e as penalidades são dadas ao perdedor. Um jogo pode ser formalmente definido como um tipo de problema de busca com os seguintes elementos: (RUSSELL; NORVIG, 2009).

- S_0*:* o estado inicial, que especifica como o jogo está configurado inicialmente.
- *JOGADOR (es):* define qual jogador tem o movimento em um estado.
- *AÇÕES (ACÇÕES):* retorna o conjunto de movimentos legais em um estado.
- *RESULTADO (s, a):* o modelo de transição, que define o resultado de um movimento.
- *TERMINAL-TEST (s):* Um teste de terminal, sendo verdadeiro quando o jogo acabou e falso quando não acabou. Os estados onde o jogo terminou são chamados de estados terminais.
- *UTILITY (s, p):* uma função de utilidade, define o valor numérico final para um jogo que termina no estado terminal *s* para um jogador *p*. Um jogo de soma zero se define como aquele em que a recompensa total para todos os jogadores é a mesma para todas as instâncias do jogo. O xadrez é zero-sum porque cada jogo tem retorno de 0 + 1, 1 + 0 ou 1/2 + 1/2. (RUSSELL; NORVIG, 2009).

O estado inicial, a função *ACÇÕES* e a função *RESULT* definem a árvore do jogo onde os *nós* são estados do jogo e as bordas são os movimentos. Para o xadrez há mais de 1040 *nós*, então a árvore do jogo será melhor considerado como uma construção teórica que não podemos perceber no mundo físico. Independentemente do tamanho da árvore do jogo, o trabalho do MAX será procurar uma boa jogada. Usamos o termo árvore de pesquisa quando se sobrepõe à árvore do jogo completo e examina *nós* suficientes para permitir que um jogador determine o próximo passo. (RUSSELL; NORVIG, 2009).

Decisões ótimos em jogos

Em um problema de busca normal, a solução ideal seria uma sequência de ações que

levem a um estado de objetivo. Na busca contraditória, MAX tenta encontrar uma estratégia contingente para os estados resultantes de cada movimento possível de MIN. O algoritmo de busca *AND-OR* com MAX desempenha o papel *OR* e *MIN* de *AND*. Em termos aproximados, uma estratégia ótima leva a resultados pelo menos tão bons quanto qualquer outra estratégia. (RUSSELL; NORVIG, 2009).

Dada uma árvore de jogo, a estratégia ideal pode ser determinada a partir do valor mini-max de cada *nó*, que se escreve como *MINIMAX* (n). O valor *minimax* de um *nó* é o utilitário (para MAX) de estar no estado correspondente, assumindo que ambos os jogadores jogam otimamente de lá até o final do jogo. Além disso, dada uma escolha, MAX prefere mover para um estado de valor máximo, enquanto que o MIN prefere um estado de valor mínimo. Então, temos o seguinte: (RUSSELL; NORVIG, 2009).

MINIMAX(s) =

$$\begin{cases} \boldsymbol{UTILITY}(\boldsymbol{s}) \textbf{ if } \boldsymbol{TERMINAL-TEST}(\boldsymbol{s}) \\ \boldsymbol{max}_{\boldsymbol{a}\in\textbf{Actions(S)}}\, \boldsymbol{MINIMAX}(\boldsymbol{RESULT}(\boldsymbol{s},\boldsymbol{a}))\ \boldsymbol{if}\ \boldsymbol{PLAYER}(\boldsymbol{s}) = \boldsymbol{MAX} \\ \boldsymbol{min}_{\boldsymbol{a}\in\boldsymbol{Actions}(\boldsymbol{s})}\boldsymbol{MINIMAX}(\boldsymbol{RESULT}(\boldsymbol{s},\boldsymbol{a}))\ \boldsymbol{if}\ \boldsymbol{PLAYER}(\boldsymbol{s}) = \boldsymbol{MIN} \end{cases}$$

Esta definição é ideal para MAX onde se pressupõe que o MIN também joga otimamente maximizando assim o resultado do pior caso para MAX. Outras estratégias contra adversários sub-ótimos podem fazer melhor do que a estratégia minimax, mas essas estratégias necessariamente pioram contra adversários ótimos. (RUSSELL; NORVIG, 2009).

O algoritmo minimax

O algoritmo minimax calcula a decisão minimax do estado atual. Usando uma simples computação recursiva dos valores minimax de cada estado sucessor, implementando diretamente as equações definidoras. A recursão prossegue todo o caminho até as folhas da árvore, em seguida, os valores minimax são copiados através da árvore à medida que a recursão se desenrola. (RUSSELL; NORVIG, 2009).

O algoritmo minimax realiza uma completa profundidade-primeira exploração da árvore do jogo. Se a profundidade máxima da árvore é *m* e há *b* movimentos legais em cada ponto, então a complexidade do tempo do algoritmo minimax é O (b^m). A complexidade do espaço é O (b^m) para um algoritmo que gera todas as ações ao mesmo tempo, ou O (m) para um algoritmo que gera ações uma de cada vez. (RUSSELL; NORVIG, 2009).

Poda alpha-beta

O problema com a pesquisa minimax é o número de estados de jogo que tem que examinar sendo este exponencial na profundidade da árvore. Infelizmente, não podemos eliminar o expoente, mas resulta que podemos efetivamente cortá-lo ao meio. Considera o truque aquele que é possível calcular a decisão de minimax correta sem olhar para cada *nó* na árvore do jogo. A técnica particular que examinamos é chamada de poda *alpha-beta*. Quando aplicada a uma árvore minimax padrão, ela retorna o mesmo movimento que seria, mas diminui os ramos que não podem influenciar na decisão final. (RUSSELL; NORVIG, 2009).

A poda *alpha-beta* pode ser aplicada em árvores de qualquer profundidade, e muitas vezes é possível podar sub-árvore inteiras em vez de apenas folhas. O princípio geral é este: considere um *nó n* em algum lugar da árvore, de modo que o jogador tenha a opção de se mudar *nó* melhor do que onde está. Se o jogador tiver uma melhor escolha *m* no *nó* pai de *n* ou em qualquer ponto de escolha mais alto, então *n* nunca será alcançado na peça real. Então, uma vez que descobrimos o suficiente sobre *n* para chegar a esta conclusão, podemos podar. (RUSSELL; NORVIG, 2009).

Lembrando que a pesquisa minimax faz pesquisa na profundidade em primeiro lugar, em qualquer momento, precisa considerar os *nós* ao longo de um único caminho na árvore. A poda *alpha-beta* obtém o nome dos seguintes dois parâmetros que descrevem os limites dos valores de *backup* que aparecem em qualquer lugar do caminho: (RUSSELL; NORVIG, 2009).

α = o maior valor que foi encontrado até agora em qualquer ponto de escolha ao longo do caminho para MAX.

β = o menor valor que foi encontrado até agora em qualquer ponto de escolha ao longo do caminho para MIN.

A pesquisa *alpha-beta* atualiza os valores de α e β à medida que ele avança e corta os ramos restantes em um *nó* assim que o valor do nó atual for conhecido por ser pior do que o valor α ou β atual para MAX ou MIN. (RUSSELL; NORVIG, 2009).

Ordem de movimento

A eficácia da poda *alfa-beta* é altamente dependente da ordem em que os estados são examinados. Isso sugere que valha a pena tentar primeiro examinar os sucessores que

provavelmente podem ser melhores. (RUSSELL; NORVIG, 2009).

Se isso pode ser feito, o *alpha-beta* precisa examinar apenas *nós* O ($b^{m/2}$) para escolher o melhor movimento, em vez de O (b^{m}) para minimax. Isso significa que o fator de ramificação efetivo se torna √b em vez de b para o xadrez. O *alpha-beta* pode resolver uma árvore aproximadamente duas vezes maior do que o minimax na mesma quantidade de tempo. Se os sucessores forem examinados em ordem aleatória em vez do melhor primeiro, o número total de nós examinados será aproximadamente O ($b^{3m/4}$) para b moderado. Para o xadrez, uma função de ordenação bastante simples leva você a aproximadamente um fator de 2 do melhor resultado O ($b^{m/2}$). (RUSSELL; NORVIG, 2009).

Adicionando esquemas de ordenação de movimentos dinâmicos, como tentar primeiro os movimentos que foram encontrados melhor no passado, aproxima bastante do limite teórico. Uma maneira de obter informações do movimento atual é a busca de aprofundamento iterativo. Primeiro, procure uma camada de profundidade e grave o melhor caminho de movimentos. Em seguida, procure uma camada mais profunda, mas use o caminho gravado para informar o pedido de movimento. (RUSSELL; NORVIG, 2009).

O aprofundamento iterativo em uma árvore de jogo exponencial acrescenta apenas uma fração constante ao tempo de busca total, que pode ser mais do que feito a partir de um melhor movimento. Os melhores movimentos são muitas vezes chamados de movimentos assassinos e para experimentá-los primeiro é chamado de heurística de movimento assassino. (RUSSELL; NORVIG, 2009).

Em muitos jogos, os estados repetidos ocorrem frequentemente por transposições diferentes permutações da sequência de movimentos que acabam na mesma posição. Por exemplo, se o *White* tiver um movimento, a1, que pode ser respondido por *Black* com b1 e um movimento não relacionado a2 no outro lado da placa que pode ser respondido por b2, então as sequências [a1, b1, a2, b2] E [a2, b2, a1, b1] ambos terminam na mesma posição. Vale a pena armazenar a avaliação da posição resultante em uma tabela *hash* a primeira vez que não seja recalculado em ocorrências subsequentes. (RUSSELL; NORVIG, 2009).

A tabela de *hash* das posições anteriormente vistas é tradicionalmente chamada de tabela de transposição. Uma tabela de transposição pode ter um efeito dramático, às vezes tanto quanto duplicar a profundidade de busca alcançável no xadrez. Por outro lado, se estamos avaliando um milhão de *nós* por segundo, em algum momento não é prático manter todos na tabela de transposição. Várias estratégias foram usadas para escolher quais *nós* devem manter e quais descartar. (RUSSELL; NORVIG, 2009).

Imperfeição da decisão em tempo real

O algoritmo minimax gera todo o espaço de busca do jogo, enquanto que o algoritmo *alpha-beta* nos permite podar grandes partes dele. No entanto, o *alpha-beta* ainda tem que procurar todo o caminho para os estados terminais para pelo menos uma parte do espaço de busca. Esta profundidade geralmente não é prática, porque os movimentos devem ser feitos em uma quantidade razoável de tempo tipicamente alguns minutos no máximo. (RUSSELL; NORVIG, 2009).

O artigo de (SHANNON, 1950), propôs em vez disso que os programas deveriam cortar a pesquisa mais cedo e aplicar uma função de avaliação heurística aos estados na pesquisa, transformando os *nós* não terminais efetivamente em folhas terminais. (RUSSELL; NORVIG, 2009).

Em outras palavras, a sugestão é alterar o minimax ou *alpha-beta* de duas maneiras: substituir a função de utilidade por uma função de avaliação heurística EVAL, que estima a utilidade da posição e substituir o teste terminal por um teste de corte que decide quando aplicar EVAL. Isso nos dá o seguinte para minimax heurístico para estado *s* e profundidade máxima *d*: (RUSSELL; NORVIG, 2009).

H-MINIMAX(s, d) =

$$\begin{cases} \text{EVAL(s) if CUTOFF} - \text{TEST(s, d)} \\ max_{\alpha \in actions(s)}\, H - MINIMAX(RESULT(s,a), d\ +\ 1)\ if\ PLAYER(s)\ =\ MAX \\ min_{\alpha \in actions(s)}\, H - MINIMAX(RESULT(s,a), d\ +\ 1)\ if\ PLAYER(s)\ =\ MIN \end{cases}$$

Funções de avaliação

Uma função de avaliação retorna uma estimativa da utilidade esperada do jogo a partir de uma determinada posição. A ideia de um estimador não era nova quando Shannon o propôs. Durante séculos, os jogadores de xadrez desenvolveram formas de julgar o valor de uma posição porque os humanos são ainda mais limitados na quantidade de pesquisa que podem fazer do que os programas de computador. (RUSSELL; NORVIG, 2009).

Deve ficar claro que o desempenho de um programa de jogo depende fortemente da qualidade de sua função de avaliação. Uma função de avaliação imprecisa orientará um agente em direção a posições que se tornem perdidas. (RUSSELL; NORVIG, 2009).

Primeiro, a função de avaliação deve ordenar os estados terminal da mesma maneira que a função de utilidade verdadeira: os estados que são vitórias devem avaliar melhor os

sorteios, o que, por sua vez, deve ser melhor que as perdas. Em segundo lugar, a computação não deve levar muito tempo! (O ponto inteiro é procurar mais rápido.) Em terceiro lugar, para os estados não terminais, a função de avaliação deve estar fortemente correlacionada com as chances reais de ganhar. (RUSSELL; NORVIG, 2009).

O xadrez não é um jogo de azar: sabe-se o estado atual e não tem nenhum dado. Mas se a pesquisa deve ser cortada em estados não-terminais, o algoritmo será necessariamente incerto sobre os resultados finais desses estados. Esse tipo de incerteza é induzido por limitações computacionais, e não informativos. Dada a quantidade limitada de computação que a função de avaliação pode fazer para um determinado estado, o melhor que pode fazer é adivinhar o resultado final. (RUSSELL; NORVIG, 2009).

A maioria das funções de avaliação funciona ao calcular vários recursos do estado. Exemplo: o xadrez, teríamos recursos para o número de peões brancos, peões pretos, rainhas brancas, rainhas negras e assim por diante. Os recursos, juntos, definem várias categorias ou classes de equivalência de estados: os estados em cada categoria têm os mesmos valores para todos os recursos. (RUSSELL; NORVIG, 2009).

Por exemplo, uma categoria contém todos os finais de jogos de dois-peões versus um-peão. Qualquer categoria dada, em geral, conterá alguns estados que levam a ganhos, alguns que levam a empates, e alguns que levam a perdas. A função da avaliação não pode saber quais estados são quais, mas podem retornar um único valor que reflete a proporção de estados com cada resultado. (RUSSELL; NORVIG, 2009).

Por exemplo, suponha que nossa experiência sugira que 72% dos estados encontrados na categoria dos dois-peões versus um-peão levem a uma vitória (utilidade +1); 20% a uma perda (0) e 8% a um sorteio (1/2). Em seguida, uma avaliação razoável para estados na categoria é o valor esperado: $(0{,}72 \times +1) + (0{,}20 \times 0) + (0{,}08 \times 1/2) = 0{,}76$. Em princípio, o valor esperado pode ser determinado para cada categoria, resultando em uma função de avaliação que funciona para qualquer estado. Tal como acontece com os estados terminais, a função de avaliação não precisa retornar os valores esperados reais, desde que a ordenação dos estados seja a mesma. (RUSSELL; NORVIG, 2009).

Na prática, esse tipo de análise exige muitas categorias e, muita experiência para estimar todas as probabilidades de vencer. A maioria das funções de avaliação calcula contribuições numéricas separadas de cada recurso e depois as combina para encontrar o valor total. (RUSSELL; NORVIG, 2009).

Por exemplo, os livros introdutórios de xadrez dão um valor material aproximado para

cada peça: peão vale 1, cavaleiro ou bispo vale 3, torre 5 e a rainha 9. Outras características como "boa estrutura de peões" e "rei Segurança" pode valer metade de um peão, esses valores de recurso são então simplesmente adicionados para obter a avaliação da posição. (RUSSELL; NORVIG, 2009).

Matematicamente, esse tipo de função de avaliação é chamada de função linear ponderada porque pode ser expressa como $EVAL(\mathrm{s}) = w_1 f_1(s) + w_2 f_2(s) + \cdots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)$. Onde cada w_i é um peso e cada f_i é uma característica da posição. Para o xadrez, os f_i poderiam ser os números de cada tipo de peça no tabuleiro, e os w_i os valores das peças. (RUSSELL; NORVIG, 2009).

Adicionando os valores dos recursos parece ser uma coisa razoável a fazer, mas na verdade envolve uma forte suposição: que a contribuição de cada recurso é independente dos valores dos outros recursos. Exemplo, atribuir o valor 3 a um bispo ignora o fato de que os bispos são mais poderosos no final do jogo, quando eles têm muito espaço para manobrar. Por esse motivo, os programas atuais para xadrez e outros jogos também usam combinações não-lineares de recursos. (RUSSELL; NORVIG, 2009).

Jogos parciais e observável

O xadrez tem sido frequentemente descrito como uma guerra em miniatura, mas falta pelo menos uma das principais características das guerras reais, a saber, a observabilidade parcial. No "nevoeiro da guerra", a existência e a disposição das unidades inimigas muitas vezes são desconhecidas até serem reveladas pelo contato direto. Como resultado, a guerra inclui o uso de espiões para coletar informações e o uso de ocultação e blefar para confundir o inimigo. Os jogos parcialmente observáveis compartilham essas características e, são qualitativamente diferentes dos jogos descritos. (RUSSELL; NORVIG, 2009).

Nesse contexto, traçou um breve estudo para colher indicadores de conteúdo adequado para a utilização posterior da IA no X.A.P.R.O. Com a proposta de desenvolver uma IA direcionado ao jogo do xadrez, além de outras funcionalidades que sejam focados com a mesma aplicação de raciocínio lógico. (RUSSELL; NORVIG, 2009).

A relação desse material nesse processo de criação do braço robótico potencializa uma forma de estratégica para cada movimento da peça criada. Combinando cálculos de probabilidade para buscar um caminho de melhor utilização pela IA, assim, otimizando o processamento de dados em cada jogada no xadrez. (RUSSELL; NORVIG, 2009).

Capítulo IV

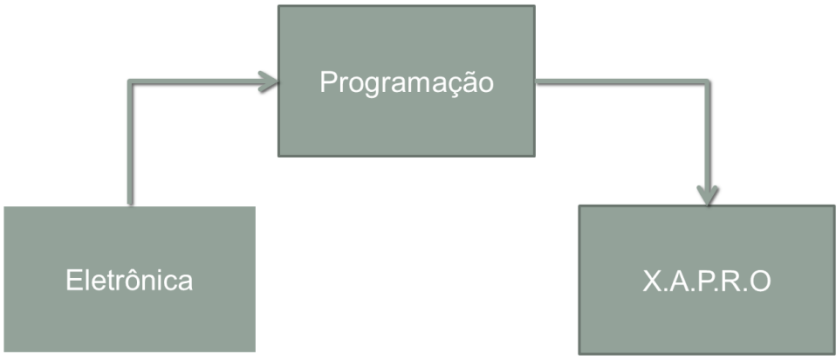
Neste capítulo foi demonstrado o procedimento metodológico: na primeira parte mostrou como foram as etapas de escolhas de tecnologias para a montagem do 6 DoF e a montagem do manipulador com os servos. Na segunda parte mostrou quais os códigos em python utilizado para fazer o manipulador se mover, sendo estes códigos um padrão para a utilização de um servo motor no Raspberry Pi. E na terceira parte as fases de teste do manipulador para a movimentação de uma peça.

4.1. Procedimento metodológico

Quando escolhido esta temática, desde o inicio sabia-se que seria um grande desafio, por dois motivos o primeiro porque trata-se de um tema multidisciplinar, e segundo por depender da combinação de conhecimento de áreas diferentes para fazê-lo funcionar.

Iniciou-se então, com a eletrônica na primeira parte simulando a quantidade de energia necessária para o funcionamento de todos os servos. Em seguida prosseguiu com a programação especifica que foi necessário à movimentação do X.A.P.R.O. Exposto na Figura 11.

Figura 11 – Passos para montar o X.A.P.R.O



Fonte: Elaborado pelo autor

As escolhas da parte eletrônica para o X.A.P.R.O parte com a escolha de um micro controlador. O Raspberry Pi, um dispositivo de sistema em chip, esse similar ao um computador: tem memória, processador, portas USB, etc. As escolhas constam na Figura 12.

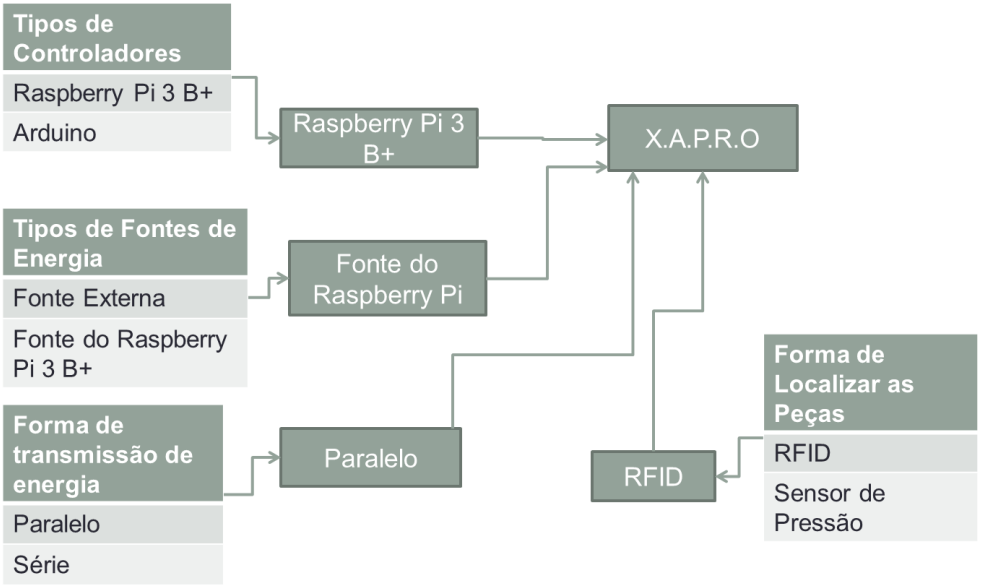
O arduino é um micro controlador, menos eficiente do que o Raspberry Pi, o qual para sua execução não precisa de nenhum sistema operacional e aplicativos de software. Há necessidade apenas escrever algumas linhas de código para usá-lo.

A forma mais eficiente de utilização de enegia sem usar os resistores foi usar os pinos de energia que o Raspberry Pi possui.

A energia paralela tem como consumo que pode ser distribuido igualmente entre os servos, enquanto que em energia série faria com que todos os servos fossem ligados em um único ponto de entrada, perdendo a energia necessária para o funcionamento dos servos.

A identificação por radiofrequência (RFID) usa campos eletromagnéticos para identificar e rastrear automaticamente as tags anexadas aos objetos. As tags ativas possuem uma fonte de energia local (como uma bateria) e podem operar centenas de metros do leitor de RFID. Figura 12.

Figura 12 – Escolhas na parte de eletrônica



Fonte: Elaborado pelo autor

Para a programação foi composta de várias tentativas de acerto e erros para se conseguir ajustar ao que era necessário para a movimentação do XAPRO.

A linguagem utilizada é a python pelo motivo de ser uma linguagem dinâmica em comparação as outras linguagens possiveis para a programação para a movimentação dos servos motores. Figura 13.

Figura 13 – Escolhas de Programação

Fonte: Elaborado pelo autor

4.2. Procedimento metodológico: Montagem

Os materiais que foram usados no desenvolvimento do XAPRO, foram os seguintes:

1. Um manipulador 6 DoF (Figura 4.1)
   1.1. 3x vigas U
   1.2. 4x suporte em U
   1.3. 5x suporte multifuncional
   1.4. 1x suporte em L
   1.5. 1x garra mecânica
2. 6x servos motores modelo (MG996R_Tower-Pro) (Figura 4.2)
3. Um raspberry Pi 3 B+ (Figura 4.3)
4. Uma fonte de energia de 300W
5. Um protoboard
6. Um microSD 16GB Classe 10 Ultra

Nas Figuras 14, 15 e 16 são mostradas as principais peças usadas, conforme está na descrição de cada figura.

Na Figura 14, é mostrado o 6 DoF desmontado com todas as peças e parafusos com o qual ele é composto, sendo estas peças montadas formando um braço robótico parecido com um braço humano e possuindo uma garra para interações com objetos tridimensionais.

Figura 14 - Manipulador 6 DoF

Fonte: Elaborado pelo autor

Na Figura 15, é mostrado os seis servos motores que são utilizados no manipulador cada um com 11 kgf.cm usando 6V e a velocidade de operação é de 0,14s/60º.

Figura 15 - Servos motores que foram utilizados no manipulador 6 DoF (Servos motores modelo MG996R_Tower-Pro)

Fonte: Elaborado pelo autor

Na Figura 16, é mostrado o raspberry Pi 3 B+, que possui um processador quad-core de 1.2 GHz, 1 GB de memória RAM, 40 pinos GPIOs, 4 portas USB 2 para o teclado, mouse entre outros dispositivos que utiliza a entrada USB 2, entrada para o cartão micro SD de acordo com as especificações do manual.

Figura 16 - Raspberry Pi 3 B+, computador de placa única para o controle dos servos.

Fonte: Elaborado pelo autor

As etapas de montagem do manipulador foram as seguintes:

Etapa 1

Sequência da montagem do manipulador: primeiramente foi realizado a montagem da base de suporte do 6 DoF utilizando 2 vigas em U com 3 parafusos e 3 porcas para a sustentação da base. Já nas partes do braço que utilizam suporte em U foi usado 2 parafusos e 2 porcas. Como mostra a Figura 17.

Figura 17 - Primeiras etapas de montagem do 6 DoF

Fonte: Elaborado pelo autor

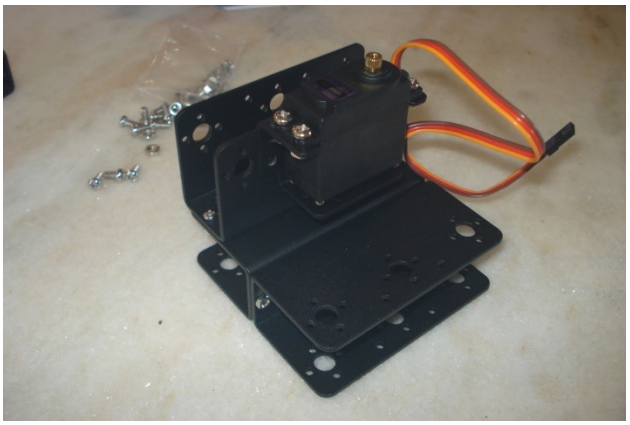
Na pré-montagem da garra com o servo, houve necessidade de alguns ajustes da garra para deixar mais firme. Utilizou-se 2 parafusos para anexar o servo junto a garra. Como demonstrado na Figura 18.

Figura 18 - Garra com servo

Fonte: Elaborado pelo autor

Expandindo a base de suporte do 6 DoF com uma viga em U, logo após adicionou-se um servo junto com o suporte multifuncional. Como mostra a Figura 19.

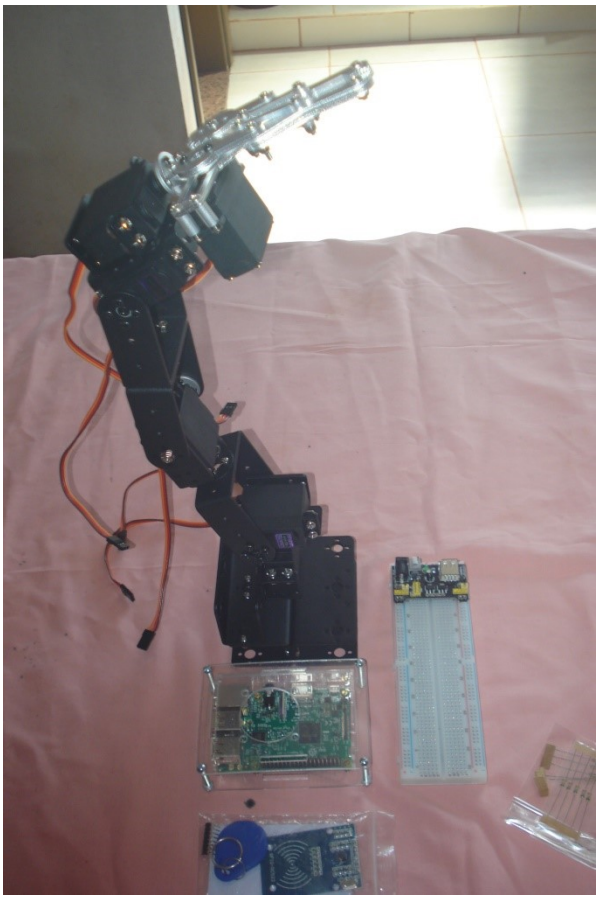
Figura 19 - Base de suporte 6 DoF



Fonte: Elaborado pelo autor

Na Figura 20 o manipulador apresenta-se completo, sendo possível medir a distância de interação do manipulador 6 DoF. Nessa fase da montagem do X.A.P.R.O ele já apresenta uma necessidade de projetar o tamanho do tabuleiro do jogo de xadrez proporcionando-lhe o raio de alcance total de ~35cm, para conseguir movimentar todas as peças.

Figura 20 - Manipulador 6 DoF montado por completo



Fonte: Elaborado pelo autor

4.3. Procedimento metodológico: programação

O programa inicial para dar movimentação ao manipulador teve como linguagem adotada o python, dentro do SO (Sistema Operacional) do raspbian, esse sistema tendo como base o debian do linux.

A seguir com o Servo ligado, foi testado o controle do mesmo. A princípio, como teste, foi utilizado o shell do Python. Para entrar nesse shell python, abriu-se uma janela do Terminal prosseguindo com a digitação do comando:

```
$ sudo python
```

Houve a importância de incluir o sudo, pois o Raspberry Pi só permite o acesso aos pinos GPIO ao superusuário. Portanto, foi preciso inserir o shell python como superusuário. Primeiro foi importado a biblioteca RPi.

```
>>> importar RPi.GPIO como GPIO
```

Precisou mencionar ao Pi o esquema da numeração de pinos que se deseja utilizar. Para isso foi usado o comando abaixo:

```
>>> GPIO.setmode (GPIO.BOARD)
```

Agora precisamos dizer ao Pi que o pino físico 11 será um pino de saída:

```
>>> GPIO.setup (11, GPIO.OUT)
```

A posição dos servos é controlada pela largura de pulso de um sinal PWM de 50 Hz. Portanto, precisamos ativar a sequência PWM em 50 Hz. Observe que, para um sinal de 50 Hz, o Período do sinal é 1/50 = .02 segundos ou 20 milissegundos.

```
>>> pwm = GPIO.PWM (11,50)
```

Então, para um sinal de 50 Hz, se configurarmos o DutyCycle para 5, deve-se observar o movimento do servo para a posição total para a esquerda. Da mesma forma, se configurarmos o DutyCycle para 7,5, devemos obter a posição do meio, e se configurarmos para 10, devemos estar na posição totalmente direita. Pode-se obter todas as posições intermediárias escalando linearmente entre 5 e 10. Podemos fazer isso com o comando:

```
>>> pwm.start (5)
```

Isso iniciará o sinal PWM, e será configurado em 5%. Para mudar a posição, pode-se utilizar o DutyCycle.

```
>>> pwm.ChangeDutyCycle (7.5)
```

Segundo programa mais a parte de instalação do wiringpi para o controle de mais de um GPIO.

```
#sudo apt-get install -y python-pip
#sudo pip install wiringpi
import time
import pigpio
SERVO = [11, 12, 13, 15, 16, 18]      # Servos conectados aos gpios 11, 12,
13, 15, 16, 18
DIR   = [1, -1, 1, 1, -1, 1]          # Direção de movimento do servo
PW    = [1500, 1500, 1500, 1500, 1500, 1500]
SPEED = [50, 50, 50, 50, 50, 50] # Velocidade de movimento dos servos

pi = pigpio.pi() # Conectado ao Pi local.
for x in SERVO:
   pi.set_mode(x, pigpio.OUTPUT) # Define o gpio como uma saída.
start = time.time()
while (time.time() - start) < 60: # Gira por 60 segundos.
   for x in range (len(SERVO)): # Para cada um dos servo.
      print("Servo {} largura do pulso {} microsegundos.".format(x, PW[x]))
      pi.set_servo_pulsewidth(SERVO[x], PW[x])
      PW[x] += (DIR[x] * SPEED[x])
```

```
      if (PW[x] < 1100) or (PW[x] > 1900): # Volta para os limites seguros.
         DIR[x] = - DIR[x]
      time.sleep(0.5)
for x in SERVO:
   pi.set_servo_pulsewidth(x, 0) # Troca o pulso dos servos para off.
pi.stop()
```

Terceiro programa e funcional para os movimentos necessários

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

##servo 5
servoPin=29
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
##while(1):

for i in range(80,50,-1):
    DC=1/18.*i+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.05)

##servo 5

##servo 4
servoPin=18
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
##while(1):
for i in range(0,35):
    DC=1./18.*(i)+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)

##servo 4

##servo 5
servoPin=29
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
##while(1):

for i in range(50,80):
    DC=1./18.*(i)+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)

##servo 5
```

```
##servo 4
servoPin=18
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
for i in range(35,0,-1):
    DC=1/18.*i+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)

##servo 4

##servo 1
servoPin=11
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)

##while(1):
for i in range(30,68):
    DC=1./18.*(i)+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)
##servo 1

##servo 4
servoPin=18
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
##while(1):
for i in range(0,35):
    DC=1./18.*(i)+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)

##servo 4

##servo 5
servoPin=29
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
##while(1):

for i in range(80,50,-1):
    DC=1/18.*i+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)

##servo 5

##servo 2
##servoPin=16
##GPIO.setup(servoPin,GPIO.OUT)
##pwm=GPIO.PWM(servoPin,50)
##pwm.start(7)

##while(1):
##for i in range(0,80):
##    DC=1./18.*(i)+2
```

```
##    pwm.ChangeDutyCycle(DC)
##    time.sleep(.02)

##servo 2

##servo 3

servoPin=15
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
##while(1):
for i in range(0,10):
    DC=1./18.*(i)+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)
##for i in range(80,0,-1):
##    DC=1/18.*i+2
##    pwm.ChangeDutyCycle(DC)
##    time.sleep(.02)

##servo 3

##servo 4
servoPin=18
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
for i in range(35,0,-1):
    DC=1/18.*i+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)

##servo 4

##servo 5
servoPin=29
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)
##while(1):

for i in range(50,80):
    DC=1./18.*(i)+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)

##servo 5

##servo 1
servoPin=11
GPIO.setup(servoPin,GPIO.OUT)
pwm=GPIO.PWM(servoPin,50)
pwm.start(7)

##while(1):
for i in range(68,30,-1):
    DC=1/18.*i+2
    pwm.ChangeDutyCycle(DC)
    time.sleep(.02)
##servo 1
```

```
pwm.stop()
GPIO.cleanup()
```

4.4. Procedimento metodológico: testes

Após a montagem, deu-se sequência com os testes, que foram efetuados com o protótipo desenvolvido, como descreve-se a seguir:

Primeiro teste:

O Raspberry utiliza um carregador de celular de 5V e 2 amp para o consumo próprio e funcionamento de seus periféricos, podendo assim fornecer energia pelos seus GPIO.

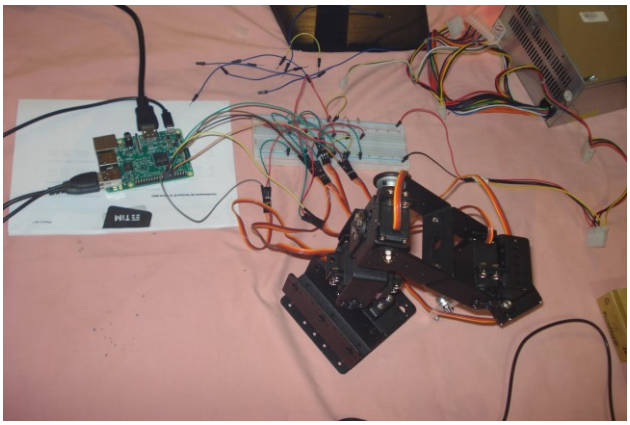
No primeiro teste do 6 DoF com o raspberry Pi 3, foi utilizado energia externa para a alimentação dos servos, pois caso fosse utilizado do próprio Raspberry não seria possível fornecer energia para os 6 servos. Com isso foi usado um *protoboard*, para montar a transmissão de energia para os servos.

Montou-se um esquema de transmissão de energia em paralelo, ao invés de usar de forma sequencial, com o qual mesmo que fosse usado uma fonte de energia externa não haveria energia o suficiente para cada um dos servos, que consomem de 4,8 a 6V. Sendo assim um total de 28,8 a 30V, caso fosse utilizado em sequência. A utilização em paralelo é possível fazer com que cada um deles utilize a energia necessária para o funcionamento.

A utilização da mesma forma de alimentação de energia, foi usado com o envio de pulsos onde também foi montado em paralelo, com apenas um cabo conectado a um dos GPIO gen.

Conectando todos os cabos realizou-se o primeiro teste iniciando com a forma de código que está exemplificado no Procedimento metodológico: programação. Ao utilizar as ondas em paralelo ele mandou um pulso para todos os servos, fazendo com que cada um deles se mova na posição que foi descrito no Procedimento metodológico: programação. Como mostra na Figura 21.

Figura 21 – manipulador conectado ao Raspberry Pi 3



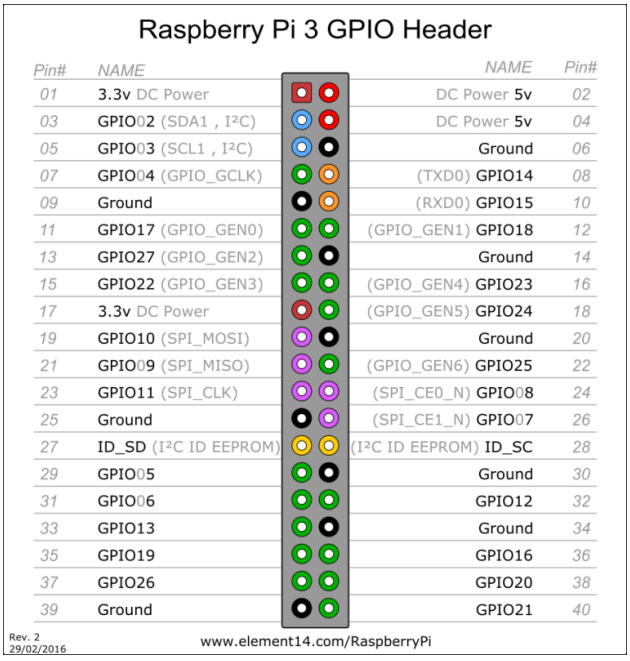
Fonte: Elaborado pelo autor

Nesse ponto pode-se observar os comandos inicias, seguido da movimentação de alguns servos que apontaram a necessidade de fazer com que cada um deles receba pulsos separados para se movimentarem por completo.

Segundo teste:

Utilizando o esquema de pinos da Figura 22 do Raspberry pi para verificar quais os possíveis pinos podem ser usados para a transmissão de pulsos.

Figura 22 - Pinagem do raspberry PI



Fonte: Gadgetoid and RogueHAL13.[1]

[1] https://pinout.xyz/

Após ter sido feito as alterações foi utilizado o segundo código descrito no Procedimento metodológico: programação, sendo possível observar que muitos servos conseguiram se mover e outros não, tais motivos apresentam talvez pelo próprio peso que ele possui na metade do manipulador ou por falta de torque.

Terceiro teste:

Foi experimentado aleatoriamente os servos um de cada vez conectando-os diretamente ao raspberry Pi, utilizando os GPIO's 29 e 11 de saída de pulsos e os GPIO's 2 de 5V e 9 de ground, para analisar o funcionamento dos servos e dos GPIO's de saída, após o teste observou-se que ambos estavam funcionando corretamente.

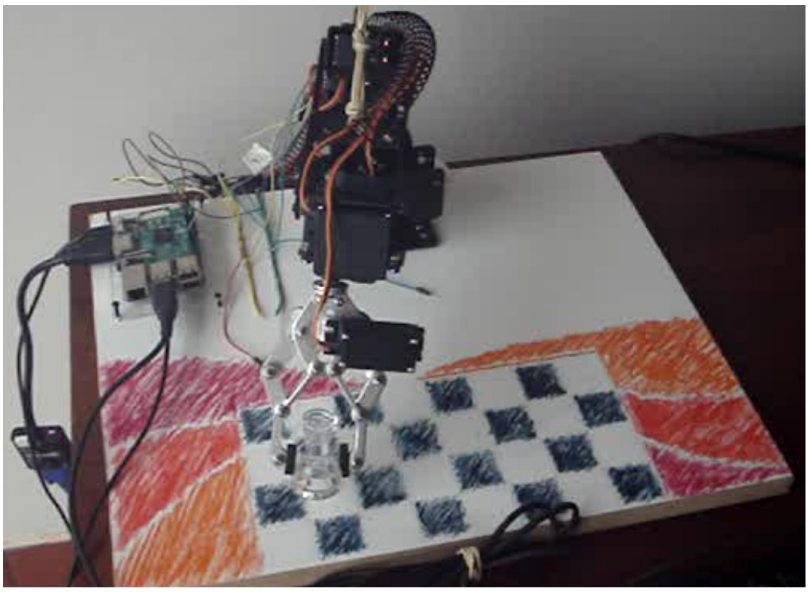
Portanto, ocorreu o teste novamente com a fonte de 300W nas saídas de 5V, logo correspondeu com uma resposta inicial aos comandos aos servos, e logo cessou os movimentos. Nesse momento foi utilizado outra fonte para o teste, sendo esta de 500W também tendo apenas uma resposta inicial, e não respondendo aos demais comandos.

Assim sendo, o provável funcionamento inicial do primeiro teste e todos os outros, com o qual se utilizou uma fonte de energia externa, observou-se que poderá ser este um dos problemas ou talvez de como está montado a transmissão de energia para os servos.

Quarto teste:

No quarto teste deixou de utilizar o *protoboarb*, e foi montado um circuito em paralelo com fios de alta condutividade. E, houve necessidade também de acrescentar na vertical uma haste de ferro na parte da base de sustentação do braço para dar suporte e estabilização ao manipulador. O qual com todo o peso dos servos ele teve dificuldade para se movimentar de um lado para o outro, com isso, logo solucionou e realizou o funcionamento de todos os servos do manipulador. Para completar utilizou-se do terceiro programa para ser possível o manipulador se movimentar, pegar a peça de xadrez e mover para a posição e o ângulo desejado dentro da base no tabuleiro pintado em xadrez em que foi usado como suporte para o manipulador, circuito e o Raspberry PI.

Figura 23 – Manipulador movendo a peça



Fonte: Elaborado pelo autor

Figura 24 – Manipulador com a haste para sustentação

Fonte: Elaborado pelo autor

Capítulo V

Considerações Finais

O braço robótico desenvolvido neste trabalho escora-se principalmente na área de mecânica (desenvolvimento estrutural, movimentação e sustentação), bem como, na programação. Entretanto, sendo um projeto piloto tratou-se com perspectiva multidisciplinar, para as áreas correlatas da programação, e da eletrônica, com prospecção para a inteligência artificial, probabilidade, entre outras.

Outra característica importante é que, além do seu propósito inicial, manipular peças para um jogo de xadrez, o robô também é capaz de desenvolver funções multiusos, dado a sua capacidade de "pegar" e "deslocar/posicionar" objetos em um plano pré-determinado. Então, há possibilidade de adaptação do sistema desde os trabalhos mais complexos até os mais simples para auxiliar o ser humano principalmente em tarefas de alta periculosidade, entretenimento entre outros.

O desenvolvimento do manipulador se deu da forma esperada e os resultados demonstram a viabilidade do seu uso, sendo que trabalhos futuros poderão desenvolver as outras partes para que o braço faça parte de um sistema completo e inteligente capaz de jogar xadrez com um adversário, quer humano ou outro robô.

Referências

BELLMAN (1978) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed.2004.

CAPEK (1920) apud **TZAFESTAS**, Spyros G., Introduction to Mobile Robot Control. Elsevier. First Edition. Waltham USA., 2014. In Freedman J. Robots through history: robotics. New York, NY: Rosen Central; 2011.

CASTELLS, Manuel. A sociedade em rede. A era da informação: economia, sociedade e cultura v. I. São Paulo: Paz e Terra, 1999.

CHARNIAK e **MCDERMOTT** (1985) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

CHI, M.; **GLASSER**, R. A capacidade para a resolução de problemas. In: **STERNBERG**, R. As capacidades intelectuais humanas: uma abordagem em processamento de informações. Porto Alegre: Artes Médicas, 1992.

CRAIG, John J., Robotica. Pearson. Sao Paulo Brasil. 2012.

DE LISI, R. A congnitive-developmental modelo f planning. In: **FRIEDMAN**, S; **SCHOLNICK**, E. K.; **COCKING**, R. Blueprints for thinking. Cambridge: Cambridge University Press, 1987.

FORBES, Duncan, The History of Chess, from the time of the early invention of the game in India, till the period of its establishment in western and central Europe, London, 1860.

HAYES, J. R. The complete problem solver. New Jersey: LEA, 1989.

HAUGELAND (1985) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

IFR. International Federation of Robotics. Presentation 12 Oct 2016 WR Service Robots. Disponível em: <https://ifr.org/>. Acesso em 19 de junho de 2017.

IFR. International Federation of Robotics. Presentation market overview World Robotics 29 9 2016. Disponível em: <https://ifr.org/>. Acesso em 19 de junho de 2017.

JOSEPH, Lentin, Learning Robotics Using Python. Pack Publising Ltd., Birmingham UK, 2015.

KURZWEIL (1990) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

MCCULLOCH, W. S.; **PITTS**, W.(1943) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

MORTON, Ella, Object of Intrigue: The Turk, a Mechanical Chess Player that Unsettled the World. 2015 Disponivel em: <http://www.atlasobscura.com/articles/object-of-intrigue-the-turk>. Acesso em 18 de junho de 2017.

NILSSOM (1998) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

POOLE et al. (1998) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

RICK e **KNIGHT** (1991) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

RUSSEL, J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.

RUSSEL, J. Stuart; **NORVIG,** Peter, Artificial Intelligence A Modern Approach , Third Edition, Pearson**,** New Jersey. 2009.

SHANNON, C. E. *Programming a Computer for Playing Chess,* revista: Philosophical Magazine, Ser.7, Vol. 41, No. 314 -March 1950.

STANDAGE, Tom. The Turk: The Life and Times of the Famous Eighteenth-Century Chess-Playing Machine, Walker, 2002.

TZAFESTAS, Spyros G., Introduction to Mobile Robot Control. Elsevier. First Edition. Waltham USA., 2014. In: Freedman J. Robots through history: robotics. New York, NY: Rosen Central; 2011.

WINSTON (1992) apud **RUSSEL,** J. Stuart; **NORVIG,** Peter, Inteligência Artificial, 2ª Ed. 2004.