PONTIFÍCIA
UNIVERSIDADE
CATÓLICA
DO RIO DE JANEIRO

INTERAÇÃO E COMUNICAÇÃO ENTRE MÚLTIPLOS ROBÔS

Aluno: Leonardo de Paula Batista Benevides

Orientador: Marley Maria B. R. Vellasco

Karla T. Figueiredo Leite

1.Introdução

1.1.Notas Iniciais

A robótica é uma área de pesquisa interdisciplinar, por natureza. Pode-se afirmar, de forma geral, que ela emprega ferramentas, metodologias e tecnologias inerentes a grandes áreas como a engenharia mecânica, engenharia mecatrônica (com história também recente e poucos cursos no Brasil), engenharia elétrica e eletrônica e engenharia de computação. A robótica utiliza-se de conceitos teóricos de grandes áreas como a matemática, mecânica, electronica, a teoria do controle de sistemas, automação industrial, a visão artificial por computador, comunicações, processamento de sinais, computação entre outras. Considera também, quando trata de modelos inteligentes de neurologia, fisiologia e psicologia.

Sob a ótica da robótica, em processos industriais e de produção, o desenvolvimento da área é de extrema importância devido ao alto risco de acidentes existente em grande parte das tarefas nestes processos. Falhas humanas não só podem trazer graves conseqüências ao trabalhador e ao equipamento industrial, mas também podem causar, como freqüentemente tem ocorrido, sérios danos ambientais e ao Homem.

Os Sistemas Multi-Agente (SMA) são uma área emergente da Inteligência Artificial que fornece os princípios para construção de sistemas complexos envolvendo múltiplos agentes e os mecanismos de coordenação entre eles. Um SMA pode ser definido como um grupo de agentes autônomos, interagindo entre sí e compartilhando um mesmo ambiente, que é percebido através de sensores, onde eles agem realizando ações. Este tipo de sistema vem encontrando uma grande variedade de campos de aplicação, como controle distribuído, times robóticos, controle de navegação, controle e planejamento de rotas , programação de elevadores, balanceamento de carga, negociação (trading) automática entre agentes, precificação dinâmica de produtos de varejo, etc.

Os benefícios trazidos pela aplicação de SMA são diversos. Em primeiro lugar, através da computação paralela, vários agentes podem trabalhar em conjunto para explorar melhor a estrutura descentralizada de uma determinada tarefa e acelerar sua conclusão. Além disso, agentes podem trocar experiências

se comunicando, podem apenas observar os mais habilidosos e aprender , os mais inteligentes podem servir de professores para outros agentes, etc. Os SMA também podem fornecer alto grau de escalabilidade, através da inclusão de novos agentes quando necessário, e ainda fazer com que agentes assumam a atividades de outros agentes em casos de falha. Como a inteligência está profundamente acoplada à interação, a melhor forma de criar máquinas inteligentes pode ser através da construção de “redes sociais” de máquinas.

Dessa forma, a motivação deste projeto é dar continuidade ao estudo da inteligência artificial que foi tópico do projeto anterior e contribuir para o desenvolvimento da pesquisa na área de robótica através de múltiplos robôs reais que devem interagir entre si utilizando o paradigma dos sistemas multi-agentes.

O presente trabalho utilizará robôs Lego *MindStorms* (Figura 1) com o objetivo de locomover os robôs de forma autônoma em ambiente desconhecido e encontrar um objetivo.

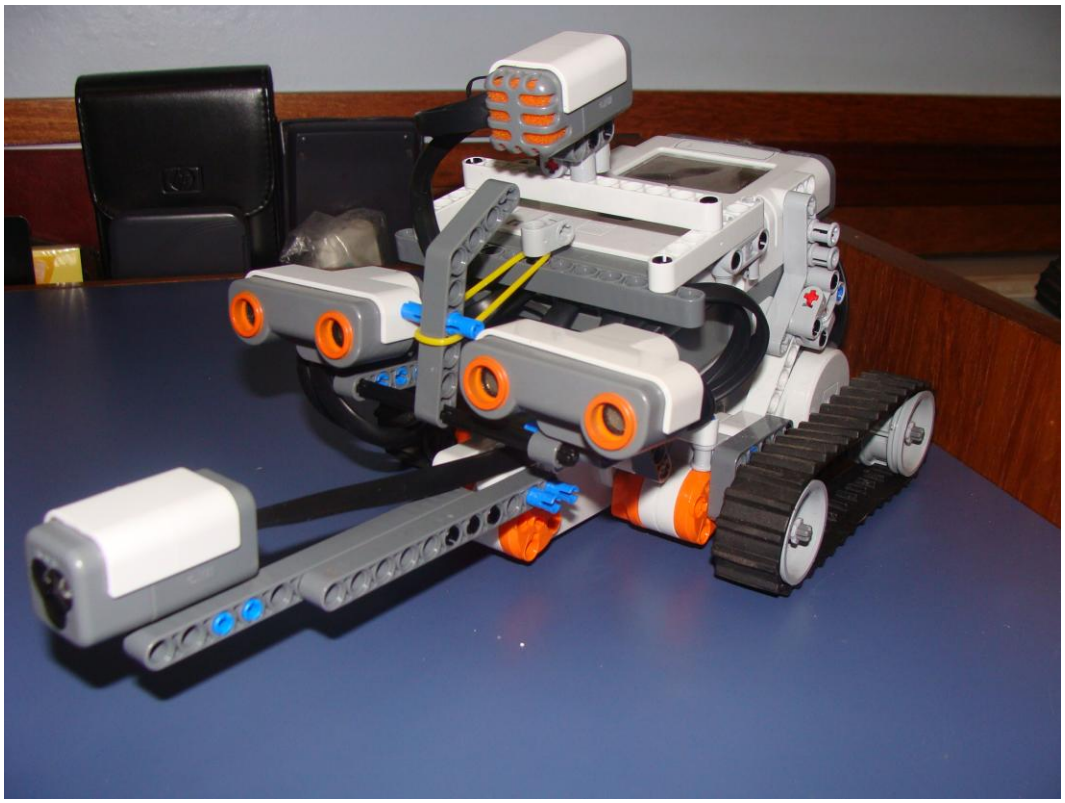

Figura 1 – Robô LEGO MindStorms NXT montado para avaliação do ambiente

A escolha do uso de múltiplos agentes foi determinada pela complexidade do problema em questão, onde cada robô terá de lidar com multipos estímulos de entradas simultâneos e diferentes situações.

Com a utilização de apenas um agente, este terá um trabalho muito maior em percorrer todo o ambiente, sendo o tempo gasto para realizar a tarefa aumentado exponencialmente.

O uso de múltiplos agentes, ao contrário do de único, prioriza a divisão de tarefas - no caso a divisão do espaço que esta sendo inspecionado - fazendo com que o esforço e o desgaste de cada agente seja reduzido produzindo ganhos de ordem financeira e de prazos.

Este método permite que problemas muito complexos sejam resolvidos através de forma simples e da compreensão global do modelo, tornando-o mais ágil e prático. Utiliza o antigo conceito de “Dividir para Conquistar” quebra-se a tarefa em pequenas tarefas de fácil resolução e atribui-se a vários agentes para que possam realizá-la simultaneamente.

1.2.Objetivos

O trabalho proposto, dando continuidade ao trabalho em andamento, tem por objetivos desenvolver o conhecimento sobre: a tecnologia de modelos multi-agentes para navegação autônoma de robôs reais, a capacitação no uso de ferramentas de simulação múltiplos robôs; investigação, estudo e aplicação de modelos para comunicação e interação entre robôs.

Desenvolvimento e implantação em Robôs LEGO MindStorms NXT de sistemas de controle com o objetivo de tornar autônoma a navegação dos robôs em ambientes de qualquer dimensão, visando um objetivo final – neste trabalho uma caixa vermelha que emite um som. Para tanto, o robô deve ser capaz de perceber e evitar obstáculos.

Ao ser acionado, o NXT deverá percorrer o ambiente a procura de um objeto que possua a cor vermelha e que o nível sonoro esteja acima de uma faixa, localizar objeto e enviar as coordenadas atuais para os outros NXT’s. Os agentes que não encontram o objetivo devem recalcular seus trajetos para que todos consigam chegar no objetivo independente da posição que estejam, sempre sem nenhum tipo de interferência humana.

Como não se dispõe de sistemas de posicionamento sofisticados, os Robôs poderão executar a rotina de localização do seu objetivo repetidas vezes, até que algum atinja o objetivo.

1.3.Justificativa

Este projeto compõe os princípios básicos de um mecanismo de navegação de múltiplos agentes autônomos em ambiente inexplorado. A partir deste conceito, pode-se evoluir a idéia para diversas situações reais, com objetivos tanto comerciais quanto de resgate.

Como exemplos comerciais, podemos citar robôs localizadores de poços de petróleo ou de sondas espaciais que buscam informações e vida em outros planetas ou aspiradores de pó e mini-enceradeiras.

Já as utilidades do algoritmo para finalidades de resgate poderiam ser robôs localizadores e recuperadores de sobreviventes e caixas pretas de aviões que sofreram acidentes sem motivos aparentes, de robôs localizadores minas terrestres (já existentes) e robôs que fizessem medições do nível radioativo de uma determinada região que sofreu contaminação nuclear.

Concluindo, pode-se perceber que se está diante de um problema recorrente em diversas áreas. Praticamente qualquer projeto que requeira busca em ambientes extensos ou desconhecidos seria um “cliente em potencial” para nossa aplicação.

2.Hardware

2.1.LEGO MINSTORMS NXT

Figura 2 – Brick LEGO MindStorms NXT

Resultado de uma parceria entre o MIT e o LEGO Group, o LEGO MindStorms é a linha que associa as peças de montar à evolução da tecnologia. O brinquedo evoluiu passando da linha *Technic*, cujos encaixes mais inteligentes permitiam movimentos de articulações, eixos e engrenagens, para serem associados a motores, para finalmente permitir a participação de sensores e microcontroladores.

A primeira versão comercializada do MindStorms denominada *"Robotics Invention Systems"* (RIS) era controlada pelo *brick* RCX provido de um microcontrolador de 8 bits e 32k de memória RAM para armazenar o firmware e o programa do usuário que aceitava diversas linguagens de programação e era realizada por uma interface infravermelha entre o computador e brick.

Este produto passou a ser adotado em escolas e universidades com fins didáticos, pois todos os aspectos da dinâmica robótica eram aplicados, de forma simplificada, desafiando a construir projetos envolvendo estruturação, mecânica, programação e a capacidade de correlacionar todos estes formando o raciocínio mecatrônico.

Em 2006 os kits LEGO MindStorms NXT (Figura 3) evoluíram em relação ao seu antecessor, o *brick* passou a contar com dois microprocessadores, um principal de 32 bits, com 48MHz, 256k de memória flash e 64k de memória RAM, e outro secundário de 8 bits, 4MHz, 4k de memória flash e 512b de RAM, um adaptador *Bluetooth* interno e uma entrada USB. Esta nova versão disponibilização o código aberto do firmware no site da própria empresa, possibilitando assim, de forma ilimitada, as capacidades de se desenvolverem novas utilidades para o produto.

O uso no meio acadêmico possibilitou a criação de competições de robôs, onde instituições de ensino apresentam robôs desenvolvidos por seus

alunos para competirem entre si. As modalidades presentes nas competições consistem desde uma simples tarefa que os participantes montem um robô que cumpra um objetivo até as mais complexas, como que as instituições tragam times de robôs que joguem uma partida de futebol.

O kit(Figura 3) conta com o *brick* (1), este com entrada para três servo-motores e quatro sensores, os três servo-motores (6), um sensor ultra-som (5), um sensor infra-vermelho (4 -utilizado para percepção luminosa de ambientes claros e escuros), um sonoro (3 - que pode ser ajustado na escala decibel (dB) para detectar todos os sons com a mesma sensibilidade) e dois de toque (2-capazes de perceberem quando são pressionado por algum objeto, e também quando é liberado novamente).

Figura 3 – Kit LEGO MindStorms NXT

A LEGO disponibiliza também sensores de cor e de movimentos como o *Gyro(*percebe a velocidade de rotação), *Compass(*percebe a direção de rotação) *(Figura 4)* e um sensor acelerômetro, que percebe aceleração nos três eixos ortogonais.

Figura 4 – Outros sensores.

2.2.Peças Utilizadas

Os servo-motores podem funcionam como entradas e/ou saídas de dados . Eles funcionam como um motor de passos, sendo possível controlar o quanto o motor deve girar, e com qual velocidade ele deve fazê-lo em uma

escala de 0 a 100. Cada motor possui também um sensor de rotações embutido(hodômetro), que permite a leitura dos movimentos do motor, podendo esta ser apresentada em graus, com precisão de +/- um grau e total para uma volta de 360 graus, ou em voltas completas, aplicada à medição de distâncias percorridas.

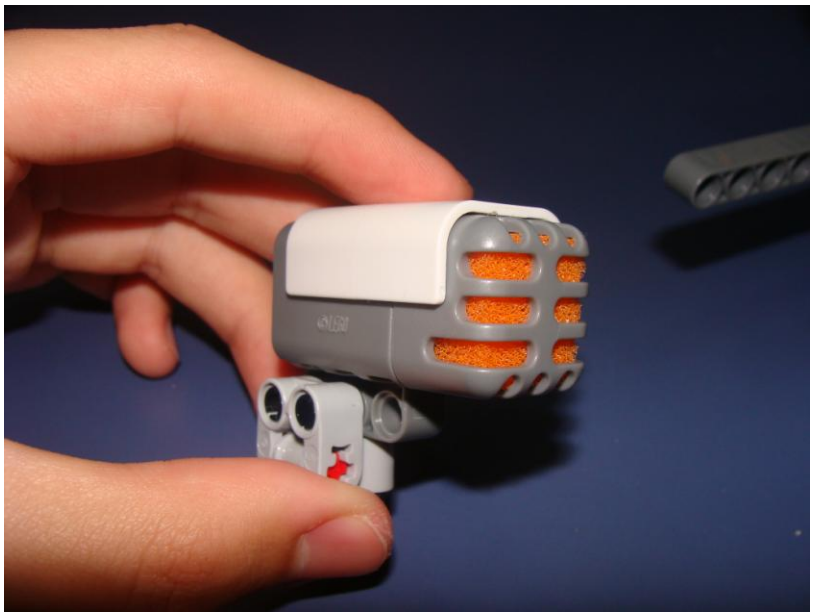

Figura 5 – sensor de som

O sensor de sons pode ser ajustado na escala decibel (dB) ou na escala decibel alocada (dBA), a primeira para detectar todos os sons com a mesma sensibilidade em uma abrangência que inclui sons que extrapolam a sensibilidade do ouvido humano, já a segunda posiciona a largura de banda para padrões do ouvido humano.

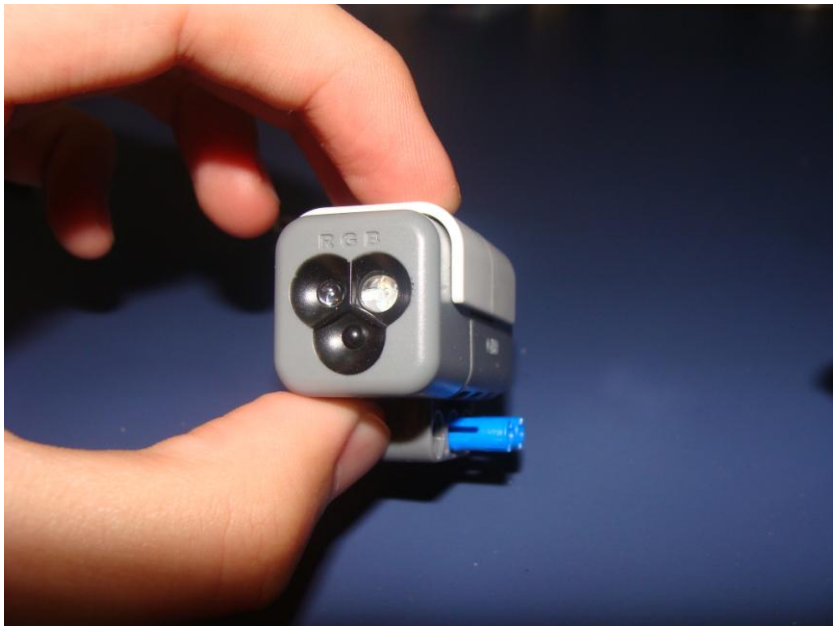

Figura 6 – Sensor de Cor

O sensor de Icor equipa o robô com a capacidade de distinguir cores diferentes. Ele é provido de três pequenos LEDs um de cada uma das cores primárias(vermelho, verde e azul) e de acordo com a vontade do usuário, podem ser acionadas para garantir a uniformidade de medidas de reflexão de superfícies. O sensor pode trabalhar em modo “FULL” com todos os três LEDs acesos ou com apenas deles e pode informar a medida através de um valor, através de uma cadeia de caracteres correspondente a cor lida ou na forma de 3 componentes RGB.

Figura 7 –sensor ultra-sônico

O sensor ultra-sônico realiza uma função que se equipara a visão de proximidade do ser humano. Funcionando do mesmo modo que os morcegos fazem para se localizarem em um ambiente, o sensor percebe obstáculos calculando o tempo de retorno de uma onda sonora refletida pelo objeto utiliza uma escala de 0 até 255.

2.3.Arquitetura Utilizada no Robô

Estrutura básica de locomoção do robô foi baseada na construção sugerida pela Lego (Figura 8), um robo dotado de dois servos ligados esteiras para deslocamento e direcionamento do robô,.

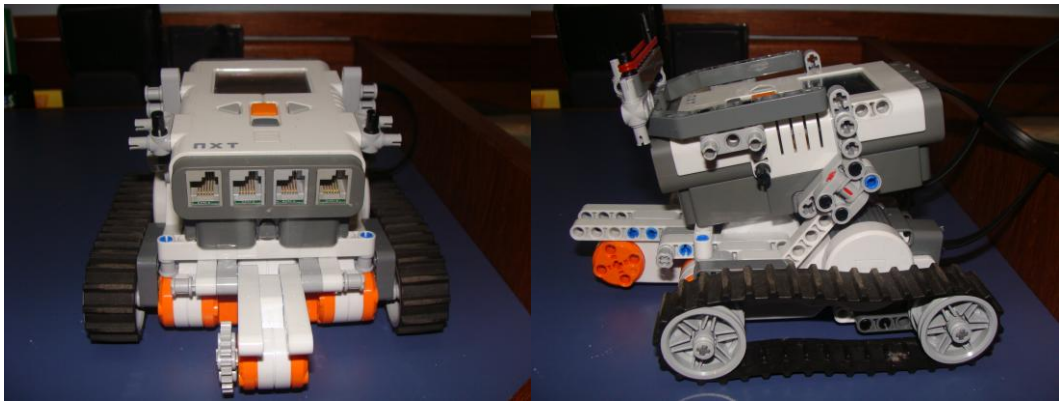

Figura 8 – Estrutura do robô de esteiras da LEGO.

Essa arquitetura foi escolhida por possibilitar controle total sobre a direção e o sentido de locomoção do robô, utilizando um mínimo de estrutura, ou seja, com apenas dois servos se tem a liberdade de movimentos desejada. Foi utilizado um terceiro motor só para preencher o espaço. Esta arquitetura do robô resulta em menos esforços necessários, entendendo-se isto por economia de energia possibilitando maior autonomia, além disso o uso de esteiras é de grande reconhecimento, visto que é de grande utilização em tanques de guerras e veículos que precisam se locomover sobre terrenos irregulares.

Para ser capaz de perceber obstáculos à sua frente, o robô foi equipado com dois sensores de ultra-som na sua dianteira (Figura 9), com uma abertura A análise das leituras realizadas por estes sensores são ao robô a visão ou percepção dos obstáculos posicionados a sua frente. Ao perceber um obstáculo localizado ligeiramente a sua esquerda, o robô toma a iniciativa de desviar para a direita. Esta noção seria impossível com a percepção utilizando apenas um sensor, o robô teria de, além de perceber o obstáculo, avaliar sua posição relativa para aí sim tomar a decisão do lado iria desviar.

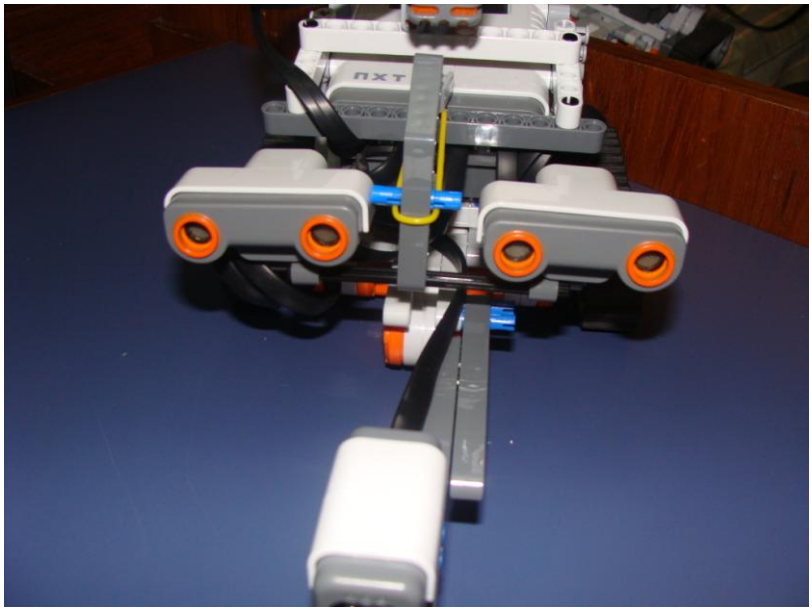

Figura 9– Disposição dos sensores ultra-sônicos.

Quanto à percepção sonora, o robô contará com um sensor de som alinhados perpendicularmente ao centro do eixo das esteiras (Figura 10) – voltado para a parte frontal do veículo – que, em uma rotina explicitada mais a frente no trabalho, será capaz de identificar a aproximação do agente em relação ao objetivo que se busca alcançar, porque quanto mais próximo o robô estiver de seu objetivo mais alto será o som e maior será o valor lido pelo sensor.

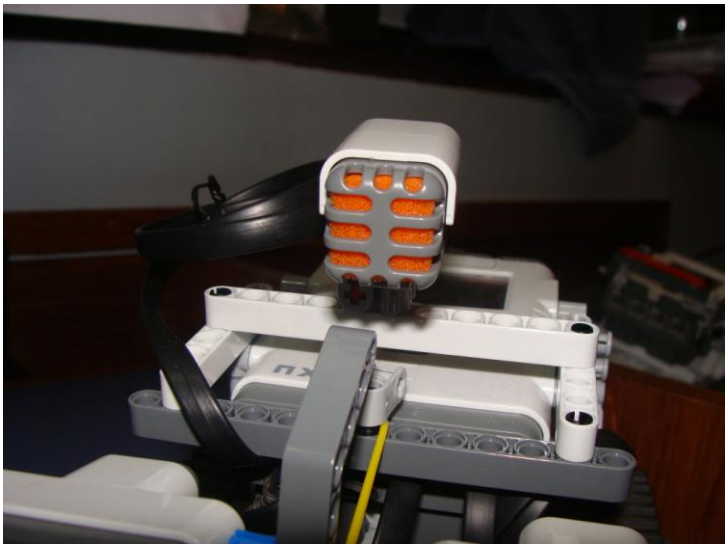

Figura 10 – Disposição do sensor sonoro

Já em relação à percepção de cor, o robô está equipado com um sensor alinhados perpendicularmente ao centro do eixo dos sensores ultra-sônicos (Figura 11) – voltado para a parte frontal do veículo – pendurado em uma haste

e avançado em relação aos sensores ultra-sônicos. Esse posicionamento deve-se ao fato dos sensores de cor conseguirem somente produzirem respostas eficientes quando estão realmente próximos dos elementos que estão examinando. Logo quando os sensores de distancia perceberem que um objeto esta perto o sensor de luz estará bem próximo do objeto podendo assim realizar a medida corretamente.

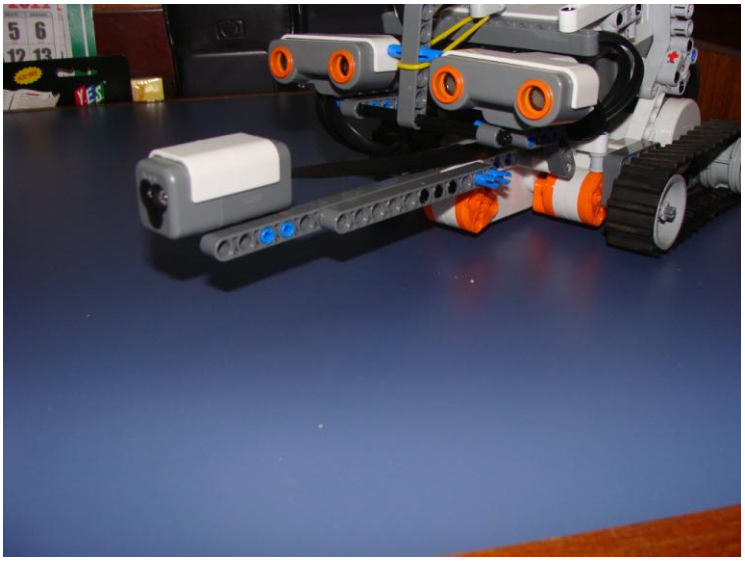

Figura 11 – Disposição do sensor de cor

3.Software

Idealmente, utiliza-se plataformas de simulação, onde o problema é modelado e as soluções propostas são avaliadas virtualmente para depois serem embarcadas no hardware e necessitar poucos ajustes.

Estão disponibilizados diversos softwares de programação para o LEGO MINDSTORMS NXT. Neste trabalho serão apresentados apenas os que foram utilizados.

3.1.LEGO MINDSTORMS Education's NXT Software v1.1

Este é o software que acompanha o Robô(Figura 12). É desenvolvido pela *National Instruments* (NI), o programa é baseado no software LabVIEW, também da mesma empresa. A linguagem de programação bastante intuitiva *"drag-and-drop"* (blocos que representam elementos são "arrastados para uma posição").

O kit LEGO MINDSTORMS possuir um enorme potencial computacional e é vendido também como brinquedo para crianças por ser muito simples, entretanto essa simplicidade traz limitações, apenas funções básicas de programação,portanto ele não é capaz de executar as rotinas desejadas.

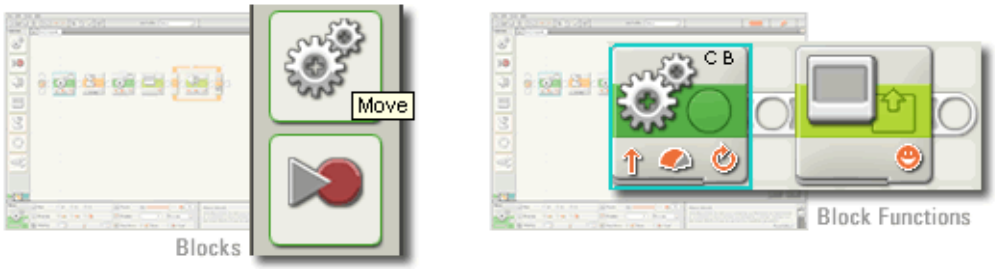

Figura 12 – Software LEGO NXT.

3.2.National Instruments LabVIEW 8.2

O LabVIEW utiliza-se de um ambiente gráfico de caráter extremamente intuitivo e fácil de operar (não é necessário conhecimentos das tradicionais linguagens textual de programação), baseado em ícones de funções, rotinas, variáveis e controles .

O programa é utilizado em larga escala pala indústria, por sua alta performance em aquisição, análise e manipulação de dados em todas as etapas do processo de produção.

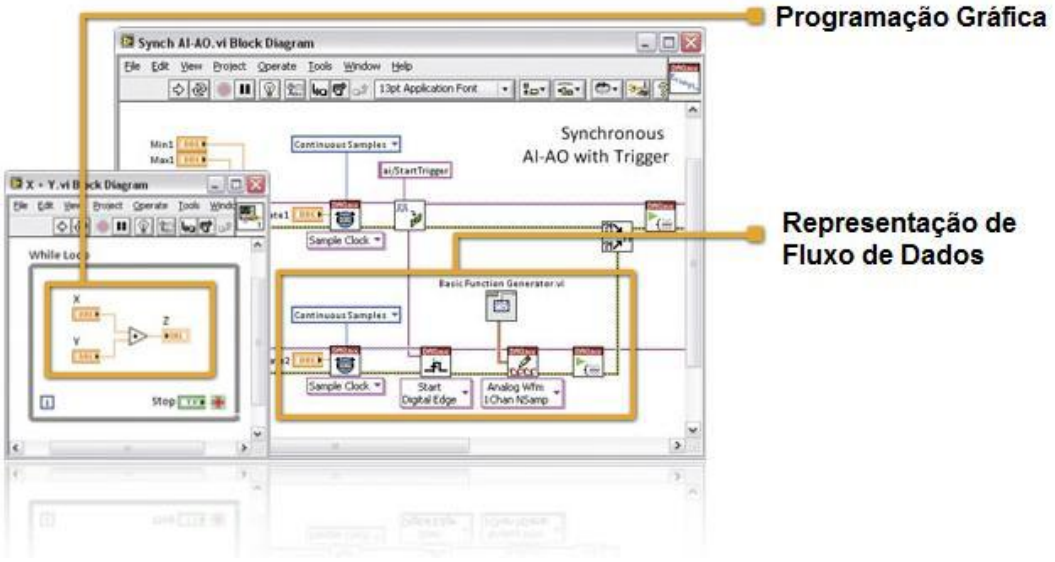

Figura 13 – LabVIEW.

A *National Instruments*, desenvolveu um toolkit para LabVIEW das funções de controle do NXT, o qual dispõe-se de todo o poder computacional do LabVIEW para realizarmos as rotinas mais elaboradas que não eram permitidas pelo programa fornecido junto com o robô.

Todo este poder computacional, porém, exige algumas limitações. O *Brick* do LEGO NXT não reconhece a integração com o software MATLAB nem arredondamentos numéricos e conversões entre variáveis. Por este motivo pode-se apenas gravar programas pouco complexos na memória do *Brick* para execução no mesmo. Assim, fica-se então limitado a uma conexão *online* entre o *Brick* e o computador, seja por *Bluetooth* ou via cabo USB, onde todo o trabalho computacional é realizado no micro, que o *Brick* apenas retransmitirá os comandos para os servos e dos sensores.

O LabVIEW possui uma interface de comunicação com o MATLAB(apresentado a seguir) , portanto facilita a construção de por exemplo um controlador Fuzzy no MATLAB, para posteriormente executá-lo no software da *National Instruments* através da janela *"MATLAB Script Node"(Figura 14)* que executas as funções perfeitamente sem acréscimo significativo de necessidade computacional nem de tempo de execução.

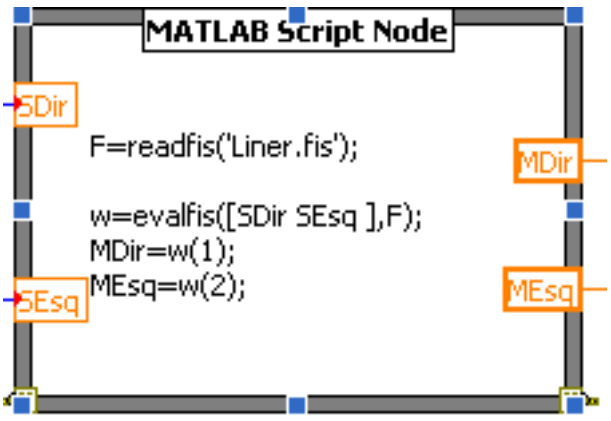

Figura 14 – MATLAB Script Node modo de chamar o MatLab dentro do LabView

A escolha deste software para elaboração do algoritmo de controle para este projeto foi devido a sua interface ágil e eficiente e o fato de ser a base do programa desenvolvido em parceria com a LEGO para controlar o NXT. Já tinha Entretanto foram identificadas limitações para a utilização de múltiplos Bricks passando-se assim necessário a utilização de apenas o próximo software apresentado.

3.3.MATLAB

O MATLAB, abreviação de *"Matrix Laboratory"*, juntamente com o Simulink, seu principal *"addon"* (Figura 15) são amplamente utilizados tanto para fins didáticos, em universidades, quanto para fins comerciais em empresas de pesquisa, de novas tecnologias e instituições financeiras.

As áreas de aplicação destas ferramentas englobam praticamente todos os ramos que envolvem análises matemáticas. Alguns exemplos são controle, com modelagem para sistemas de controle e produção de código para sistemas externos; processamento de sinais, com modelagem para processamento de sinais, contando também com ambiente de simulação e produção e verificação de código; computação técnica, com desenvolvimento, visualização e análises de algoritmos matemáticos; processamento de imagens, com rotinas de aquisição, análise, visualização e ambiente de desenvolvimento de algoritmo; finanças, com modelos financeiros, ambientes de análises e aplicações.

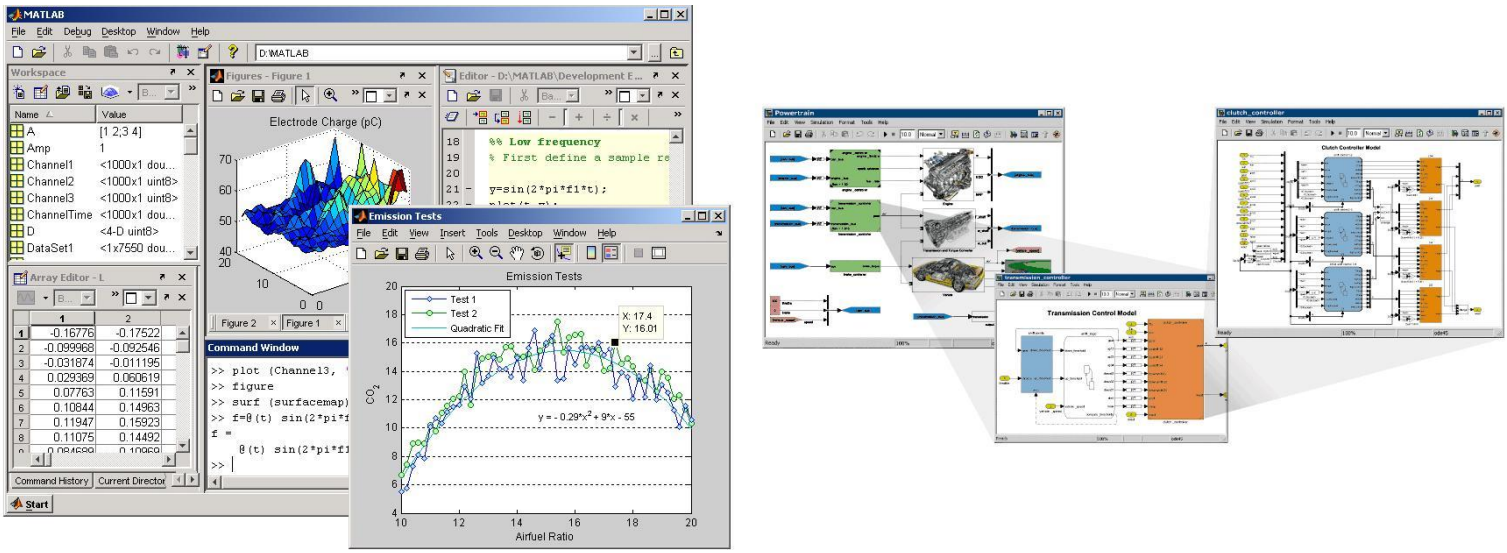

Figura 15 – Interface do usuário do MATLAB e Simulink.

Neste trabalho, conforme citado anteriormente, será feito o uso do MATLAB para construção e manipulação do controlador Múltiplos agentes. Será utilizada a biblioteca RWTH (apresentado a seguir) no desenvolvimento do controlador que será apresentado mais adiante no trabalho.

3.4.RTWH

Este toolbox é desenvolvido para controlar os robôs NXT com o MATLAB através de uma conexão sem fios Bluetooth ou via USB. Este software é um produto de fonte aberta livre e está sujeito à Licença Pública Geral GNU (GPL). Este conjunto de ferramentas fornece funções MATLAB para interagir com um robô diretamente. A principal vantagem dessa biblioteca é que permite a combinação de aplicações para robôs com operações matemáticas complexas e visualizações no MATLAB. Esta caixa de ferramentas abre possibilidades ilimitadas para fornecer a inteligência artificial de robôs e outras melhorias usando os recursos de múltiplas MATLAB e cálculos para processamento de sinal digital.

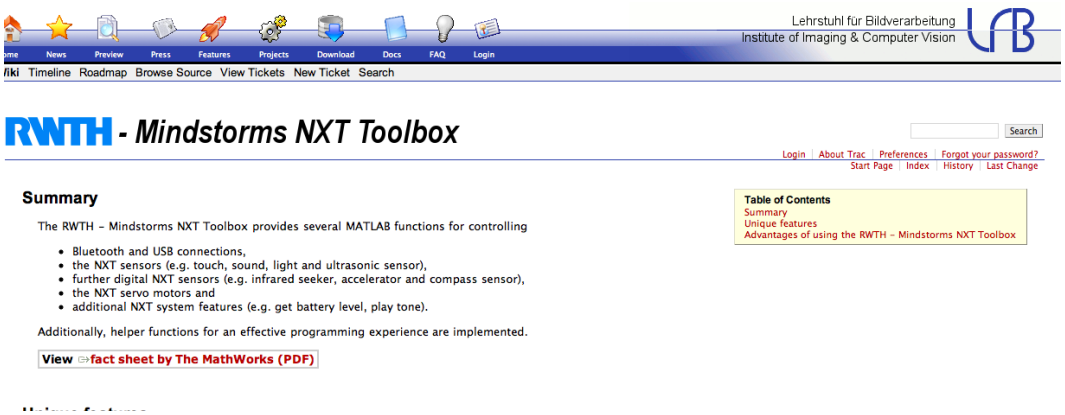

Figura 12 – Site da biblioteca RWTH

4.Conceitos Básicos

4.1.Introdução

Agentes são o paradigma utilizado no desenvolvimento de aplicações de que visam a atender a uma funcionalidade, eles percebem o ambiente através de sensores e através de atuadores agem neste ambiente.

Dentre as muitas características que um agente pode possuir, a autonomia é
aquela que determina que algoritmos podem ser classificados aqueles agentes. A autonomia é a capacidade do agente seguir seus objetivos automaticamente, isto é, sem interações ou comandos que venham do ambiente.

Outra característica resonsável por aumentar a capacidade autônoma de um agente, é a inteligência, agentes capazes de aprender são considerado autônomo.
A inteligência é caracterizada pelo aprendizado juntamente com a base de conhecimento e o raciocínio.

Um agente inteligente deve interagir com o meio para realizer seus objetivos: deve ser capaz de obter informações do ambiente, tomar decisões baseadas nesses dados e tomar uma ação específica com base nas decisões.

Um campo emergentena área de pesquisa de Inteligência Computacional, com o objetivo de desenvolver sistemas complexos envolvendo *múltiplos agentes* e os mecanismos de coordenação entre eles é o chamado Sistemas Multi-Agentes (SMA). Um SMA pode ser definido como um grupo de agentes autônomos, interagindo entre si e presents no mesmo ambiente, que é percebido através de sensores, e onde eles atuam realizando ações. Este tipo de sistema vem encontrando uma
grande variedade de campos de aplicação(como sera apresentado numa seção a seguir).

4.2.Sistemas Multi-Agentes

Os Sistemas Multi-Agentes são uma área emergente da Inteligência Artificial que fornece os princípios para construção de sistemas complexos envolvendo múltiplos agentes e os mecanismos de coordenação entre eles. Um SMA pode ser definido como um grupo de agentes autônomos, interagindo entre si e compartilhando um mesmo ambiente, que é percebido através de sensores, onde eles agem realizando ações.
Os benefícios trazidos pela aplicação de SMA são diversos. Primeiramente, através da computação paralela, vários agentes podem trabalhar em
equipe para quebrar uma tarefa em tarefas menores e acelerar sua conclusão. Além disso, agentes podem compartilhar experiências, podem apenas observar os mais habilidosos e aprender, os mais inteligentes podem servir de professores para outros agentes, etc. Os SMA também podem fornecer alto grau de escalabilidade,
através da inclusão de novos agentes quando necessário, por exemplo, agentes podem assumir as atividades de outros agentes em casos de falha. A interação entre os agentes está diretamente ligada inteligencia do sistema, a melhor forma de criar máquinas inteligentes pode ser através da “sociabilidade”entre os

agentes.

Divide-se em quatro grandes subáreas de pesquisa dentro do aprendizado de SMA's: 3 prescritivas e uma descritiva. As três linhas de pesquisa prescritivas trabalham como os agentes devem aprender. A primeira, Inteligência Artificial Distribuída estuda problemas de controle distribuído. Geralmente, um agente central controla múltiplos agentes, mas não define as ações que cada um deve executar. Define apenas um procedimento generico que converge para uma "política ótima". A segunda ("*Equilibrium Agenda*") analisa o equilíbrio entre as estratégias de diferentes agentes. A terceira ("*AI Agenda*") estuda a escolha da melhor estratégia para um agente, considerando uma classe fixa de agentes no jogo, ou a definição de um agente ótimo para um determinado ambiente. Essa estrategia não utiliza totalmente alguns dos conceitos de Multi-Agentes, como a coordenação entre os agentes. A descritiva, explica como humanos aprendem no contexto de outros humanos.

O SMA tamém pode ser visto como a interseção de Computação Distribuída e Inteligência Artificial, ou seja um um sub-campo da Inteligência Artificial Distribuída (DAI). Na Computação Distribuída utiliza diversos processadores que compartilham dados (mas não o controle) com foco em uma paralelização ou em sincronização de tarefas. Na DAI, além dos dados, o controle também é distribuído, com o objetivo da solução do problemas na coordenação e comunicação. Nos Sistemas Multi-Agentes, o foco é na coordenação do comportamento dos agentes e nao da informação dos dados.

Embora o comportamento dos agentes em um ambiente multi-agente possa ser pré-programado, em algumas das situações é necessário algum tipo de aprendizado on-line, para que de uma maneira gradativa haja uma melhora no desempenho dos agentes.

4.3.Conceitos e Taxonomia

De acordo com os seguintes fatores: tipo da tarefa, ambiente, grau de homogeneidade do aprendizado, conhecimento inicial dos agentes, etc, pode-se determinam a categoria de uma aplicação do SMA.

O ambiente de uma aplicação SMA pode ser estático ou dinâmico, onde mudanças no estado podem ocorrer ao longo do processo de interação entre os agentes e dos agentes com o estado.

Existem três categorias principais em relação ao tipo da tarefa envolvida no problema. A primeira é a das aplicações *totalmente competitivas*, que sempre têm um vencedor e um perdedor no final. Neste tipo de problema, um dos princípios que pode ser aplicado é o do *minimax* ("minimizar a máxima perda possível"). A segunda categoria é composta pelas aplicações *totalmente cooperativas*. Neste caso, o objetivo é maximizar o retorno comum dos múltiplos agentes. Neste tipo de aplicação, os retornos são positivamente repassados a todos os mebros da equipe. Finalmente, a terceira categoria é a de tarefas mistas, onde nenhuma restrição é imposta aos retornos dos agentes.

SMAs com inteligência tem uma caracteristica importante que é a homogeneidade do aprendizado. O aprendizado dos múltiplos agentes pode ocorrer de três formas: homogêneo, com todos aprendendo igualmente; heterogêneo, onde cada agente aprende de forma independente, isto faz aumentar o espaço de busca e híbrido, onde cada grupo de agentes aprende de uma forma

homogênea, mas cada grupo de agentes com uma especialização.

Um dos pontos mais importantes dos SMA é o grau de comunicação entre os agentes. De uma maneira geral, existem duas formas de comunicação entre os agentes: direta e indireta. Na indireta, um agente central pode ter acesso a tudo e repassar aos demais as informações que podem ser colocadas em uma área de transferência que todos ou o pode-se deixar diretrizes que outro agente pode usar no para realizarem suas tarefas.

Combinando o grau de comunicação e a homogeneidade do aprendizado,
Stone e Veloso (2000) sugerem uma definição dos principais grupos e o nível de complexidade de um SMA. Conforme mostra a figura 1, quanto maior a heterogeneidade (ou maior especialização) e maior grau de comunicação, mais complexo é o sistema.

Em relação ao nível de homogeneidade do aprendizado, o aumento da complexidade com a maior especialização ocorre porque cada agente ou grupo de
agentes aprenderá utilizando uma dinâmica diferente.

Com elevado grau de comunicação, um SMA pode chegar a ser equivalente a um sistema de grande complexidade, onde um único agente, tomando todas as decisões e controla os demais como se fossem “escravos”. Com isso, não há nenhum tipo de distribuição ou partição de tarefas.
Na construção de um sistema com múltiplos agentes existem diversas categorias, pautadas em diferentes definições e técnicas de aprendizado que devem ser escolhidas.

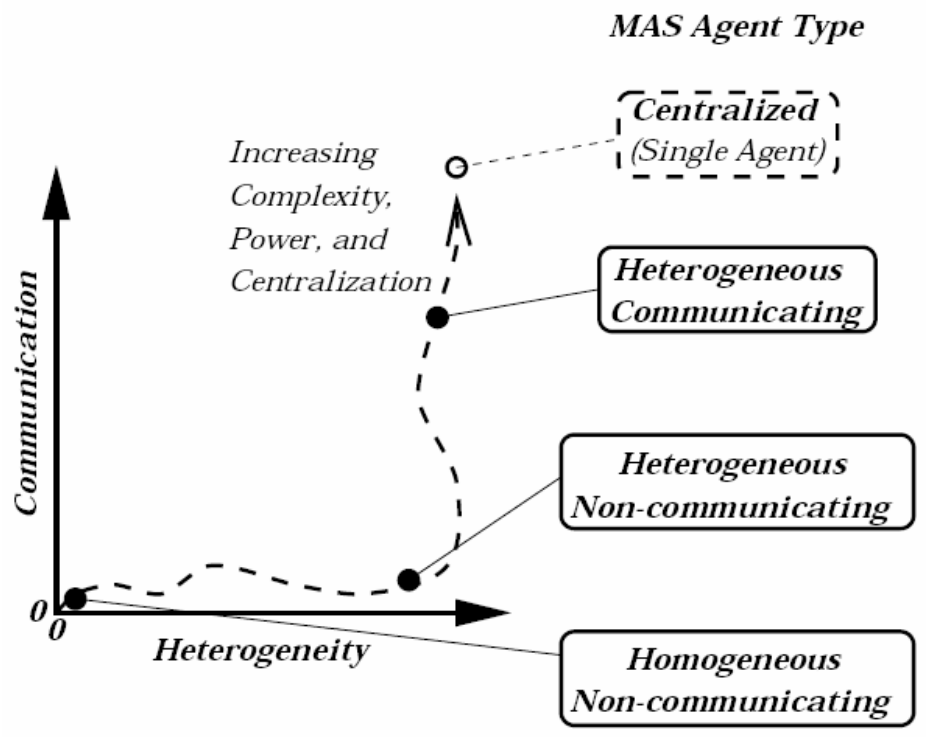

Figura 17: Categorias de SMA de acordo com o nível de comunicação e heterogeneidade dos agentes (Stone e Veloso, 2000).

4.4.Aspectos Gerais de todos os Sistemas Multi-Agentes

- Estabilidade x Evolução: é necessário definir se os agentes são

estáveis ou se evoluem suas estratégias(apenas na em ambientes dinâmicos). Em aplicações onde há a competição com evolução, podem ocorrer uma busca sem fim pela melhor estratégiae nunca se estabilizado em um bom comportamento, pode-se ter também um problema de avaliação da estratégia corrente onde não é possível afirmar se o agente melhorou seu comportamento ou se foi por causa da piora de seu competidor.

- Cooperação x Competitividade: deve-se definir se os agentes irão ajudar um ao outro para chegar ao objetivo (cooperativos) ou se irão competir entre si, considerando apenas seus próprios objetivos. Podem ocorrer situações de "soma-zero", onde os agentes conquistam seus objetivos se opondo às estratégias dos demais.

- Conhecimento inicial dos agentes: os agentes podem possuir ou não algum conhecimento a priori do modelo das tarefas que devem ser cumpridas.

- Divisão do Reforço (*Credit Assignment* ou *Reinforcement Sharing*):Em sistemas com inteligencia, as vezes apenas o ambiente não é sufiviente para gerar reforço. Logo, se faz necessário a utilização de algum mecanismo de divisão do reforço. Entre as principais formas de divisão de reforço podem ser citados: *Global reinforcement* (reforço do "time" é dividido igualmente a todos), *Local reinforcement* (atualização de reforço feita somente com base no desempenho de cada agente individualmente), *Local reinforcement + social reinforcement*(melhorado com alguns tipos de "*reforço social*", por exemplo: *observational reinforcement* ou *vicarious reinforcement*).

4.5.Caracteristicas de Sistemas Homogêneos

4.5.1Sistemas Homogêneos com Comunicação (Sistemas com "Agente Central")

Sistemas onde os agentes são homogêneos (possuem as mesmas características), e podem se comunicar diretamente e sem restrições, muitas vezes são consideradas sistemas de um agente único. Neste tipo de sistema, um agente central armazena todas as informações e coordena as ações dos demais, que não possuem nenhuma inteligência. Embora possa parecer que sistemas com apenas um agente inteligente sejam mais fáceis de compreender, a distribuição do controle permite a criação de agentes simples, podem ajudar na solução na quebra da soluçao de problemas complexos.

- Nível de paralelismo desejado: neste tipo de SMA o objetivo desejado é a exploração paralela do ambiente. O nível de exploração em paralelo será definido pela quantidade de agentes no ambiente. Quanto mais agentes, maior será o volume de tarefas executadas em paralelo.

- Grau de Controle do Agente Central: Variável, na maioria das vezes, neste tipo de SMA a relação dos agentes com o agente central é do tipo mestre-escravo, ou seja, o agente central coordena todas as ações e informações dos demais agentes.

4.5.2 - Sistemas Homogêneos sem Comunicação

- Agentes reativos x deliberativos: definição do grau de "raciocínio" dos agentes.

Quando reativos, os agentes se limitam apenas arecuperar comportamentos já existentes de acordo com seus reflexos. Por exemplo, nas aplicações em que agentes precisam fugir de obstáculos e, para isso, convertem os dados obtidos através de seu sensor para ações de movimento. Já, se deliberativos, os agentes assumem que cada um escolherá suas ações para concluir seus objetivos.teoria dos jogos para encontrar pontos de equilíbrio e decidir como agir. Existe também a possibilidade de combinar “pensamentos” reativos e deliberativos.

- Perspectiva Local x Global: definiçao de quanto os agentes receberão de informações. Em alguns casos se todos os agentes tenham uma perspectiva global e usem o mecanismo de aprendizagem, isso pode levá-los a escolher sempre as mesmas ações, logo com menos informação, melhores resultados podem ser obtidos.

- Modelagem dos estados dos outros agentes: muitas vezes é necessário modelar o estado interno de outros agentes com objetivo de prever suas ações. Mesmo que os agentes conheçam os objetivos e estruturas dos demais, eles podem não conhecer suas futuras ações.

- Como afetar os outros agentes: Como não existe comunicação, os agentes podem afetar os demais de forma direta ou indireta. Diretamente, se eles forem ser percebidos pelos sensores alheios ou alterar o estado dos outros. Indiretamente, eles podem afetar o resto do grupo através de dois tipos modos, ativamente se o agente altera o ambiente de forma que as mudanças sejam captadas pelos outros sensors ou passivamente, sea alteração no ambiente ocorre com a finalidade de mudar o efeito da ação do outro agente.

4.6.Caracteristicas de Sistemas Heterogêneos

- Papéis e Funções: quando os agentes possuem diferentes habilidades, eles devem ser organizados como um time, onde cada um desempenha suas funções específicas. Isso vale para sistemas onde existe ou não comunicação entre os agentes. Em ambientes dinâmicos, pode ainda ser necessário que cada agente tenha um determinado papel dependendo da situação em que se encontra. Uma das subáreas de pesquisa dentro de SMA é justamente como um agente pode assumir novas funções em ambientes em constante evolução.

- Gerenciamento de recursos: em algumas aplicações, agentes possuem recursos limitados para serem compartilhados, logo havendo comunicação entre os agentes é possíveis a coordenação do tempo das atividades os gastos dos recursos e o controle de cronogramas .

4.6.1 - Sistemas Heterogêneos sem Comunicação

- Modelagem dos objetivos, ações e conhecimento dos outros agentes: como os agentes não são homogêneos, não basta modelar seus estados internos para prever suas ações, a previsão desta só pode ser feita modelando os objetivos, ações e conhecimento dos demais, tornando o problema extremamente complexo e caro. A alternativa mais barata é tentar deduzir as ações dos agentes observando suas ações.

- “Convenções sociais”: mesmo não havendo comunicação entre os agentes, existem formas de se chegar a “acordos” ou escolhas coincidentes, por exemplo

buscando pontos prováveis de referências no ambiente.

4.6.2 - Sistemas Heterogêneos com Comunicação

- Padrões de Comunicação: existe a necessidade de definição do padrão que será usada na comunicação entre os agentes para troca de informações que podem ser pré-definidos ou elaborados para attender as especificações dos projetos.

- Planejamento das ações de comunicação: consideramos que uma comunicação seja feita da mesma forma que uma ação qualquer, ou seja, cada transmissão de informação pode possuir pré-condições e efeitos como uma ação teria. Em um SMA, o efeito da comunicação pode ter como objetivo a alteração da crença de um agente em relação ao estado de outro(s) agente.

- Acordos e Compromissos: havendo comunicação entre os agentes, eles podem realizar acordos de cooperação para alcançarem objetivos comuns. Os acordos e compromissos podem ser de vários tipos, dentre eles: compromissos internos (o próprio agente se compromete a fazer algo), compromisso com outro agente e compromissos coletivos, onde o agente se dispõe a desempenhar uma determinada função.

4.7.Aplicações de Sistemas Multi-Agentes

Conforme já discutido, os sistemas Multi-Agentes vêm encontrando uma grande potencial de variedade de aplicações. A seguir, serão apresentados os principais campos de pesquisa.

4.7.1 - Controle Distribuído

Nesse tipo de aplicações, um conjunto de agentes autônomos interage em paralelo, através de ações de controle, com um mesmo objetivo. De maneira geral, tem-se um problema de controle distribuído sempre que o ambiente é um processo de controle e os agentes são os controladores deste processo. São tipicamente aplicações que envolvem a cooperação entre os agentes, dado que o objetivo é o mesmo. Entre os principais problemas abordados estão: controle de processos, controle elétrico de redes de energia e controle de sinais de trânsito.

4.7.2 - Times Robóticos

Talvez por ser a mais natural, instintiva e de compreensão geral a criação de times robóticos é considerada uma das aplicações mais exploradas dos SMA. De maneira geral, existem duas dimensões, real ou simulado., nas quais agentes podem usar qualquer tipo de algoritmo, podem aprender para melhorar a forma de se comportar e adquirir habilidades.
Uma aplicação largamente explorada como benchmark é o jogo da presa/predador, neste jogo, vários agentes predadores devem capturar a presa convergindo ao seu redor prendendo-a.
Há também problemas de controle de navegação, onde cada robô deve encontrar seu caminho a partir de um determinado ponto de origen, até seu objetivo, mesmo que para isso precise passar por obstáculos e evitar que a interferência de outros robôs o prejudique. Entre outras aplicações relacionadas à navegação estão: a localização e busca de objetos, a cobertura de superfícies da forma mais ampla possível, e a exploração de ambientes com o objetivo de

informações possíveis atraves de sensors.
Um dos jogos mais populares para SMAs e que merece uma atenção especial é o futebol robótico que ja ganhou muito espaço na mídia.

Figura 18 – Futebol de Robôs

4.7.3 - Sistemas de Negociação e Trading

Existem diversos mercados eletrônicos que permitem a comercialização de produtos entre os agentes, seja através de negociações ou leilões. Um exemplo de benchmark para este tipo de aplicação é o *Trading Agent Competition*, aplicação onde os agentes devem organizar viagens negociando passagens e reservas de hotel (Wellman et al., 2003).
Em algumas aplicações, os agentes, representando uma determinada empresa ou indivíduo, apenas cooperam em tarefas distintas do processo de um sistema de negociação multi-agente proposto, por exemplo, com quatro agentes, cada um desenvolvendo capacidades específicas – gerar o sinal de compra, definir o preço de compra, gerar o sinal de venda e definir o preço de venda pode-se ter uma aplicação que busca maximizar seus retornos interagindo em paralelo com os mercados.

4.7.4 - Gerenciamento de Recursos

Nesse tipo de atividade, os agentes sempre agem de forma cooperativa e possuem basicamente 2 tipos de comportamento: *gerentes dos recursos* - cada um gerencia um recurso e aprende a melhor forma de responder a solicitações de uso deste, buscando otimizar uma determinada medida de desempenho; *clientes dos recursos:* os agentes aprendem a selecionar os recursos de maneira ótima. Como exemplo podemos citar aplicações de balanceamento de carga e o problema de programação de elevadores. Exemplos de medidas de desempenho são: tempo de processamento de rotinas, tempo de espera por recursos, utilização e disponibilidade de recursos, distribuição justa de recursos aos clientes, etc.

5.Controle

5.1.Introdução

Para atingir o objetivo o controle proposto foi elaborado um time robótico com um sistema heterogêneo com comunicação, com características de estabilidade(não há mudança na base de regras do conhecimento), baseado em cooperação(os agentes expolram o ambiente de forma conjunta, a fim de encontrarem o objetivo de forma mais rápida possiva e transmitir a olocalização para todos) e com conhecimento inicial de todas as tarefas que devem ser cumpridas. O sistema é composto de três rotinas principais, uma que cada agente realiza individualmente para buscar o objetivo e navegar pelo ambiente, uma segunda realizada para identificar qual o agente foi o responsável por alcançar o objetivo e faz o cálculo da nova, a terceira utiliza os valores da etapa anterior e navegar o(s) outro(s) agente(s) até o objetivo. As rotinas serão executadas sequencialmente para que o Robô atinja seu objetivo, casa haja mais de dois agentes a segunda e terceira rotina serão executadas tantas vezes quantos forem o número de agentes.

5.2.Rotina de Navegação e Busca do Objetivo

Esta rotina é a responsável pela forma como o agente se comunica e é influenciado pelo ambiente, nela o robô utiliza os dados obtidos pelos sensores e busca em sua base de regras (estável – conhecimento estático e a priori das ações que deve tomar para cada estímulo do ambiente) para garantir que o robô termine a rotina encontrando um objeto de cor vermelha que emite uma determidada frequência sonora. Para alcançar o objetivo o robô pode se deparar com paredes (devendo girar para um lado e continuar sua busca) ou pode se afastar da fonte sonora (valor obtido no sensor de som é menor que o filtro de media de valores de som implementado), todos esses eventos são devidamente tratados pelo algoritimo que é executado de forma individualmente por cada agente que teoricamente é realizado de forma paralela. Ao final da rotina se tem a informação de qual agente encontrou o objetivo e qual foi a sua direção inicial de partida e o quanto foi percorrido em cada um dos eixos(informações

importantes para a próxima rotina que fará o cálculo dos deslocamentos dos outros agentes até o objetivo.

➔ Setando parâmetros iniciais como motores, sensores e variáveis de estados iniciais

```
NXT = COM_OpenNXTEx('bluetooth', '001653127939',
'bluetooth1.ini');%robo1 56
NXT3 = COM_OpenNXTEx('bluetooth', '00165310EA24',
'bluetooth3.ini');%robo3 60

%setando os parametros do motor
myMotors                  = NXTMotor();
myMotors.Port             = [MOTOR_A; MOTOR_B];
myMotors.Power            = 40;
myMotors.SmoothStart      = true;
myMotors.ResetPosition();
myMotors.SpeedRegulation = false;

%setando os parametros para giro
Motor1                    = NXTMotor();
Motor1.Port               = MOTOR_A;
Motor1.Power              = 50;
Motor1.SmoothStart        = true;
Motor1.ResetPosition();
Motor1.SpeedRegulation = false;

Motor2                    = NXTMotor();
Motor2.Port               = MOTOR_B
Motor2.Power              = -50;
Motor2.SmoothStart        = true;
Motor2.ResetPosition();
Motor2.SpeedRegulation = false;

%setando os parametros para giro
Motor3                    = NXTMotor();
Motor3.Port               = MOTOR_A
Motor3.Power              = -50;
Motor3.SmoothStart        = true;
Motor3.ResetPosition();
Motor3.SpeedRegulation = false;

Motor4                    = NXTMotor();
Motor4.Port               = MOTOR_B;
Motor4.Power              = 50;
Motor4.SmoothStart        = true;
Motor4.ResetPosition();
Motor4.SpeedRegulation = false

COM_SetDefaultNXT(NXT3); %seta o agente NXT3 como corrente

%Setando os sensores
OpenSound(SENSOR_1, 'DB');
OpenNXT2Color(SENSOR_2, 'FULL');
OpenUltrasonic(SENSOR_3, 'ACTIVE');
```

```
OpenUltrasonic(SENSOR_4, 'ACTIVE');

%setando as flags
searching = true;

%variaveis para filtro de som
sound0 = 0;sound1 = 0;sound2 = 0;
sound3 = 0;sound4 = 0;mediaSound = 0;

direcao = 1;
direcaoIni = 1;
% Significado dos valores da variavel de direcao do movimento
% 1-frente
% 2-direita
% 3- tras
% 4-esquerda
```

➔ Para facilitar a navegação o ambiente foi dividido em dois eixos cartesianos e o robô so pode andar sobre esses eixos , logo o agente anda ou para a posição positiva do eixo x (frente), oupara a posição positiva do eixo y(direita), ou para posição negativa do eixo x (trás) ou para a posição negativa do eixo y(esquerda). Como visto na figura 19.

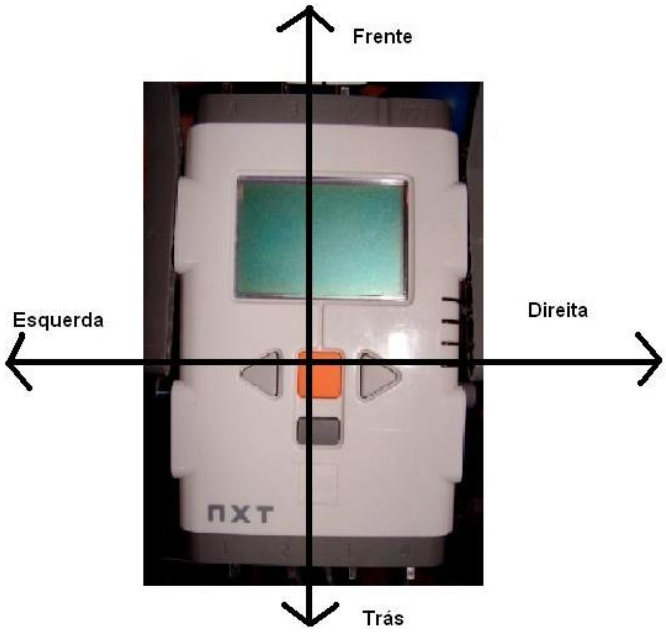

Figura 19 – eixos de movimentação do agente

```
%variaveis para acumular o valor percorrido em cada direcao
df =  0;
dd = 0;
de = 0;
agente = 0;
segchance = 0;
```

➔ Além disso o quanto o agente percorreu por cada eixo é guardado em variáveis (df = quanto andou pra frente menos quanto andou para trás, dd = quanto percorreu para a direita e de = quanto percorreu para esquerda) esses valores serão importantes para ou passer como imformação para outro(s) agentes a posição final ou para o calculo de sua rota até a posição final.

```
%ligando os motores
myMotors.SendToNXT();
```

➔ A seguir começa a rotina pela busca do objeto. Primeiramente avalia-se os sensores de distância para avaliar se existe algum objeto proximo. Se existir entao atribuirá-se um valor a variavel **dist** dizendo fazendo referência a posição do objeto (1 - na esquerda do agente , 2 - na direita do agente e 3 – na frente do agente). Caso a variável tiver valor diferente de zero sera avaliada a cor do objeto e se esta for vermelha e o nivel sonoro acima do esperado o agente encontrou seu objetivo ,seta-se uma variavel com o valor de identificaçao do agente e encerram-se as buscas, caso contrário o agente desviará da obstáculo girando para o lado oposto deste.

```
 %loop de busca pela presa
while ( searching )

  color = GetNXT2Color(SENSOR_2);
  disp(sprintf('The detected color is %s', color));

%flag dos sensores de distancia
    dist=0;%nao ha nada a frente

    a = GetUltrasonic(SENSOR_3);
    if a < 24
        dist = dist +1;%sensor da esquerda indica presenca de
objeto
    end

    b = GetUltrasonic(SENSOR_4);
    if  b < 24
        dist = dist +2;%sensor da direita indica presenca de
objeto
    end
    disp(sprintf('ultra 3 =%d ultra4 = %d', a,b));

    if(dist ~= 0)
        myMotors.Stop();

           soundCorrente = GetSound(SENSOR_1)
           if(soundCorrente > 339)
               disp(sprintf('Som ok! %d',soundCorrente));

               pause(1);
                if strcmp(color, 'RED')
                    searching = false;%achou a presa
                    agente = 1;
                    disp(sprintf('Achou!'));
                end

               myMotors.SendToNXT();
               aux = myMotors.ReadFromNXT().Position;
               while ((myMotors.ReadFromNXT().Position) < aux +
40)
                    disp(sprintf('Aproximou'));
               end
```

```
            else

                %o que esta na frente È uma parede
                %gira para direita
```

➔ Toda vez que o agente realizar uma operação de giro é necessário guarda quanto ele percorreu naquela direção(valor armazenado pelos hodômetros) e qual é a direção corrente.

```
% guarda quanto percorreu naquela direcao

          space = myMotors.ReadFromNXT().Position;
             if( direcao == 1)
                  direcao = 4;
                  df = df + space;
              elseif( direcao == 2)
                  direcao = 1;
                  dd = dd + space;
              elseif( direcao == 4)
                  direcao = 3;
                  de = de + space;
             else
                  direcao = 2;
                  df = df - space;
             end

            myMotors.ResetPosition();

            %gira para esquerda
            if ( searching )
                  Motor1.ResetPosition();
                  Motor2.ResetPosition();
                  Motor1.SendToNXT();
                  Motor2.SendToNXT();

                  while ((Motor1.ReadFromNXT().Position) < 250)
                      data = Motor1.ReadFromNXT();
                      disp(sprintf('D1 position %d',
(data.Position)));
                  end
                  Motor1.Stop();
                  Motor2.Stop();
                  Motor1.ResetPosition();
                  Motor2.ResetPosition();

                  pause(1);
                  myMotors.SendToNXT();
            end

             end
    end
 if(soundCorrente > 339)
             segchance = segchance + 1;
              myMotors.SendToNXT();
                 aux = myMotors.ReadFromNXT().Position;
                 while ((myMotors.ReadFromNXT().Position) < aux +
40)
                      disp(sprintf('Aproximou'));
```

```
              end
 end
   if(dist == 2)
       if segchance > 0
        %o sensor da esquerda esta de frente para um objeto
        % guarda quanto percorreu naquela direcao
       ...
       ...
       ...
         %gira para direita
      ...
      ...
      ...
       end
         segchance = 0;
       end
   elseif(dist == 1)
       if segchance > 0
         %o sensor da direita esta de frente para um objeto
         % guarda quanto percorreu naquela direcao
         ...
       ...
       ...
  %gira para esquerda
      ...
       ...
       ...
    segchance = 0;
    else
        % nao tem nenhum objeto proximo
        soundCorrente = GetSound(SENSOR_1);
        if soundCorrente < mediaSound %esta se afastando da presa
          % gira pra um lado
          %...
            ...
            ...
         end

        %filtro de media do valor do sensor de som pega os ultimos
5 valores e fornece amédia.

        sound0 = soundCorrente;
        sound1 = sound0;
        sound2 = sound1;
        sound3 = sound2;
        sound4 = sound3;
        mediaSound = (sound0 + sound1 + sound2 + sound3 + sound4
)/5 ;
    end
end

myMotors.Stop();
```

5.3.Rotina de Identificação e Cálculo das Novas Coordenadas

Essa rotina é responsável por setar os valores das variáveis de deslocamento que serão utilizadas na no cálculo das novas coordenadas (d1f,

d1dir, d1esq, d2f, d2dir e d2esq - que são respectivamente os valores de quanto o agente que encontrou o objetivo percorreu pra frente pra direita e para esquerda e quanto o outro agente percorreu nessas mesmas direções, cada um em seu sistema de eixos). Ela guarda também o valor das direções de partida dos agentes que estão sendo analisados e a posição corrente do agente que terá que navegar até o objetivo.

A rotina abaixo é um exemplo para um sistema de dois agentes.

```
%pos1 = direcao;%
pos1;% posicao de partida do agente 1 agente q encontrou o objeto
pos2 ;% posicao de partida do agente 2
pCorrente % posicao na qual se encontra o agente 2

if agente == 1

      d1f = df;
      d1dir = dd;
      d1esq =de;

      d2f = dfB;
      d2dir = ddB;
      d2esq = deB;

      pos1 = direcaoIni;
      pos2 = direcaoIniB;
      pCorrente = direcaoB;

elseif agente == 2

      d1f = dfB;
      d1dir = ddB;
      d1esq =deB;

      d2f = df;
      d2dir = dd;
      d2esq = de;

      pos1 = direcaoIniB;
      pos2 = direcaoIni;
      pCorrente = direcao;
end
```

➔ O cálculo da nova rota se baseia na seguinte idéia:

Tomemos o primeiro caso: o agente que achou partiu da posição “frente” e o outro agente partiu da posição “direita” (figura 20).

Figura 20 – Disposiçao dos Agentes

Imaginemos que o agente1 que encontrou a caixa vermelha tenha percorrido d1f = 500 , d1dir =1000 , e d1esq = 800 , enquanto o agente 2 percorreu d2f = 1000, d2dir = 500 , e d2esq = 800 como indica a figura 21.
Ou seja, o agente1 partindo da direção frente andou 500 unidades para frente e 200 unidades para direita e o agente2 partindo da direção direita percorreu 100 unidades para frente e 300 unidades para direita .

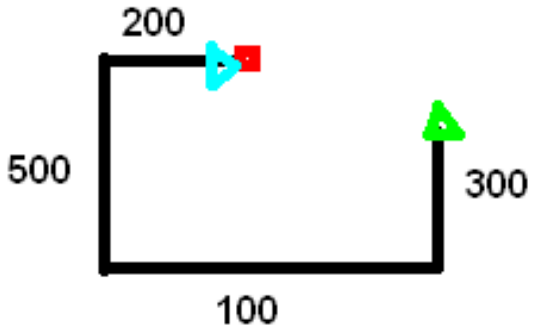

Figura21 – Deslocamento dos agentes.

Portanto o vetor resultante que queremos é o quanto o agente2 deve percorrer para chegar na mesma posição que o agente1.
Vemos os calculus realizados:

```
df2= (d1dir-d1esq)-d2f;
dd2= d2esq -d2dir- d1f;
de2= d1f -d2esq  +d2dir;

df2= (1000-800) - 1000;
dd2= 800 - 500 - 500;
de2= 500 - 800 + 500;

df2= -800;
dd2= -200;
```

de2= 200;

Logo o agente2 deve percorrer 800 unidades para trás e 200unidades para esquerda(Figura 22).

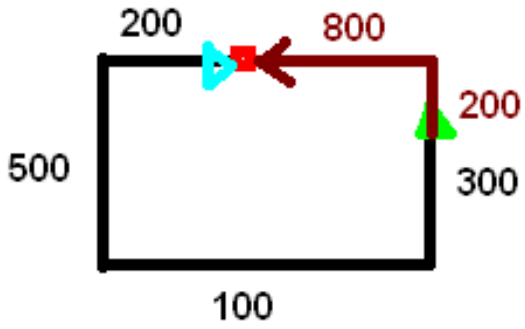

Figura22 – Vetor Resultante.

```
achou partiu da posicao frente e o outro partiu da posicao direita

%//////////////////////////////////
%Calculo das novas coordenadas
%//////////////////////////////////
if pos1 == 1
    %frente
    if pos2 == 2
%quem achou partiu da posicao frente e o outro partiu da posicao
direita
%quem anda e quem saiu da direita
        df2= (d1dir-d1esq)-d2f;
        dd2= d2esq -d2dir- d1f;
        de2= d1f -d2esq  +d2dir ;

    elseif pos2 == 4
%quem achou partiu da posicao frente e o outro partiu da posicao
esquerda
%quem anda e quem saiu da esquerda
        df2= (d1esq-d1dir) - d2f;
        dd2= d1f - (d2dir-d2esq);
        de2= (d2dir-d2esq) - d1f;
    end
end
if pos1 == 2
%quem achou partiu da posicao direita e o outro partiu da posicao
esquerda
%quem anda e quem saiu da esquerda
    if pos2 == 4
        df2= -d1f-d2f;
        dd2= (d2dir-d2esq) - (d1dir-d1esq);
        de2= (d1dir-d1esq) - (d2dir-d2esq);
    end
end
```

5.4.Navegação até o ponto Final

Esta rotina consiste em girar o segundo agente até a direção inicial sendo que o numero de graus é calculado de acordo com a posição corrente dele e apartir daí navegar utilizando os dados calculados na rotina anterior até o ponto onde foi encontrado o objetivo. A navegação consiste em percorrer o vetor resultante do movimento na direção x e depois na direção y do eixo cartesiano do agente.

```
%////////////////////////////////////////////////////
%Gira o agente 2 para a msm direÁao da posicao inicial
%////////////////////////////////////////////////////

nVezes = 0 ;
%pos2 == pinicial do agente 2
if pCorrente ~= pos2
    if pCorrente < pos2
        nVezes = pos2 - pCorrente;
    else
        if pCorrente == 4
            nVezes = pos2;
        elseif pCorrente == 3;
            if pos2 == 1
            nVezes = 2;
            elseif pos2 == 2
                nVezes = 3;
            end
        elseif pCorrente == 2;
            nVezes = 3;
        end
    end

        Motor1.ResetPosition();
                Motor2.ResetPosition();
                Motor1.SendToNXT();
                Motor2.SendToNXT();

                while ((Motor1.ReadFromNXT().Position) <
nVezes*485)
                    data = Motor1.ReadFromNXT();
                    disp(sprintf('D1 position %d',
data.Position));
                end
                Motor1.Stop();
                Motor2.Stop();
                Motor1.ResetPosition();
                Motor2.ResetPosition();
```

->Por exemplose estou na posição “frente” e quero ir para a posição “esquerda” eu preciso de N *giros sendo n = 3;

```
end
%////////////////////////////////
```

```
% percorrendo as novas coordenadas
%//////////////////////////////////
virou = false;
if df2 < 0
    virou = true;
    %gira para tras
                Motor3.ResetPosition();
                Motor4.ResetPosition();
                Motor3.SendToNXT();
                Motor4.SendToNXT();

                while ((Motor4.ReadFromNXT().Position) < 1000)
                    data = Motor4.ReadFromNXT();
                    disp(sprintf('D2 position direita? %d',
(data.Position)));
                end
                Motor3.Stop();
                Motor4.Stop();
                Motor3.ResetPosition();
                Motor4.ResetPosition();

                df2 = df2 * (-1);
end

pause(0.5);
if df2>0
%anda ate o odometro chegar no valor especificado com mymotors
    myMotors.SendToNXT();
    while ((myMotors.ReadFromNXT().Position) < df2)
    data = myMotors.ReadFromNXT();
    disp(sprintf('Voltando primeira parte frente at position %d',
(data.Position)));
    end
    myMotors.Stop();
end
pause(0.5);

    if (virou)
    %gira para frente
                Motor1.ResetPosition();
                Motor2.ResetPosition();
                Motor1.SendToNXT();
                Motor2.SendToNXT();

                while ((Motor1.ReadFromNXT().Position) < 1000)
                    data = Motor1.ReadFromNXT();
                    disp(sprintf('D2 position direita? %d',
(data.Position)));
                end
                Motor1.Stop();
                Motor2.Stop();
                Motor1.ResetPosition();
                Motor2.ResetPosition();



    end
    pause(0.5);
lateral = 0;

if dd2 > 0
```

```
    lateral = dd2;
    %gira para direita
      ...
      ...
      ...
elseif de2 >0
    lateral = de2;
    %gira para a esquerda
...
...
...
end

myMotors.ResetPosition();
pause(1);
    %anda ate o odometro chegar no valor especificado
    myMotors.SendToNXT();
    while ((myMotors.ReadFromNXT().Position) < lateral)
    data = myMotors.ReadFromNXT();
    disp(sprintf('Voltando segunda parte at position %d',
(data.Position)));
    end
    myMotors.Stop();
    %////////////////////////////////////////////
    % agente 2 chegou na msm posiÁao q o agente 1
    %////////////////////////////////////////////
```

6.Resultados

Foram realizados diversos experimentos com os agentes, em ambientes diferentes e aleatórios(ambiente controlado no laboratório e corredor de apartamento). Como não se contou com modelagem computacional adequada para simulações virtuais, estes foram realizados fisicamente diretamente no próprio hardware, assim como todas as etapas de desenvolvimento.

Os mundos eram compostos de paredes (delimitadores do ambiente) e uma caixa de plástico da cor vermelha que emitia um som, a ser considerado como o objetivo da da busca dos agentes NXT deverá traçar, que deveria navegar pelo mundo e procurar seu objetivo.

Figura 23 – Um dos ambientes de testes do projeto.

Depois de repetidas rotinas de testes, o resultado foi satisfatório, com uma boa quantidade de sucessos e desvios padrão de erro baixos. O sucesso do programa é obtido quando o robô encontra a caixa vermelha e o sucesso do sistema todo se dá quando o agente 2 chega ao mesmo destino após as devidas rotinas.

Neste trabalho, alguns resultados preliminares do sistema obtidos para múltiplos agentes se mostraram promissores. Os testes demonstraram que o sistema mostrou capacidade de coordenar das ações entre agentes e uma velocidade de convergência bem maior que a versão original, desenvolvida apenas para o uso de um único agente.

7.Considerações Finais

7.1.Quanto ao Software

- Utilizando o sistema operacional *Windows Seven o dispositivo bluetooth utilizado não possuía múltiplos canais, isto dificultou o andamento do projeto, visto que não se conseguia utilizar mais de um agente ao mesmo tempo. O problema foi resolvido usando outro dispositivo bluetooth de uma marca distinta.*

- O programa *MATLAB não realiza comandos concorrentemente, logo os comandos são executados de forma sequencial, ou seja existia um tempo entre a execução da rotina de busca de um agente para outro causando erros de precisão em relação ao tempo que o objetivo era encontrado*.

7.2.Quanto ao Hardware

- Os hodômetro dos motores não eram muito confiáfeis ou o tempo de resposta das funções de leitura das posições era demorado, uma vez que muitas vezes o agente girava mais do que era o expecificado.

- O sensor ultra-sônico se utiliza de ondas sonoras que viajam pelo ambiente, refletem em objetos que esteja em seu caminho e volta para o sensor. Este calcula o tempo decorrido entre o disparo e o retorno da onda para calcular a distância até o objeto. De acordo com o fabricante: “Objetos grande e com superfícies dura retornam as melhores leituras. Objetos macios, curvos (como uma bola) ou muito finos ou pequenos podem ser de difícil detecção pelo sensor.” e “Dois ou mais sensores ultra-sônicos operando no mesmo ambiente podem comprometer as leituras uns dos outros.” A primeira consideração é comprovada, sendo importante a utilização de obstáculos “corretos” para o perfeito funcionamento do robô. A a segunda tem seu efeito observado as vezes quando o agente sem estar perto de nenhum objeto gira como se tivesse encontrado algo pela frente.

- A arquitetura ideal de sensores ultra-sônicos seria atingida com três destes. O sensor extra estaria voltado para a frente do robô, eliminando o ponto cego ali localizado. Este ideal não fora utilizado para possibilitar a

utilização de um sensor de cor e outro de som, essenciais para identificarem o objetivo, uma vez que o NXT possui apenas quatro entradas de sensores.

- Todas as percepções do robô ocorrem pela leituras dos sensores do próprio robô, ou seja, ele não possui conhecimento prévio da posição de nenhum objeto no ambiente. Dessa forma, o robô precisa necessariamente “ver” o seu objetivo cor vermelha diretamente. Por isso a nem sempre o robô reconhecia o objetivo, pois estava em alguma posição que o sensor de cor não estava próximo suficiente da caixa.

8.Bibliografia

8.1.Referências Bibliográficas

[1] - G. WEISS. “Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence”, edited by Gerhard Weiss, 1999.

[2] - WOOLDRIDGE, Michael (2002) An Introduction to Multiagent Systems. West Sussex. John Wiley. 348p.

[3] - B. M. Good. “Evolving multi-agent systems: Comparing existing approaches and suggesting new directions”. Masters’s thesis, University of Sussex, 2000.

[4] - C. F. Touzet, “Robot awareness in cooperative mobile robot learning,” Auton.Robots, vol. 8, no. 1, pp. 87–97, 2000.

[5] - E. Yang, D. Gu, “Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey”, CSM-404, Technical Reports of the Department of Computer Science, University of Essex, 2004.

http://pt.wikipedia.org/wiki/Sistema_multiagente visitado em 31 de Julho de 2011.

http://www.inf.ufrgs.br/~alvares/INF01048IA/SistemasMultiagentes.pdf visitado em 31 de Julho de 2011.

8.2.Software

http://www.mindstorms.rwth-aachen.de/ visitado em 31 de Julho de 2011.

http://www.ni.com/academic/mindstorms/ visitado em 31de Julho de 2011.

http://www.dienxteebene.blogspot.com visitado em 31 de Julho de 2011.

http://www.mathworks.com/ visitado em 31 de Julho de 2011.