LICENCIATURA EM ENGENHARIA INFORMÁTICA

Robótica Evolutiva
- Propriedades Emergentes -

Alunos

Mário Vaz - 20061925

Sérgio Aires - 20061983

Professor Orientador

Prof. Manuel da Costa Leite

Relatório de Projecto Final de Curso apresentado no âmbito do curso de Licenciatura em Engenharia Informática.

Lisboa – Portugal

Outubro de 2010

"A robot may not injure a human being or, through inaction, allow a human being to come to harm."

"A robot must obey orders given to it by human beings, except where such orders would conflict with the First Law."

"A robot must protect its own existence as long as such protection does not conflict with the First or Second Law."

Three Laws of Robotics; Isaac Asimov

"I can envision a future in which robotic devices will become a nearly ubiquitous part of our day-to-day lives. I believe that technologies such as distributed computing, voice and visual recognition, and wireless broadband connectivity will open the door to a new generation of autonomous devices that enable computers to perform tasks in the physical world on our behalf. We may be on the verge of a new era, when the PC will get up off the desktop and allow us to see, hear, touch and manipulate objects in places where we are not physically present."

em "A Robot in Every Home" por Bill Gates, 16 de dezembro de 2006
[http://www.scientificamerican.com/article.cfm?id=a-robot-in-every-home]

Agradecimentos

Queremos agradecer a todos aqueles que de alguma forma nos incentivaram ou pressionaram positivamente a fazer mais e melhor, em particular o nosso Coordenador Professor Manuel Costa Leite.

Índice

Índice de figuras

Índice de tabelas

Resumo

O projecto Robótica Evolutiva - Propriedades Emergentes – tem como principal objectivo o desenvolvimento de um sistema robótico que consiga, num ambiente controlado, deslocar-se entre dois pontos (origem-destino) que podem ser fixos ou aleatórios, através de uma rede de pontos e respectivas ligações entre eles, recorrendo ao uso de um algoritmo que lhe permita definir qual o melhor trajecto a efectuar.

Este relatório de projecto descreve:

- a composição e análise do kit robótico Lego® Mindstorms® NXT;
- a listagem de linguagens e plataformas de desenvolvimento de programação robótica disponíveis;
- a análise e comparação das principais plataformas de desenvolvimento de programação para o kit Lego® Mindstorms® NXT;
- a utilização do algoritmo de Dijkstra;
- o desenvolvimento e procedimentos realizados.

O cenário imaginado é o de uma grande superfície laboral, por exemplo uma oficina de reparação de aviões, onde se encontrem varias zonas de trabalho espalhadas, mas identificadas e delimitadas.

Para que este Projecto alcançasse os seus objectivos, foi necessário planear, implementar, desenvolver e inovar!

Abstract

At the present time robotic systems are a ubiquitous reality. Robotic solutions are part of our everyday life in our own houses as domestic intelligent appliances like washing machines, smart vacuum cleaners or complex self-cleaning cooking machines. Although the unawareness of the daily user, devices so complex like these are considered automatons in a sense that they have actuators responding to sensors and are provided of artificial intelligence which made them able of decision making. Robotic systems are now an area of interest for education as they are seen not as a closed system already developed but as systems in continuous evolution and open to whoever wants to take them one step further. Learning kits have a critical role in education as they promote the freedom to develop algorithmic solutions for automatons.

This report presents a project that intends to implement a robotic solution capable of dislocating a robot between a starting point and a destination among a grid of points identified by colors, using the Dijkstra's algorithm. For this project it is used the Lego NXT robot commercial platform, programmable by using the well-known C language on the RobotC platform. The work presented in this report shows mainly the description and inventory of the Lego NXT robot, the platforms and languages available for this robot programing, the choice of the Dijkstra's algorithm, the choice of RobotC platform, the code developed and the procedures and experiences taken. The RobotC developing platform enables efficient programming and easiness of communication through Bluetooth technologies that allows easy deployment of the code developed to the Lego® Mindstorms® NXT kit.

1. Introdução

Propusemo-nos fazer o trabalho sobre Robótica – Propriedades Emergentes, por acharmos esta área altamente motivadora e interessante do ponto de vista pessoal e pela aplicabilidade social.

A robótica está já implementada em diversas áreas e tem ainda ampla margem de progressão, quer na robótica propriamente dita (agentes sem aprendizagem), quer na robótica autónoma com recurso à inteligência artificial.

Inicialmente algumas ideias nos surgiram no âmbito do tema escolhido para o Trabalho Final de Curso. Para a escolha deste trabalho, tivemos que ter em conta alguns aspectos:
- Seria desenvolvido num ambiente virtual ou físico?
- Conseguiríamos adquirir o equipamento para a demonstração que pretendíamos fazer?
- Como se comportaria o equipamentos em ambientes diversos?

Estas foram algumas dúvidas que nos surgiram. Mesmo sem saber a resposta de algumas decidimos avançar com a ideia da apresentação em cenário físico.

Contactámos algumas pessoas ligadas à robótica que nos elucidaram sobre esta área em Portugal e como funcionam os campeonatos escolares e universitários. Para cimentar ideias, assistimos a campeonatos e demonstrações de robótica, como o Robótica 2010 - Festival Nacional de Robótica e o TechDays 2010. Nestas visitas tentámos identificar os equipamentos utilizados nas competições, tanto em hardware como em software, sendo que desde logo percebemos que seria numa fase: criar ambientação ao equipamento e ao software, e realizar um estudo das diversas alternativas à programação do robot Kit Lego Mindstorms NXT.

Da análise que fazemos a estas visitas, apercebemo-nos que a programação de Robots caracteriza-se por um uso intensivo do hardware, por apresentar desafios ao nível dos algoritmos e por desafiar a capacidade sensorial, de raciocínio e interactividade. Soluções robóticas como o kit da Lego permitem minorar dificuldades generalizadas como a falta de componentes de software que sejam reutilizáveis, hardware não uniformizado e a realização de testes em ambientes reais com custos elevados.

2. Enquadramento teórico

A Robótica é uma área tecnológica, multidisciplinar do ponto de vista do conhecimento prático, que abraça áreas como a programação, o desenvolvimento de software, a matemática, a electrónica e a mecânica. É a ciência da engenharia e da tecnologia de robots e sua concepção, fabrico, disposição estrutural e aplicação funcional.

Por ser uma área multidisciplinar e por a sua concepção, desenvolvimento, montagem e programação implicar custos elevados tanto monetários como de tempo, o seu acesso torna-se mais restrito a algumas pessoas ou empresas. É neste aspecto que a educação tem um papel fundamental como dinamizadora do acesso mais acessível a qualquer pessoa interessada, sendo esta uma área de enorme potencial no ensino.

Os robots têm "per si" o poder de exercer um fascínio imediato em indivíduos de todas as idades pelas suas capacidades de aplicação prática e imediata. Por exemplo, quem ao observar o robot Asimo da Honda não se lembra de um vasto conjunto de situações em que este robot o poderia ajudar no seu dia-a-dia?

Figura 1 – Asimo, Robot da Honda
(http://deviceguru.com/explore-asimo-via-an-interactive-3d-model/)

Assim sendo, e como forma de ajudar a colmatar a distância entra a robótica de alto nível e os utilizadores iniciados com interesse pela aprendizagem, mas sem precisar de recorrer a alargados estudos multidisciplinares de electrónica, mecânica ou linguagens de programação, surgem kits como o Kit Lego® Mindstorms® NXT. Este tipo de kits

vem proporcionar o aumento do número de interessados pela robótica, o que por sua vez facilita o desenvolvimento académico e a aquisição mais económica de um destes kits.

2.1. O KIT LEGO®MINDSTORMS® NXT

Em Julho de 2006, a Lego® lançou o Kit Lego® Mindstorms® NXT, que é um kit de robótica completamente programável. Este Kit substituiu o Robotics Invention System que era a primeira geração deste produto. É actualmente comercializado na versão "Retail" (referência Lego® nº 8527) e na versão "Education Base Set" (referência Lego® nº 9797).

Figura 2 – Kit Lego® Mindstorms® NXT #9797

Figura 3 – Exemplo de Robot construído com o Kit Lego® Mindstorms® NXT #9797

O kit escolhido para este Trabalho Final de Curso foi a versão "Education Base Set". Não sendo o âmbito do TFC mostrar conhecimentos mais aprofundados de mecânica ou electrónica, torna-se óbvia a escolha deste tipo de kits, uma vez que são fáceis de trabalhar e possibilitam o acesso mais fácil e mais imediato ao desenvolvimento de programação e à aplicação prática de resolução de problemas e sua algoritmia. A construção de robots desta marca tira partido das características conhecidas das peças

Lego, enquanto a programação pode ser realizada em vários ambientes, sendo um destes vendido conjuntamente com o equipamento e possuí características de programação visual que o tornam mais acessível. Daí também o seu uso frequente em ambientes escolares como pudemos comprovar aquando da nossa visita ao Robótica 2010 - Festival Nacional de Robótica, onde nas várias provas a decorrer foi recorrente a utilização de kits Lego® por participantes iniciantes desde os 8 anos até participantes mais avançados em provas com um grau de dificuldade mais elevado.

Composição do kit Lego® nº 9797	Quantidade
Bloco "NXT Intelligent Brick" com bateria Li-Ion e carregador	1x
Motor de passo	3x
Sensor de toque	2x
Sensor de Luz	1x
Sensor de Som	1x
Sensor Ultra-sónico	1x
Cabos de conversão	3x (39 cm)
Cabos de ligação	1x (20 cm) 4x (35 cm) 2x (50 cm)
Peças várias da linha Lego® Technic®	431x
Caixa de transporte e manual com exemplo de construção	1x

Tabela 1 – Composição do Kit Lego® Mindstorms® NXT #9797

Descrição da composição do Kit:

- O bloco "NXT Intelligent Brick" paralelepípedo de dimensões aproximadas de 7x11x5 cm funciona como "cérebro do" Kit. É o componente chave deste kit, uma vez que na prática se trata de um computador de dimensões bastante reduzidas que dispõe de entradas para 4 sensores e controla até 3 motores, tudo através de cabos formato RJ12 muito similares aos cabos telefónicos RJ11, mas incompatíveis. Este bloco dispõe de um "display" LCD monocromático com uma resolução de 100x64 píxeis, de 4 botões para navegação nos menus hierárquicos e de uma coluna de som capaz de reproduzir amostras de som até aos 8 kHz. O processador é um Atmel® ARM7, com 64Kb de memória RAM e 256 Kb de memória flash. Para entrada de dados dispõe de interface Bluetooth ou por USB com ligação por cabo tipo A-B.

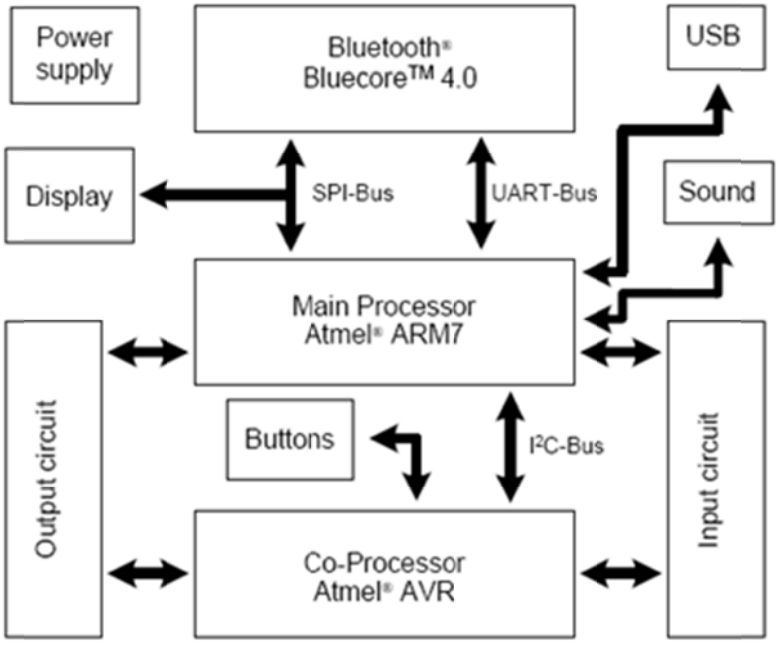


Figura 4 – Desenho da arquitectura do "NXT Intelligent Brick" (http://sine.ni.com/cs/app/doc/p/id/cs-12501)

No kit escolhido ("Education Base Set") a energia é fornecida por uma bateria Li-Ion recarregável e respectivo transformador/carregador com entrada directa equipamento, não sendo necessária a remoção da bateria para carregar. Na versão "Retail" a energia é fornecida por 6 pilhas formato AA de 1,5V, embora o "upgrade" seja sempre possível.

- O motor de passo, com sistema embutido de rotação através de um codificador rotativo óptico, permite medir o grau de rotação do motor. Este sistema mede a distância e a velocidade, o que permite ao bloco NXT controlar o motor com uma precisão de 1 grau.
- O sensor de toque detecta se está a ser pressionado, se foi premido e se foi libertado. O valor '0' significa que o sensor não está a ser pressionado, quando apresenta um valor '1' está a ser pressionado.
- O sensor de luz detecta o nível de luz ambiente ou a luz reflectida e inclui um LED para iluminação de superfícies e facilitar a leitura. A escala de valores vai de '0' muito escuro a '100' muito claro.
- O sensor de som mede o nível de som numa escala de '0' a '100'sendo que '0' é o silêncio completo e '100' é um som muito alto.
- O sensor ultra-sónico mede a distancia do sensor à superfície para a qual está orientado e detecta movimento. A distância pode ser apresentada em centímetros ou

polegadas. A distância máxima de leitura é de 233 cm com uma precisão de 3 cm. Este sensor funciona através do envio e recolha de ondas de som que são reflectidas na superfície dos objectos à sua frente, medindo o lapso de tempo entre envio e recolha.

- Os cabos de conversão servem para retro compatibilidade de utilização dos sensores e motores da edição "Robotics Invention System"
- Os cabos de ligação servem para ligação aos motores e sensores actuais
- As peças da linha Lego® Technic® são o equivalente aos elementos mecânicos, eixos, engrenagens, cremalheiras, rodas, pneus, polias, diferenciais e braços de apoio numa qualquer outra construção robótica

Figura 5 – Bloco "NXT Intelligent Brick" ligado a motores e sensores

2.2. PLATAFORMAS DE PROGRAMAÇÃO

Há actualmente disponíveis várias plataformas de desenvolvimento para o Kit Lego® Mindstorms® NXT para os diversos níveis de utilizadores, desde os iniciantes com escassos conhecimentos de linguagens de programação e preferência por um ambiente gráfico mais apelativo, a utilizadores experientes que exploram um nível de programação de acesso mais directo ao Kit, até utilizadores que pretendem criar ambientes de simulação onde pode ser testado o comportamento do robot no ambiente onde fisicamente irá actuar. Nem todas as plataformas são gratuitas, mas no mínimo disponibilizam sempre um período de teste ao utilizador. Assim sendo e no âmbito do Trabalho Final de Curso, analisámos algumas das plataformas disponíveis para percepção de qual seria mais adequado ao desenvolvimento deste trabalho. Na tabela 2, em anexo, estão algumas das aplicações utilizadas e respectivas linguagens de desenvolvimento. De seguida descrevemos mais em pormenor três dessas plataformas, que consideramos serem as mais utilizadas.

2.2.1. ROBOT C

Como constatámos, aquando da nossa visita ao Festival de Robótica 2010 pelos participantes em concurso, a plataforma de programação RobotC é uma das mais difundidas entre utilizadores quer sejam iniciados ou avançados. Esta plataforma tem um período de utilização à experiência de 30 dias, pelo que após esse período há um valor a pagar para obtenção de uma licença. O RobotC pertence à Academia de Robótica da Universidade de Carnegie Mellon e foi desenvolvido pela equipa de Educação Robótica.
Como principais pontos fortes destacamos a programação em C standard, o fórum de suporte online, a prototipagem rápida com robots Lego®, o suporte a sensores de outras marcas que não a Lego®, o editor próprio, o depurador compreensivo em tempo real e um conjunto de tutoriais orientados para a programação do kit Lego®.
Esta plataforma é usada no ensino de Programação Avançada, Engenharia, Mecatrónica e Sistemas Embebidos e Inteligência Artificial.

2.2.1.1. A LINGUAGEM

A plataforma RobotC utiliza a linguagem C standard, há muito utilizada tanto em sistemas operativos como em desenvolvimento de aplicações. A linguagem está extremamente bem documentada e existem inúmeras publicações e em sítios internet dedicados, o que torna esta plataforma ainda mais acessível a programadores menos experientes, mas ainda assim tornando mais fácil o desenvolvimento de aplicações mais complexas e robustas a programadores avançados.

O Kit Lego® Mindstorms® NXT é suportado no RobotC através da inclusão de extensões ao C para comunicação e desenvolvimento de código. O RobotC tem, além das capacidades multi-tarefas, características como o suporte a funções trigonométricas e matemática de vírgula flutuante. O RobotC é uma plataforma de controlo remoto e por isso precisa estar sempre a comunicar com o NXT quer seja por cabo USB ou por Bluetooth para que o código seja executado.

2.2.1.2. A FIRMWARE

O RobotC tem a sua própria firmware desenvolvida para o NXT com o intuito de aproveitar ao máximo o desempenho do Kit, disponibilizando funcionalidades extra como, por exemplo, a leitura de sensores por tarefas em segundo plano, envio de informação para o ecrã LCD, multitarefas com prioridades, aviso do nível de energia e tempos de execução entre 10 a 100 vezes superiores a outras plataformas para o NXT, o que torna o RobotC uma das plataformas mais rápidas a executar código.
O RobotC é um software multi-plataforma ou cross-plataform, com suporte para os blocos NXT, RCX e VEX. Os programas escritos para um destes blocos são portáveis para os outros apenas com poucas alterações ao código, sendo que cada um dos blocos irá ter uma firmware própria e a respectiva tradução das instruções da máquina virtual (V.M.) para a linguagem específica do bloco. A máquina virtual permite esta portabilidade como uma camada interface que fornece a ligação entre o código escrito no idioma ROBOTC e o hardware que é executado. As instruções da V.M. serão interpretadas de acordo com a firmware e kit utilizado, minimizando erros inerentes à má gestão de memória, entre outros (estas instruções são chamadas de bytecode).

O RobotC utiliza um sistema de interpretação, ou seja, o código é gerado para uma máquina virtual em vez de serem geradas instruções nativas para o controlador do robot (VM estas instruções são chamadas de bytecode). As instruções da máquina virtual são enviadas para o controlador do robot e a execução do programa utiliza um interpretador.

A utilização de interpretadores é amplamente utilizada no desenvolvimento em indústria, por exemplo, a linguagem Java é um sistema interpretativo.
Uma máquina virtual interpretativa é uma solução robusta orientada a salvaguardar contra erros de programação por parte de quem está a desenvolver o código, uma vez que ao programar com instruções nativas, torna-se mais frequente corromper a memória ou a execução de um programa. Num ambiente virtual são verificadas as instruções para garantir que não geram acções inesperadas, como por exemplo, permitir aceder a endereços de memória inválidos.

2.2.1.3. O EDITOR

O editor do RobotC é um dos pontos fortes por ser um editor de código desenvolvido especificamente para a linguagem C. Trata-se de um editor de código desenvolvido a pensar na linguagem C e possui características como: indentação automática de acordo com a estrutura e sintaxe do código e "autocomplete", um editor de texto preditivo com código colorido para facilitar a identificação de variáveis e keywords no código, sugestões que surgem ao seleccionar uma keyword e é o único depurador interactivo disponível para o Kit Lego® Mindstorms® NXT.

Este editor tem ainda um compilador que permite o teste e o envio directo de código para o NXT, sem ser necessário quaisquer outros programas para o fazer. Estão disponíveis neste editor "templates" de código, que permitem programar mais facilmente, com melhor ambientação à sintaxe e às funções fornecidas por forma a minimizar erros decorrentes de desconhecimento das funções ou da sintaxe da plataforma ou da linguagem. Há no editor dois perfis de utilização: o modo básico para utilizadores iniciados e o modo avançado para programadores experientes.

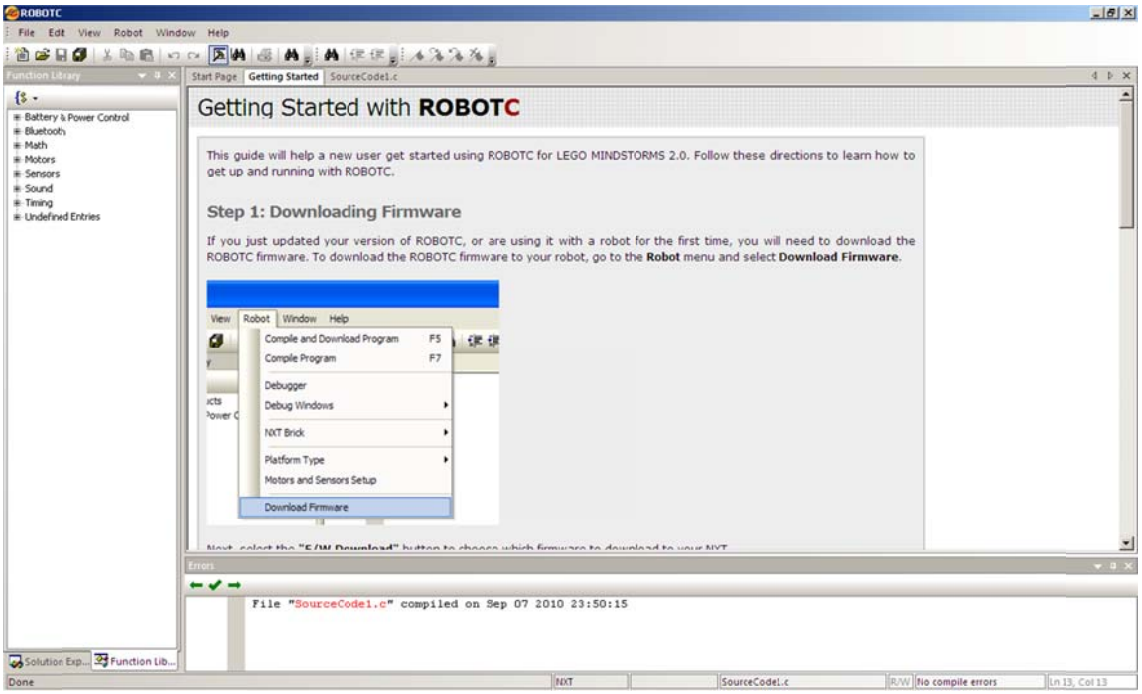


Figura 6 – Janela principal do editor da plataforma RobotC

2.2.2. LEGO® MINDSTORMS® NXT SOFTWARE

Outra plataforma disponível para a programação do bloco NXT é a Lego® Mindstorms® NXT Software, disponibilizada pela Lego®. Permite o acesso à robótica tanto a utilizadores iniciados (a partir dos 8 anos) como avançados, ainda que a um nível bastante simples de utilizar. Nesta plataforma é feita a programação dos robots NXT, sendo o upload do código para o bloco feito por comunicação USB ou Bluetooth. Esta plataforma da Lego é baseada na Labview da National Instruments e pretende ser de fácil e intuitiva utilização e programação com recurso a instruções práticas, guias de programação e objectos "drag-and-drop". De resto, o Labview era já utilizado desde 1986 por engenheiros e cientistas como ferramenta empresarial onde não é necessário conhecimentos avançados de programação. A programação por ícones tipo "drag-and-drop" torna a programação muito mais rápida e reduz o número de erros de sintaxe.

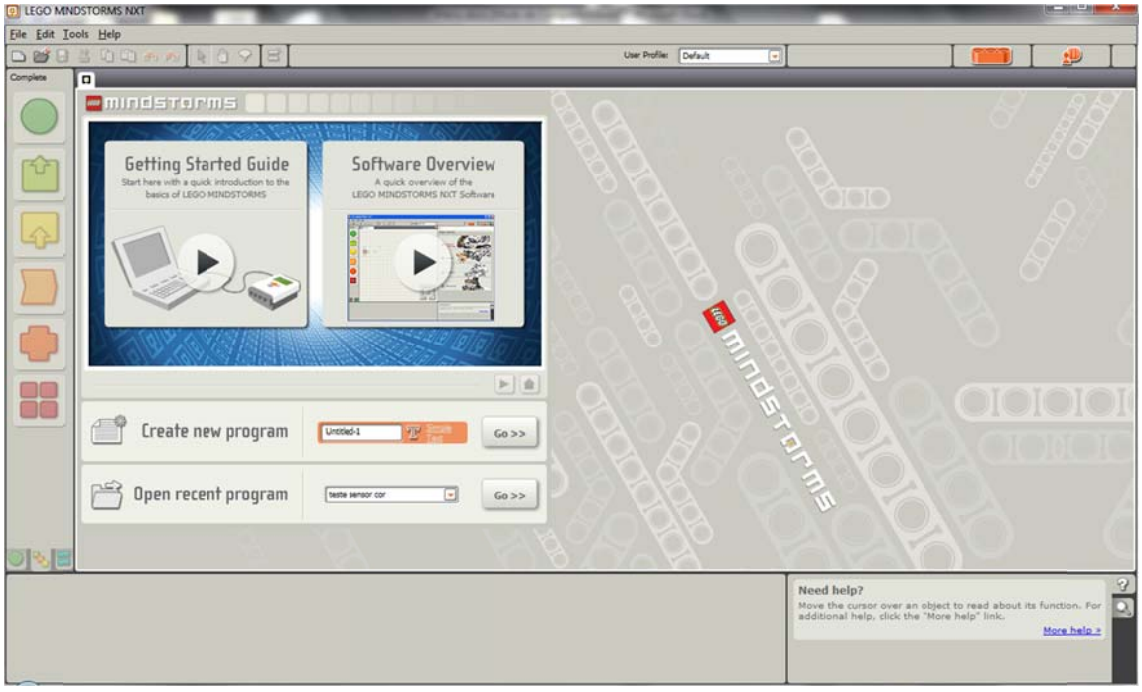


Figura 7 – Janela principal do editor da plataforma Lego® Mindstorms® NXT Software

2.2.2.1. A LINGUAGEM

A plataforma Lego® Mindstorms® NXT Software utiliza a linguagem NXT-G. Esta é uma linguagem de programação visual e, como tal, é um ambiente de programação visual que

contém elementos gráficos ou ícones como representação de métodos ou funções que podem ser manipulados pelos utilizadores de forma interactiva de acordo com gramática espacial especifica para construção de programas em vez da escrita mais exaustiva de linhas de código. Os programas construídos são depois carregados no bloco NXT e ficam disponíveis para teste imediato e interactivo com a plataforma ou podem executados mais tarde a partir da memória do bloco, navegando através dos menus do mesmo. Porém, esta plataforma apresenta limitações como o espaço reduzido de memória, o que leva a que seja necessário "perder" programas mais antigos para libertar espaço para programas mais complexos e consequentemente maiores consumidores de espaço. Assim, a Lego® Mindstorms® NXT Software é recomendada para construção de programas mais simples como utilização de funcionalidades básicas dos sensores, movimentos básicos de motor, entre outros. Para desenvolvimento de programas mais complexos é necessário recorrer a outro tipo de plataformas, não só pelo desenvolvimento de programas mais complexos, mas também pelas limitações quanto ao controlo de movimentos, às leituras de sensores e erros na aplicação de código mais complexo.

2.2.2.2. A FIRMWARE

A firmware do software Lego® Mindstorms® foi inicialmente de código fechado, mas é agora disponível a terceiros para edição de melhorias e inclusão ou modificação de funcionalidades. Existem vários Software Development Kit (Kit de Desenvolvimento de Software) que desenvolvimento de novas plataformas e para desenvolvimento de sensores para o Kit Lego® Mindstorms® NXT. A firmware é simples de utilizar e intuitiva. Permite fazer vários testes aos motores e sensores, com apresentação dos valores de medição dos sensores no ecrã.

2.2.2.3. O EDITOR

Para utilizar o editor integrado na plataforma Lego® Mindstorms® NXT Software não são necessários conhecimentos específicos de programação, basta arrastar para o ambiente de trabalho os ícones que representam os blocos de código e funções. Os blocos de código são ligados de forma automática aos já inseridos ou ao bloco inicial que já se encontra no ambiente de trabalho da plataforma aquando do início de um novo projecto.

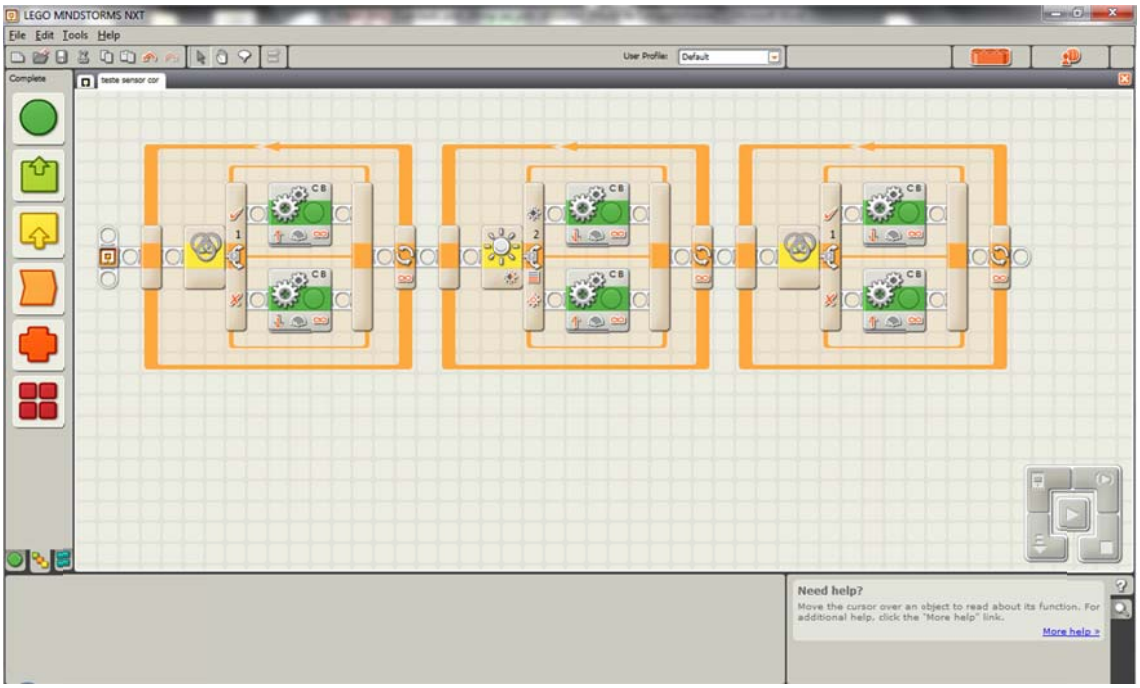


Figura 8 – Exemplo de programa desenvolvido na plataforma Lego® Mindstorms® NXT Software

Para os utilizadores iniciados está disponível um assistente de desenvolvimento de pequenos projectos robóticos, o Robo Center™, que através de desafios e manuais ajuda a alguns robots simples. Este assistente é útil num contexto de ensino e permite preparar a passagem para ambientes de desenvolvimento mais avançados onde podem ser criados robots mais complexos.

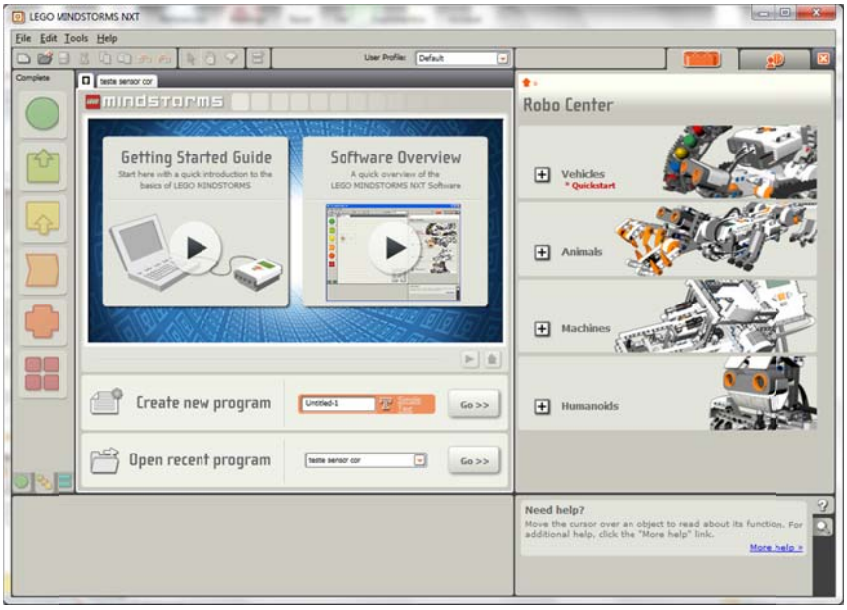


Figura 9 – Aspecto da plataforma Lego® Mindstorms® NXT Software com Robo Center à direita

2.2.3. Microsoft Robotics Developer Studio 2008 R3

O Microsoft Robotics Developer Studio (Microsoft RDS, MRDS) é a plataforma da Microsoft para desenvolvimento de projectos de robótica. Esta plataforma resulta da integração de vários outros produtos já existentes, tal como a plataforma .NET e o Microsoft Visual Studio, e de outras desenvolvidas de raiz especificamente para aplicações robóticas.
Ao nível mais baixo o Concurrency and Coordination Runtime é responsável pelo "scheduling" e I/O. Este software de baixo nível segue uma arquitectura de software Representational State Transfer (REST). A comunicação é feita através de um novo protocolo denominado Decentralized Software Services Protocol baseado em SOAP.

Tem como principais características: novo paradigma de desenvolvimento com base na criação de serviços que permite a reutilização de componentes; uniformização através de um paradigma de abstracção de hardware; Computação concorrencial e distribuída baseado no CCR - Concurrency and Coordination Runtime e DSS - Decentralized Software Services para simplificação de tarefas; ambiente de simulação em 3D totalmente editável e personalizável; apresentação de VPL - Visual Programming Language para facilitar o desenvolvimento de soluções robóticas.
O CCR - Concurrency and Coordination Runtime é um modelo de programação baseado na troca de mensagens altamente concorrenciais, que permite evitar a utilização tradicional de *Threads*, *Locks*, *Semaphores*. Destina-se a facilitar as necessidades existentes ao nível da programação concorrente, facilita o processamento de operações assíncronas e concorrenciais, além de explorar as capacidade do processamento paralelo.
O DSS - Decentralized Software Services é um identificador de contrato e de serviço e o canal de comunicação principal pelo qual são enviadas e recebidas mensagens directas entre serviços.
O VPL - Visual Programming Language é uma linguagem visual tipo *workflow e representa um b*om ponto de entrada para perceber como funciona o MRDS. Produz resultados rápidos sem ser necessário grande experiência com o MRDS e o código gerado é em linguagem C.

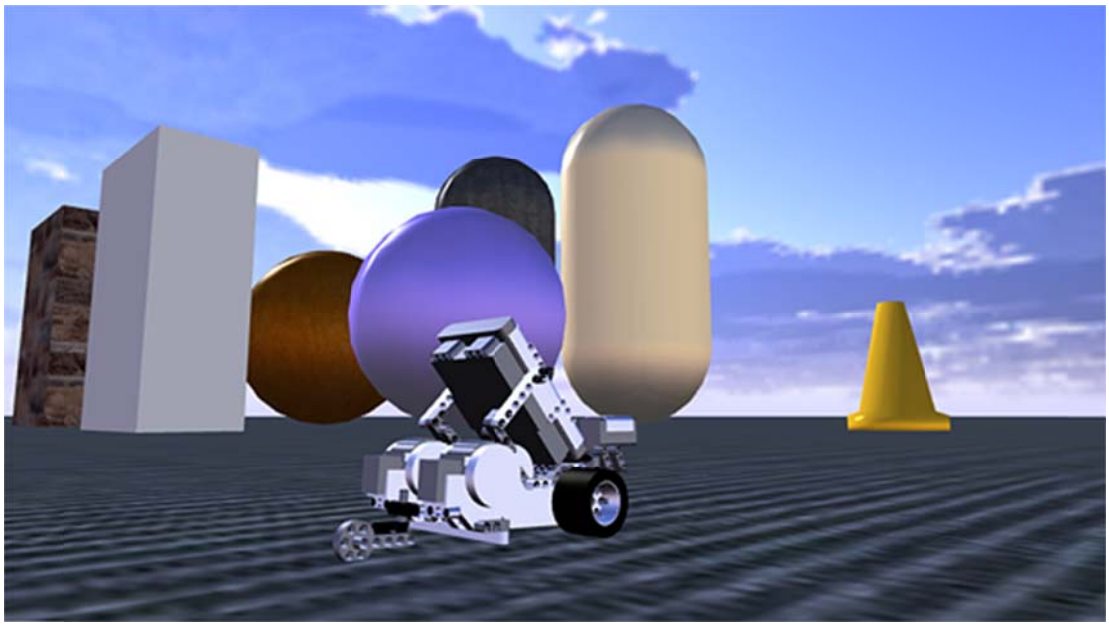


Figura 10 – Exemplo de ambiente de simulação desenvolvido na plataforma Microsoft Robotics Developer Studio

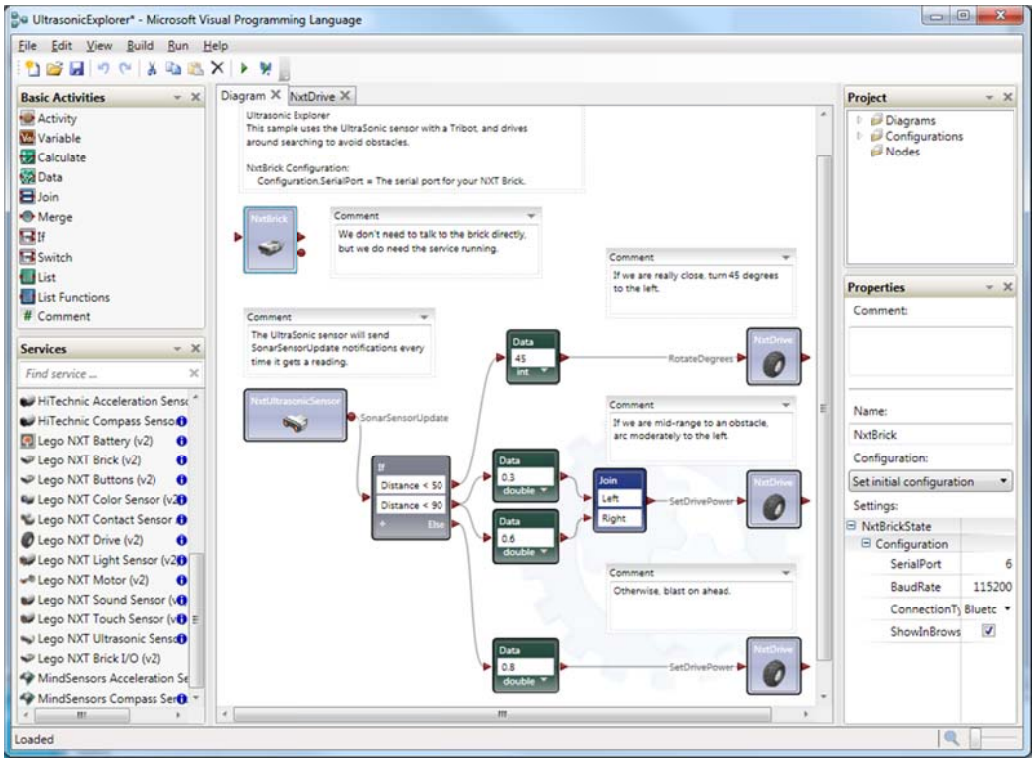


Figura 11 – Exemplo de programação desenvolvida na plataforma Microsoft Robotics Developer Studio

2.3. CONCLUSÕES SOBRE AS PLATAFORMAS

Após o estudo destas plataformas, apenas três das muitas disponíveis como podemos constatar no Anexo 2 - Lista de plataformas de programação para o Kit Lego® Mindstorms® NXT -, verificamos que constituem ferramentas para programação do Kit Lego® Mindstorms® NXT bastante robustas e ajustadas aos diversos níveis de programadores, desde os iniciantes sem conhecimentos específicos de linguagens de programação até aos programadores avançados capazes de desenvolver projectos mais complexos e exigentes. Para o bloco NXT, a plataforma Lego® Mindstorms® NXT Software é a mais adequada para utilizadores inexperientes (até porque é indicada para crianças a partir dos 8 anos), embora seja limitativa caso se pretenda implementar conceitos mais pertinentes do ponto de vista da Inteligência artificial, pois a sua limitada expressividade justifica-se por ser uma V.P.L., o que coloca entraves na implementação de conceitos mais avançados. Já a plataforma RobotC permite desenvolver projectos complexos de robótica e é por isso a recomendada para programadores mais exigentes e conhecedores de linguagens de programação, mas tem porém a obrigatoriedade de adquirir uma licença no caso de se pretender utilizar a plataforma por mais de 30 dias, o que para um utilizador ocasional não será limitativo, mas no caso de um estabelecimento de ensino, em que há a necessidade de comprar várias licenças, tornar-se-á dispendioso. A plataforma Microsoft Robotics Developer Studio permite, além de desenvolver programas complexos, testar o comportamento dos robots não só em ambiente virtual como em ambiente virtual e físico em simultâneo. Esta pode ser uma grande vantagem em relação às restantes plataformas, uma vez que permite o desenvolvimento de projectos de robótica mesmo que o programador não tenha disponível fisicamente um qualquer dos modelos de robots disponíveis no mercado. O desenvolvimento de programação puramente em ambientes virtuais tem no entanto alguns inconvenientes dos quais são exemplos de limitações:
-a falha ou desfasamento na leitura feita pelos sensores de luminosidade, o que conforme a iluminação do ambiente físico onde se realiza o desempenho do robot, pode resultar em comportamento aleatório ou errático por leituras de sensibilidade luminosa diferentes, conforme se trate de iluminação natural, de halogéneo, fluorescente ou outra;
-a ausência de atrito no ambiente virtual pode influenciar negativamente a deslocação do robot se o movimento não for acompanhado/verificado por sensores que detectem e corrijam a trajectória pretendida.

2.4. OPÇÕES DE AQUISIÇÃO

Assim, optámos por utilizar o Kit Lego® Mindstorms® NXT com a plataforma RobotC.

A escolha do Kit Lego® Mindstorms® NXT, deveu-se à grande publicação de vídeos que existem no Youtube com exemplos interessantes das capacidades desta consola, desde demonstrações de robots que seguem linhas (uma das publicações com mais vídeos) a tarefas com reacção a cores, sons, luz, etc. O factor económico também foi ponderado pelo facto do preço estar dentro do que achámos razoável despender.

Embora o kit base já inclua alguns sensores, depois de alguns testes achámos não serem os suficientes para este trabalho. Por exemplo, para seguir uma linha são necessários dois sensores de reacção à luz e o kit apenas inclui apenas um. Embora seja possível fazer o robot seguir uma linha apenas com um sensor, não é aconselhável, uma vez que o movimento do robot se torna bastante irregular e em situações de ângulos apertados a falha é frequente. Assim, além do kit NXT, adquirimos mais dois sensores de reacção à luz e ainda outro de reacção à cor. Este era o equipamento mínimo para a execução das tarefas pretendidas.

Para um melhor desempenho, os sensores de reacção à luz deveriam ser substituídos por sensores de reacção à cor, uma vez que praticamente não há variação de valores de input com as alterações de incidência de luz proveniente do ambiente.

Julgamos que seria útil a aquisição de um sensor de movimento, pois a haver interacção entre os dois robots, este sensor seria de grande utilidade. Para a detecção do efeito de presença do outro robot, utilizaríamos o sensor sonar.

Apesar do Kit NXT integrar um software próprio da lego, decidimos, com base nas visitas feitas, optar pelo RobotC. Adquirimos o software, também a um preço acessível. Além do RobotC, testamos o da Lego, o que nos pareceu, embora intuitivo, bastante limitativo. O Microsoft Robotics Studio, pareceu-nos uma boa plataforma de programação, visto incluir uma linguagem de programação visual e também interacção com o Visual Studio.NET, tirando partido das diversas linguagens disponíveis. Apesar das potencialidades do Microsoft Robotics Studio, este software é de certa forma complexo para ser explorado num curto espaço de tempo.

3. Método

3.1 – Requisitos da base de trabalho e ambiente

Devido à sensibilidade dos sensores, o ambiente de utilização do robot deverá ter uma incidência de luz conhecida, ou seja, devem ser efectuados vários testes de adaptação das variáveis aos valores de input dos sensores.

Deverá existir uma área de 160X180 cm, em papel branco, colocado numa base de modo a que fique totalmente plano para que não sejam visíveis sombras provocadas pelo efeito do relevo. A área em papel terá duas linhas paralelas a preto que definem o percurso que o robot irá fazer. As linhas deveram distar 1,5 cm entre si e ter uma espessura de 1 cm. O percurso terá zonas de referência indicativos de cruzamentos identificadas com círculos coloridos com 8 cm de diâmetro, preenchidos com as cores (azul, vermelho, cyan, amarelo, magenta e verde) e o seu centro coincide com o centro de cada cruzamento. Todos os segmentos de linha formam nos cruzamentos ou entroncamentos ângulos de 90º entre si, para que seja sempre proporcionada uma viragem de 90º ou múltiplos de 90º.

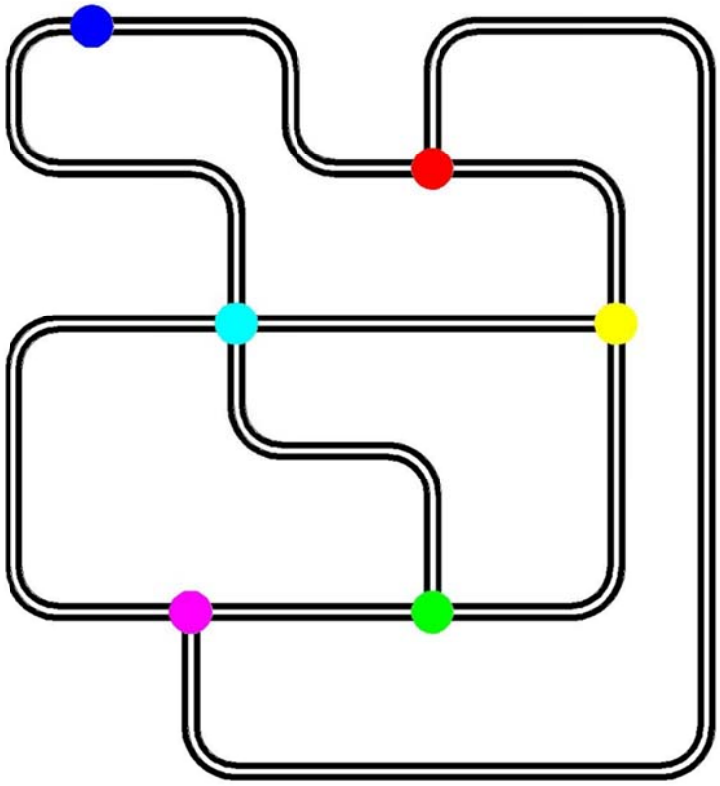


Figura 12 – Área de trabalho do projecto com representação de trajectos e nós de ligação

A consola e sensores estão assentes sobre duas rodas dianteiras com direcção fixa e uma roda traseira de tamanho inferior e com rotação a 360º. As rodas dianteiras têm tracção independente e distam 11,5 cm entre si. No topo de robot, está o sensor Sonar assente numa estrutura que lhe permite uma rotação até 180º.

Os sensores de reacção à luz estão colocados na dianteira do robot com uma posição vertical, paralelos entre si, apontados ao solo e justapostos lado a lado. A distância desde a superfície da zona de leitura no solo até ao receptor dos sensores é de cerca de 15mm.

Figura 13 – Robot NXT desenvolvido, onde são visíveis os sensores de luz á frente e o sensor de sonar no topo

Na apresentação será necessário um portátil com o software utilizado e com ligação permanente ao robot para envio do código desenvolvido. Pode ser necessário, conforme

as condições e tipo de iluminação da sala onde decorrer a apresentação, alterar parâmetros ou valores de variáveis devido à variação de intensidade da luz.

É necessário um cabo ou um dispositivo de Bluetooth para fazer a passagem dos dados.

Poderá haver necessidade de intervenção humana para alinhar o robot, principalmente nos pontos de partida. Durante o percurso não será necessário.

3.2 – Ideias Consideradas e Abandonadas

Como já referimos, a aplicabilidade da robótica é imensa e transversal a praticamente todas as áreas. Foram muitas as ideias que nos foram surgindo e que de alguma forma nos ajudaram a optar por uma. Enunciamos algumas:

- *Os ratos e as minas*

  Assistimos a uma reportagem sobre a desminagem de terrenos em Moçambique. São colocados no terreno ratos treinados para detecção de explosivos, esses ratos estão presos por um fio para seguirem uma linha definida. Ao detectar o cheiro a explosivo, o rato assume uma posição que indica aos especialistas em desactivação minas que foi encontrada uma mina. Este trabalho é moroso e requer muita atenção dos especialistas aos movimentos do rato. De modo a optimizar o processo, surgiu-nos a ideia de tentar associar a robótica a este trabalho de desminagem. Trocámos algumas ideias com um investigador que treina ratos para as suas investigações de Neurociências no Instituto Gulbenkian de Ciência em Oeiras, e colocamos-lhe a questão de qual a possibilidade de treinar um rato por forma a interagir com um robot. A ideia seria o robot seguir o rato, usando um sensor de movimento, ao detectar um cheiro, o rato pressionava um sensor para que este fizesse uma marca na zona do cheiro, simulando assim uma marca numa zona com mina. O investigador mostrou-se interessado em colaborar connosco e disse-nos que não seria difícil treinar o rato.

- *Busca e Salvamento*

  Um dos exercícios usado em campeonatos de robótica é o de Busca e Salvamento. Este exercício consiste em fazer o robot percorrer determinado percurso, até encontrar dois objectos, onde um desses objectos representa uma vitima e outra o atacante. A ideia é o robot resgatar o objecto com a cor identificada como vitima.

- *Segue Linha*

  Seguir uma linha é um dos primeiros exercícios que se tenta fazer ao adquirir um robot do tipo NXT. Nesta ideia, quisemos acrescentar algo para que se tornasse um exercício mais interessante. Acrescentaríamos a possibilidade de dois robots circularem no mesmo espaço mas que detectassem a presença um do outro, evitando deste modo o contacto. Testámos a detecção com o sensor Sonar, mas este mostrou-se pouco eficaz. Seria necessário a obtenção de novos sensores, nomeadamente um sensor de detecção e identificação posicional do movimento.

- *Robots no Hospital*

  No fundo, esta ideia foi a que despoletou a base do nosso trabalho.

  A ideia *Robots no Hospital* assentava no seguinte:

  Os hospitais principais são compostos por grandes áreas, muitas salas, corredores enormes, vários pisos, etc. Acontece por vezes, os enfermeiros terem que percorrer distâncias consideráveis para recolherem utensílios, medicamentos ou materiais para tratar de determinado doente acamado. Achamos que este tempo poderia ser eliminado com a utilização de uma via de circulação para robots.

Haveria uma sala onde os robots estavam em parque, cada robot continha integrado um ou vários Kits de assistência.

A assistente de saúde apenas teria que emitir um sinal digital, com dispositivo próprio ou mesmo telemóvel, por forma a chamar o robot com o kit necessário ao tratamento de determinado doente. O robot seguiria até ao destino pela via para robots.

Esta ideia pareceu-nos interessante porque pensamos que seria de possível aplicação, embora viéssemos a reconhecer que nos hospitais a componente humana tem grande peso, e talvez não fosse de bom-tom termos robots a percorrem hospitais.

Estas foram algumas das ideias que achamos que seriam exequíveis caso tivéssemos disponibilidade suficiente de equipamentos.

3.3 – Ideia escolhida

A ideia escolhida teve por base as experiencias que fizemos com os equipamentos, a conversa com o Professor Orientador depois de apresentadas as potencialidades dos robots e também o que achamos ser possível efectuar no espaço de tempo disponível e ainda que tivesse interesse quer académico quer para aplicação real.

O cenário imaginado é o de uma grande superfície laboral, por exemplo uma oficina de reparação de aviões, onde se encontrem varias zonas de trabalho espalhadas, mas identificadas e delimitadas. Para as suas tarefas, os trabalhadores de cada zona necessitam de diversas ferramentas e materiais, que normalmente também são usados por outros trabalhadores de zonas distintas.

Para evitar as perdas de tempo tanto na procura dos utensílios, tanto no próprio acto de os transportar, será implementado um sistema de distribuição de utensílios assistido por via robótica.

Haverá uma zona de parqueamento de robots, onde cada um tem um kit definido de utensílios, ferramentas ou materiais. O mecânico aeronáutico não precisa de saber em

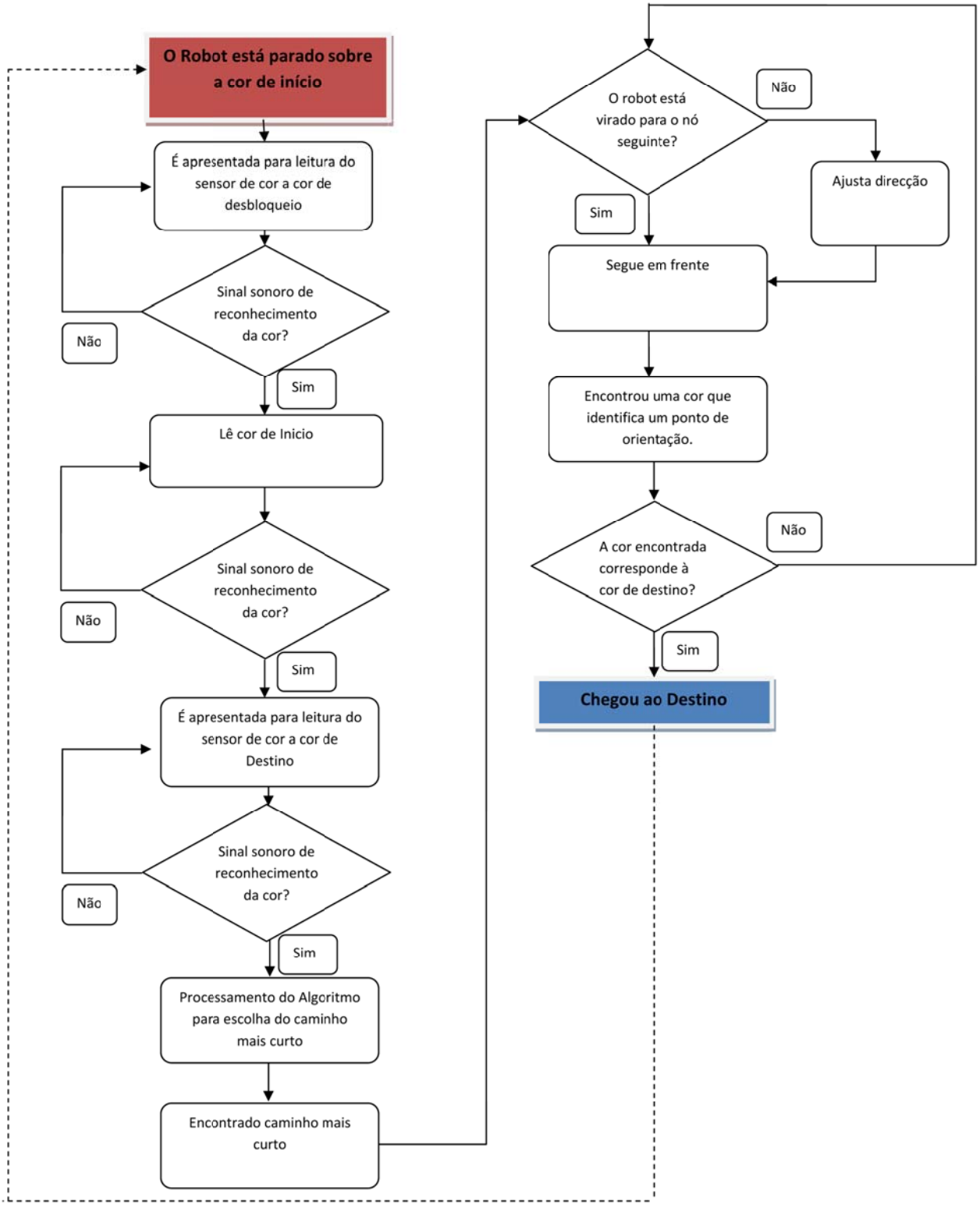
que zona determinado robot se encontra para lhe levar o material, basta para isso, marcar num dispositivo electrónico o número que identifica o robot com o equipamento pretendido.

Ao receber o sinal, o robot escolhe o caminho mais curto para lá chegar. Qualquer ponto de destino de um robot pode ser também um ponto de partida, não havendo necessidade de por cada viagem voltar ao parque. Os robots terão a capacidade de identificar a presença de outros robots na via e assim evitar a colisão.

Com base na interacção entre os robots, terão também alguma capacidade de *decisão* no percurso a escolher, ou seja, se num determinado percurso um dos robots avariar deverá emitir um sinal por forma a avisar os outros de que a via está impedida, permitindo assim que aquele caminho não seja considerado nas escolhas ainda que seja o mais curto.

A implementação deste sistema tem como finalidade reduzir os tempos de reparação dos aviões, bem como diminuição de custos de mão-de-obra.

3.4 – Fluxograma



4. Resultados

Desenvolvimento

Para a implementação da simulação do sistema na oficina, criámos um cenário numa área em papel com linhas a simular a via para robots e pontos que representam os cruzamentos. Os pontos nos cruzamentos são coloridos e são estas cores que dão informação do posicionamento do robot no espaço, bem com o caminho a seguir até ao próximo ponto.

Todo o mapa do percurso teve que ser incluído no código de processamento com as respectivas métricas entre pontos. Havia outra possibilidade que era a adaptação de um sensor Giroscópio que identifica o posicionamento do robot no espaço bem como pontos previamente indicados. Salientamos que com o Giroscópio não seriam necessárias linhas a indicar o percurso. Não adquirimos este tipo de sensor.

A cada ponto foram atribuídas identificações numéricas, sendo que cada percurso é definido pelo par (número do ponto de partida – número do ponto de chegada).

O software foi desenvolvido em duas linguagens distintas. Para uma melhor visualização da execução do código, usámos o VBA no Excel, ai conseguimos de uma forma fácil perceber os valores calculados nas variáveis de cada percurso. Paralelamente fomos avançando com o desenvolvimento em linguagem C adaptada ao RobotC.

Para escolha do caminho óptimo, usámos o algoritmo Dijkstra.

Depois de carregado o array com as métricas, é iniciado o código Dijkstra, que irá devolver o percurso mais curto entre o Ponto de Inicio e o Ponto de Chegada. Em cada ponto em que o robot se encontra, é processada a informação de quais os pontos adjacentes e que rotação será necessária dar aos motores de modo a ajustarem o posicionamento para seguirem a linha até ao próximo ponto. Foram eliminadas as

possibilidades de repetição de caminhos, quer no avanço, quer em voltar atrás no percurso.

No ponto de partida, e uma vez que o robot não tem inicialmente qualquer indicação do seu posicionamento direccional, optámos por o alinhar sempre em direcção ao ponto adjacente com o valor mais alto. Assim que é conhecida a direcção a seguir, é executado o código que irá alinhar o robot na direcção desse ponto.

Chegado ao ponto de destino, é possível iniciar novo percurso, assumindo para inicio o ponto onde se encontra, sendo que neste caso é necessário direccionar manualmente o robot para o ponto adjacente com valor mais baixo.

5. Conclusões e trabalho futuro

Foi com grande satisfação que nos dedicámos a este trabalho e fizemo-lo sempre com uma visão para além daquilo que realmente nos propusemos executar. A cada passo, o entusiasmo de explorar a robótica foi crescendo e chegado este momento, sabemos que não vamos parar de testar novas funcionalidades e tentar aprofundar os conhecimentos adquiridos. A opção da aquisição dos Robots teve por base precisamente o querer continuar nesta área para além do trabalho académico.

Grande parte do nosso tempo até agora foi dedicado a explorar a mecânica e o funcionamento dos robots no seu todo, embora tenhamos incluído algum código que simula alguma decisão, no fundo o robot faz a escolha do caminho mais curto. Sabemos que apenas abrimos uma janela daquilo que verdadeiramente é a Robótica. Pensamos evoluir este trabalho com recurso à inteligência artificial. Este é um caminho que pretendemos começar já a seguir porque ficámos convencidos que mesmo com este equipamento conseguiremos resultados bastante interessantes.

Iremos continuar em contacto com as pessoas que fomos falando ao longo deste trabalho. São pessoas ligadas à robótica em Portugal e que tem todo o interesse em divulgar esta área, algumas até mostrando disponibilidade para uma divulgação na Universidade Lusófona. Lembramo-nos por exemplo do presidente da Evoluir21, Sr Noah J. Revoy, que mostrou total abertura para nos conceder uma entrevista.

Referências e Bibliografia

WWW

http://pt.wikipedia.org/wiki/Rob%C3%B3tica_evolucion%C3%A1ria

http://www.ortop.org/NXT_Tutorial/html/essentials.html

http://en.wikipedia.org/wiki/Lego_NXT

http://en.wikipedia.org/wiki/Artificial_intelligence

http://en.wikipedia.org/wiki/Evolutionary_robotics

http://robotica2010.ipleiria.pt/robotica2010/

http://www.robotc.net/download/nxt/

http://www.hitechnic.com/

http://www.ni.com/academic/mindstorms

http://www.ni.com/academic/mindstorms/works.htm

http://en.wikipedia.org/wiki/Evolutionary_robotics

http://lis.epfl.ch/index.html?content=resources/documentation/EvolutionaryRobotics/index.php

http://mindstorms.lego.com

http://en.wikipedia.org/wiki/Lego_Mindstorms

http://pt.wikipedia.org/wiki/LEGO_Mindstorms

http://www.ortop.org/NXT_Tutorial/ Página com Tutoriais de programação NXT-G.

http://www.wired.com/geekdad/2007/11/the-best-progra/

http://wiki.mytimeworld.com/files/attachments/documents/ROBOTClego.pdf
Carnegie Mellon, Robotics Academy., "RobotCLego." [Online] 2007

http://blog.bodurov.com/Lego-NXT-Mindstorms-with-Microsoft-Robotics-Developer-Studio/

http://pesquompile.wikidot.com/microsoft-robotics-studio

Literárias

- RUSSEL, S.; NORVIG, P. Artificial Intelligence: A Modern Approach. New Jersey: Prentice Hall, 1995.

- Alexandre Pereira, Carlos Poupa (2004, Pereira). Como escrever uma Tese, monografia ou livro científico usando o Word, 3ª edição, Lisboa: Edições Sílabo

Anexos

Anexo 1 – **Código desenvolvido**

```
Sub pesquisa()
'Elaborado pelo grupo Mário Vaz e Sérgio Aires[ordem alfabetica]
'Ano lectivo 2009-2010 - Trabalho Final de Curso - Robotica-Propriedades Emergentes
'Universidade Lusofona
'Licenciatura em Engenharia Informática
'--------------------------------------------------------------------------------
'ESTE CODIGO SERVE APENAS PARA UMA VISUALIZAÇÃO DOS
'VALORES DAS VARIAVEIS E DO CAMINHO A SEGUIR PELO ROBOT
[F9].Select

    Dim intI As Integer, intJ As Integer
    Dim sngMulti(1 To 6, 1 To 6) As Integer 'array das metricas
    Dim percurso(1 To 20, 1 To 6) As Integer
    Dim percursoLinha(10) As Integer
    Dim percursoColuna(10) As Integer
    Dim PercursoFinal(1 To 2, 1 To 14) As Integer

    'vectores de posições dos pontos para no final serem dados como argumentos ao
array sngMulti(intI, intJ)
    Dim inicio As String 'ponto de partida
    inicio = "D" 'Identificação das cores por letras
    fim = "F"

   Dim i As Integer
   Dim j As Integer
```

```
 'CARREGA ARRAY
 For intI = 1 To 6
    For intJ = 1 To 6
       sngMulti(intI, intJ) = ActiveCell.Value
       ActiveCell.Offset(0, 1).Range("A1").Select
    Next
    ActiveCell.Offset(0, -6).Range("A1").Select 'anda 5 para a esq
    ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
 Next
'-----------------------------------------------------------------------------------------
inicio = "4"
fim = "1"
p = 1
Z = 0
linha = inicio
N = 1
coluna = 1 'coluna, variavel que anda na matriz de distancias
m = 1
N = 1
pp = 1
xt = 1
EncontradoInversoUltimaColuna = 0
valorAcAnterior = 0
Dim NovaColuna As Integer
menorValorAcumuladoProvisorio = 100 'este valor inicial deveria ser infinito
menorValorAcumulado = 100
```

```
  valorDeLinhaParaComparacao = 1 'VARIAVEL PARA AJUDAR A EVITAR O CIRCUITO
INVERSO

  valorDeColunaParaComparacao = 1 'VARIAVEL PARA AJUDAR A EVITAR O CIRCUITO
INVERSO

  Dim linhaReferencia As Integer

  Dim VerificaPercurso(1 To 8, 1 To 14) As Integer

  'POSICIONA NO PONTO DE PARTIDA NO ARRAY DAS DISTANCIAS


  [m2].Select

  While xt < 14 '14 POSIÇÕES SÃO A QUANTIDADE VALORES POSSIVEI EM CADA LINHA

     xt = xt + 1 'para contar os ciclos

     EncontradoInversoUltimaColuna = 0

     'verifica qual o ultimo valor na linha da matriz distancias------

     x = 6

  While sngMulti(linha, x) = 0

   x = x - 1

  Wend

  '---------------------------------------------------------------

   While sngMulti(linha, coluna) = 0

     coluna = coluna + 1

   Wend

 ab = 0

 'verifica caminho inverso

 For b = 1 To N - 1

     If (VerificaPercurso(1, b) = coluna And VerificaPercurso(2, b) = linha) Or
(VerificaPercurso(1, b) = linha And VerificaPercurso(2, b) = coluna) Then


        ab = ab + 1

        If coluna < 6 Then

           coluna = coluna + 1
```

```
        End If
        abc = 1
        While sngMulti(linha, coluna) = 0 And coluna < 7 And abc < 7

           If coluna < 6 Then 'para não rebentar
              coluna = coluna + 1
           End If
           abc = abc + 1
         Wend

          If coluna = 6 Then
             EncontradoInversoUltimaColuna = 1

          End If
       End If

Next
metricaValida = 0
If ab = b Then 'se entrar aqui significa que não encontrou metrica valida
    metricaValida = 1
End If
If EncontradoInversoUltimaColuna = 0 Then
     VerificaPercurso(m, N) = linha
     m = m + 1
     VerificaPercurso(m, N) = coluna
     m = m + 1
     VerificaPercurso(m, N) = valorAcAnterior 'valor acum anterior - a rever
     m = m + 1
```

```
VerificaPercurso(m, N) = sngMulti(linha, coluna) 'valor do percurso
m = m + 1
VerificaPercurso(m, N) = VerificaPercurso(m - 1, N) + VerificaPercurso(m - 2, N)
m = m + 1
VerificaPercurso(m, N) = linhaAnterior
m = m + 1
VerificaPercurso(m, N) = colunaAnterior
m = m + 1
'APRESENTAÇÃO NA FOLHA DE EXCEL
ActiveCell.Value = VerificaPercurso(1, N)
ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
ActiveCell.Value = VerificaPercurso(2, N)
ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
ActiveCell.Value = valorAcAnterior
ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
ActiveCell.Value = VerificaPercurso(4, N)
ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
ActiveCell.Value = VerificaPercurso(5, N)
ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
ActiveCell.Value = VerificaPercurso(6, N)
ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
ActiveCell.Value = VerificaPercurso(7, N)
ActiveCell.Offset(1, 0).Range("A1").Select 'anda 1 para baixo
ActiveCell.Value = VerificaPercurso(8, N)
ActiveCell.Offset(-7, 1).Range("A1").Select
 m = m + 1
'movimentos de linha e coluna na matriz VerificaPercurso
m = 1
```

```
      N = N + 1
    End If
    If metricaValida = 0 Then ' se a metricaValida for zero,.. quer dizer q
                        'não houve nenhum valor aceitavel (todos os caminhos eram
inversos ou
                        'ja tinham sido calculados. Assim, é necessario colocar a 1 na matriz
o valor
                        'de caminho ja escolhido
     For ff = 1 To N - 1
            If VerificaPercurso(8, ff) = 0 And VerificaPercurso(2, ff) = linha Then
               VerificaPercurso(8, ff) = 1
            End If
      Next
    End If
'-------------------------------------------------------------------------------------------
   NovaColuna = 0
   If coluna = x Or coluna = 6 Then  '

      'SE CHEGOU AO FIM DA COLUNA, TEM QUE SALTAR PARA O MENOR
      'VALOR(POSSIVEL) VERIFICADO ATÉ AGORA. ISTO É PARA ESCOLHER O NOVO
CAMINHO
      'vai andando dentro do array acabado de carregar, para escolher o menor valor na
linha dos acumulados
      pp = 1
      aa = 0
      cc = 0
      For t = 1 To N - 1
         'menorValorAcumuladoProvisorio = VerificaPercurso(4, t) 'VERIFICAÇÃO NA
LINHA DOS ACUMULADOS
```

```
        If VerificaPercurso(8, t) = 0 Then 'se é igual a zero, é porque o
                                           'nó ainda não foi escolhido com o menor valor


            If aa = 0 Then 'para escolher o primeiro valor disponivel (so acontece uma vez no ciclo for)
                valorescolhido = VerificaPercurso(5, t)
                linha = VerificaPercurso(2, t)
                valorAcAnterior = VerificaPercurso(5, t)
               aa = 1 'com a passagem da variagem a 1, garante que so acontece uma vez a passagem neste if
                NColunainicial = t 'VERIFICAR SE ISTO PODE ESTAR AQUI OU SE TEM QUE PASSAR PARA FORA
                NovaColuna = 1
                d = 1
                cc = 1
            End If
          '-------------------------------------
        'se é o menor valor encontrado, então a variavel linha toma o valor q antes estava na coluna
            'If VerificaPercurso(1, t + 1) > 0 And VerificaPercurso(5, t + 1) <= VerificaPercurso(5, t) And VerificaPercurso(8, t + 1) = 0 Then
            If VerificaPercurso(5, t) < valorescolhido Then
                linha = VerificaPercurso(2, t)
                valorAcAnterior = VerificaPercurso(5, t)
                valorescolhido = VerificaPercurso(5, t)
                NColuna = t
                NovaColuna = 1
                d = 1
                cc = cc + 1
                'para memorizar as coordenadas actuais
```

```
                linhaAnterior = VerificaPercurso(1, t)
                colunaAnterior = VerificaPercurso(2, t)
            End If
          End If

          If cc = 1 Then
              VerificaPercurso(8, NColunainicial) = 1
        End If
        If cc > 1 Then
            VerificaPercurso(8, NColunainicial) = 0
            VerificaPercurso(8, NColuna) = 1
        End If
        'serve para marcar o nó como ja escolhido
         'CONFIRMAR SE PODE ESTAR AQUI
         zz = zz + 1
     Next
End If
Z = 0
  '-------------------------------------------------------------------------------------
 If coluna < 6 Then
   coluna = coluna + 1

End If

If NovaColuna = 1 Then
    coluna = 1
End If
Wend
```

```
h = 1
hh = 1
'em busca do trajecto
While VerificaPercurso(2, h) <> fim
    h = h + 1
Wend
'h = h + 1
 menorValorFinalDoNo = VerificaPercurso(5, h)
 PercursoFinal(1, 1) = VerificaPercurso(1, h)
 PercursoFinal(2, 1) = VerificaPercurso(2, h)
  For h = h To 14

    If VerificaPercurso(5, h) < menorValorFinalDoNo And VerificaPercurso(2, h) = fim
Then
        menorValorFinalDoNo = VerificaPercurso(5, h)

        'CRIAR ARRAY COM VALORES FINAIS
        pp = h

    End If

Next

'na saida do for, temos o valor da coluna onde
'se encontra o valor mais baixo, no valor h
PercursoFinal(1, hh) = VerificaPercurso(1, pp)
PercursoFinal(2, hh) = VerificaPercurso(2, pp)
hh = hh + 1
PercursoFinal(1, hh) = VerificaPercurso(6, pp)
```

```
PercursoFinal(2, hh) = VerificaPercurso(7, pp)

  pp = pp - 1

While pp > 0


      If PercursoFinal(1, hh) = VerificaPercurso(1, pp) And PercursoFinal(2, hh) =
      VerificaPercurso(2, pp) And (VerificaPercurso(6, pp) <> 0) Then

      hh = hh + 1

      PercursoFinal(1, hh) = VerificaPercurso(6, pp)

      PercursoFinal(2, hh) = VerificaPercurso(7, pp)

   End If


   kk = kk - 1

   pp = pp - 1

Wend

 End Sub
```

Anexo 2 - **Lista de plataformas de programação para o Kit Lego® Mindstorms® NXT [http://en.wikipedia.org/wiki/Lego_Mindstorms]**

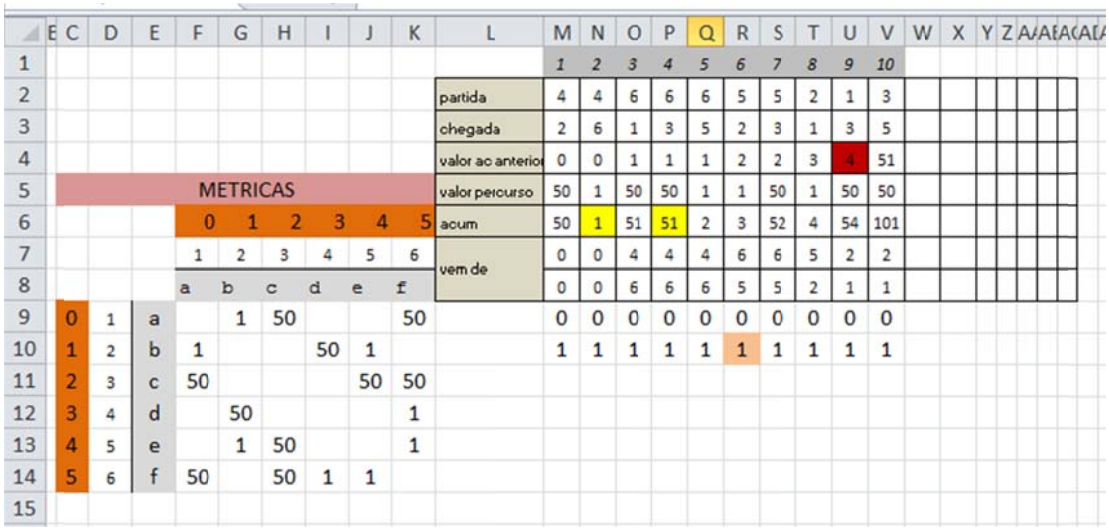
Name	Language type(s)	Notes
Actor-Lab	Custom flowchart-like language	
Ada	Ada	Requires nxtOSEK
Ada Interface to MindStorms	Ada	
brickOS	C/C++	
Ch	C/C++ Interpreter	Control Lego Mindstorm in C/C++ interactively without compilation
FLL NXT Navigation	Uses NXT-G and .txt files	
GCC	C/C++, Objective-C, Fortran, Java, Ada, others	
GNU Toolchain for h8300	C/C++, ASM	
jaraco.nxt	Python	Python modules providing low-level interfaces for controlling a Lego NXT brick via Bluetooth. Also includes code for controlling motors with an Xbox 360 controller using pyglet.
LabVIEW	National Instruments LabVIEW visual programming language (G code)	Core language used to develop Mindstorms NXT software. Can use available add-on kit to create and download programs to NXT, create original NXT blocks or control robot directly via USB or Bluetooth using NXT fantom.dll
Lego.NET	Anything that can compile to .NET, works best with C#	Does not come with a compiler, converts bytecode to machine code
Lego::NXT	Perl	Set of Perl modules providing real-time low-level control of a Lego NXT brick over Bluetooth.
LegoNXTRemote	Objective-C	Remote control program for remotely operating and programming a Lego NXT Brick. Supports NXT 2.0 and 1.0, sensors, all 3 motors, automatic "steering" control, and running preloaded programs.
leJOS	Java	A java based system for advanced programmers can handle most sensors and things like GPS, speech recognition and mapping technology. Can be interfaced with the Eclipse IDE or run from the command line
NXTGCC	Assembly, C, makefiles, Eclipse, etc.	The first GCC toolchain for programming the Lego Mindstorms NXT firmware.
nxtOSEK	C	
librcx	C/C++	A library for GCC
Logitech	Visual Basic, Visual	Can be combined with an RCX control library such

SDK	C++	as spirit.ocx from the MindStorms SDK to make use of the Lego Cam
MicroWorlds EX Robotics Edition		This is a program in the MicroWorlds series that allows students to control the NXT.
NQC	NQC, a C-like language	This is the most widely used unofficial language
NXT++	C++	Allows controlling the NXT directly from any C++ program, in Visual Studio, Windows.
NXT_Python	Python	NXT_Python is a package for controlling a LEGO NXT robot using the Python language. It can communicate via USB or Bluetooth.
NXT-Python	Python	NXT-Python, based on NXT_Python, includes additional advanced features and support for around 25 sensors. It has two major branches: a backwards-compatible version and one with a heavily-improved API. Works on all major OS's.
Lestat	C++	Allows you to control the NXT directly from any C++ program in Linux.
OCaml Mindstorm	Objective Caml	Module to control LEGO NXT robots using OCaml through the Bluetooth and USB interfaces.
Mindstorms SDK	Visual Basic, Visual C++, MindScript, LASM	You do not need VB to use the VB features as MS Office comes with a cut down version of VB for making macros
OnScreen	A custom language which can be programmed directly on the RCX	
pbForth	Forth	
PBrickDev	PBrickDev, a flowchart based language.	Has more functionality than the RIS language, such as datalogs and subroutines/multithreading.
PRO-BOT	A kind of Visual Basic/spirit.ocx-based language	Designed for robots which are in contact with the workstation at all times
QuiteC	C	A library for use with GCC and comes with GCC for Windows.
RCX Code	RCX Code, a custom flowchart-based language	Included in the Mindstorms consumer version sold at toystore
ROBOLAB	A flowchart language based on LabVIEW	This is the programming environment offered to schools who use MindStorms, supports the Lego Cam
RoboRealm	A multi-platform language that works with IRobot Roomba, NXT, RCX, VEX, and many other popular robotic sets.	

	This language is also capable for video processing using a webcam, this gives your robot excellent vision since it can filter out certain colors, lock-on to a certain area of color, display variables from the robot or computer, and much more. The software works with keyboard, joystick, and mouse. This software is freeware.	
Robotc	A multi-platform C language designed for users needing powerful debugging tools for the NXT, RCX, VEX, and soon-to-be FIRST Controller (for FRC).	ROBOTC gives the ability to use a text-based language based on the C language. It includes built-in debugger tools, as well as (but not limited to) code templates, Math/Trig operations (sin, cos,tan, asin,acos... etc.), user-friendly auto-complete function built into the interface, built-in sample programs
ROS	A Linux based library for writing robots. The stack "nxt" provides interface with the NXT.	
ruby-nxt	Ruby	Provides low-level access to the NXT via Bluetooth as well as some preliminary high-level functionality.
RWTH – Mindstorms NXT Toolbox	MATLAB	Interface to control the NXT from MATLAB via Bluetooth or USB (open-source).
SqLego	Squeak	
TclRCX	Tcl	
Terrapin Logo	LOGO	
TinySoar	Soar	An implementation of the Soar artificial intelligence architecture that runs on the RCX brick. Soar incorporates acting, planning, and learning in a rule-based framework.
TinyVM	Java	A predecessor to the lejos language. An open source Java based replacement firmware for the Lego Mindstorms RCX microcontroller.
The Transterpret	Occam	

er		
TuxMinds	(Linux) GUI for various distributions, an open source IDE based on Qt. Supports a lot of bots. RcX, NxT and Asuro are predefined.	With the XML-based configuration file almost any kind of bot (or microcontroller) can be added. Own equipment can be added in the same manner.
Gostai URBI for Lego Mindstorms NXT	URBI, C++, Java, Matlab	Easy to use parallel and event-driven script language with a component architecture and opensource interfaces to many programming languages. It also offers voice/speech recognition/synthesis, face recognition/detection, Simultaneous localization and mapping, etc.
Vision Command	RCX Code	The official programming language for use with the Lego Cam, that allows you to control your robot with color, motion, and flashes of light.
XS	Lisp	
LegoLog	Prolog	Uses an NQC program to interpret commands send from the PC running the Prolog code
Microsoft Visual Programming Language (VPL)	Graphical flowchart, based on .NET	With the Microsoft Robotics Studio, it uses a native NXT program msrs to send and receive messages to and from a controlling program on a computer via Bluetooth
DialogOS	Graphical Flowchart for voice controlled robots	DialogOS combines speech recognition and speech synthesis with robotics, enabling you to build talking robots that react to your voice commands.
Processing	Java (Simplified / programmed C-style)	Processing (programming language) is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. To control the NXT with Processing you can use the NXTComm Processing library developed by Jorge Cardoso.
Interactive C	C-style language.	Language developed for the MIT Lego Robot Design Contest
pbLua	API for the Lua programming language for the Mindstorms NXT, text-based	pBLua: ... is written in portable C, with minimal runtime requirements; can be compiled on the fly on NXT; is a small, easy to read, and easy to write language; has extensive documentation available online and in dead-tree format, and a very friendly newsgroup

Anexo 3 – **Resultado da programação em VB (ver ficheiro: exe do código VBA.xlsm)**



												1	2	3	4	5	6	7	8	9	10											
										partida		4	4	6	6	6	5	5	2	1	3											
										chegada		2	6	1	3	5	2	3	1	3	5											
										valor ac anterio		0	0	1	1	1	2	2	3	4	51											
				METRICAS						valor percurso		50	1	50	50	1	1	50	1	50	50											
				0	1	2	3	4	5	acum		50	1	51	51	2	3	52	4	54	101											
				1	2	3	4	5	6	vem de		0	0	4	4	4	6	6	5	2	2											
				a	b	c	d	e	f			0	0	6	6	6	5	5	2	1	1											
	0	1	a		1	50			50			0	0	0	0	0	0	0	0	0	0											
	1	2	b	1			50	1				1	1	1	1	1	1	1	1	1	1											
	2	3	c	50				50	50																							
	3	4	d		50				1																							
	4	5	e		1	50			1																							
	5	6	f	50		50	1	1																								

ANEXO 4 - Código desenvolvido para o RobotC

```
#pragma config(Sensor, S1,     HTCS2,               sensorLowSpeed)
#pragma config(Sensor, S4,     sonarSensor,         sensorSONAR)
#pragma config(Sensor, S3,     lightSensor2,         sensorLightActive)//esquerda do robot
#pragma config(Sensor, S2,     lightSensor3,         sensorLightActive)//direita do robo


#include "common.h"
#include "HTCS2-driver.h"

//------VARIAVEIS-------------------------------------------------------------------------
  int intI;
    int intJ;
    int sngMulti[7][7];
    int percurso[20][7];
    int percursoLinha[11];
    int percursoColuna[11];
    int PercursoFinal [4][15];//ALTEREI O PRIMEIRO PARAMETRO DE 1 PARA 3 - (CONFIRMAR)

    int inicio;// - PONTO DE PARTIDA
    int fim;// - PONTO DE CHEGADA



//-----------------------------------------------------------------------


task main()
{

   //
  sngMulti[0][0]=0;
  sngMulti[0][1]=0;
  sngMulti[0][2]=0;
  sngMulti[0][3]=0;
  sngMulti[0][4]=0;
  sngMulti[0][5]=0;
  sngMulti[0][6]=0;

  //PRIMEIRA LINHA
  sngMulti[1][0]=0;
  sngMulti[1][1]=0;
  sngMulti[1][2]=1;
  sngMulti[1][3]=50;
  sngMulti[1][4]=0;
  sngMulti[1][5]=0;
  sngMulti[1][6]=50;



  //SEGUNDA LINHA
  sngMulti[2][0]=0;
  sngMulti[2][1]=1;
  sngMulti[2][2]=0;
  sngMulti[2][3]=0;
  sngMulti[2][4]=50;
```

```
sngMulti[2][5]=1;
sngMulti[2][6]=0;

//TERCEIRA LINHA
sngMulti[3][0]=0;
sngMulti[3][1]=50;
sngMulti[3][2]=0;
sngMulti[3][3]=0;
sngMulti[3][4]=0;
sngMulti[3][5]=50;
sngMulti[3][6]=50;

//QUARTA LINHA
sngMulti[4][0]=0;
sngMulti[4][1]=0;
sngMulti[4][2]=50;
sngMulti[4][3]=0;
sngMulti[4][4]=0;
sngMulti[4][5]=0;
sngMulti[4][6]=1;

//QUINTA LINHA
sngMulti[5][0]=0;
sngMulti[5][1]=0;
sngMulti[5][2]=1;
sngMulti[5][3]=50;
sngMulti[5][4]=0;
sngMulti[5][5]=0;
sngMulti[5][6]=1;


 //SEXTA LINHA
sngMulti[6][0]=0;
sngMulti[6][1]=50;
sngMulti[6][2]=0;
sngMulti[6][3]=50;
sngMulti[6][4]=1;
sngMulti[6][5]=1;
sngMulti[6][6]=0;


while (HTCS2readColor(HTCS2)!= 7)
{
//CICLO PARA AGUARDAR ATE QUE SEJA MOSTRADA A COR PARA LEITURA
}


inicio = HTCS2readColor(HTCS2);//LE A COR DE INICIO

 nVolume = 4;
PlaySoundFile("heavy_taunts02.rso");

wait10Msec(500);//AGUARDA 5 SEGUNDOS


fim = HTCS2readColor(HTCS2);//LE A COR DE DESTINO
```

```
   nVolume = 4;
  PlaySoundFile("heavy_taunts02.rso");//SOM DE CONFIRMACAO


  int p = 1;
  int z = 0;
  int x;
  int linha = inicio;
  int coluna = fim;//CONFIRMAR SE INTEIRO OU STRING
  int m = 1;
  int n = 1;
  int pp = 1;
  int xt = 1;
  int b = 1;
  int EncontradoInversoUltimaColuna = 0;
  int ValorAcAnterior = 0; //valor acumulado anterior
  int NovaColuna;
  int menorValorAcumuladoProvisorio = 1000;//este valor inicial deveria ser infinito
  int menorValorAcumulado = 1000;
  int valorDeLinhaParaComparacao = 1; //variavel para evitar o circuito inverso
  int valorDeColunaParaComparacao = 1;
  int linhaReferencia;
  int VerificaPercurso[10][15];//ALTEREI O PRIMEIRO PARAMETRO DE 8 PARA 9  (CONFIRMAR)
  int abc;
  int metricaValida = 1;
  int linhaAnterior;
  int colunaAnterior;
  int ff;
  int valorescolhido;
  int NColunainicial;
  int d;
  int t;
  int Ncoluna;
  int zz;
  int kk;
  int menorValorFinalDoNo;
  int h;
  int hh;
  int teste; // TIRAR



  //INICIO DO CALCULO DAS
DISTANCIAS------------------------------------------------------------------------

  while (xt < 15) //15 posicoes sao a quantidade de valores possiveis em cada linha
  {
        xt= xt+1;//para contar os ciclos
        EncontradoInversoUltimaColuna = 0;
        //verifica qual o ultimo valor na linha da matriz distancias
        x=6;

  while (sngMulti[linha][x]==0)
      {
          x = x-1;

      }//FECHA WHILE
```

```
          while (sngMulti[linha][coluna]==0)
          {//ABRE WHILE

              coluna = coluna +1;

          }//FECHA WHILE


      int ab = 0;



//-------------------------------------------------------------------------------------------------
-------------------------
 //VERIFICA CAMINHO INVERSO

          for (b = 1; b<= n-1; b++)
          {
        if (VerificaPercurso[1][b] == coluna && VerificaPercurso [2][b]== linha ||
VerificaPercurso[1][b]==linha && VerificaPercurso[2][b]==coluna)
            {
                ab = ab + 1;

                    if (coluna < 6)
                    {
                        coluna = coluna + 1;
                    }
                    abc = 1;

                    while (sngMulti[linha][coluna] == 0 && coluna < 7 && abc < 7)
                    {
                        if (coluna < 6)
                        {
                          coluna = coluna +1;
                    }//FECHA IF

                    abc = abc +1;
                    }//FECHA WHILE


                    if (coluna == 6)
                    {
                        EncontradoInversoUltimaColuna = 1;
                    }//END IF

      }//END IF
            }//FECHA
FOR-------------------------------------------------------------------------------------------------
-------------


          metricaValida = 0;

          if (ab == b)
          {
              metricaValida = 1;
```

```
        }//END IF


        if (EncontradoInversoUltimaColuna == 0)
        {
                VerificaPercurso[m][n] = linha;
                m = m +1;
                VerificaPercurso[m][n] = coluna;
                m = m +1;
                VerificaPercurso[m][n] = ValorAcAnterior;
                m = m +1;
                VerificaPercurso[m][n] = sngMulti[linha][coluna];
                m = m +1;
                VerificaPercurso[m][n] = VerificaPercurso[m-1][n] + VerificaPercurso[m -2][
n];
                m = m +1;
                VerificaPercurso[m][n] = linhaAnterior;
                m = m +1;
                VerificaPercurso[m][n] = colunaAnterior;
                m = m +1;
                m = m +1;//confirmar se e' necessario
                m=1;
                n = n + 1;
        }//END IF



        if (metricaValida == 0 )//se a metricaValida for zero, quer dizer q n houve nenhum
valor aceitavel
                                //(todos os caminhos eram inversos ou ja tinham sido
calculados, Assim, e' necessario colocar a 1 na matriz o valor do caminho ja escolhido
        {
          ff = 1;
          for (ff = 1; ff<= n-1; ff++)
            {


              if (VerificaPercurso [8][ff] == 0 && VerificaPercurso[2][ff] == linha)//ESTA
A DAR ERRO DEVIDO AO 8 (EXCEDE A DIM DO ARRAY (CORRIGIR).....
                {

                  VerificaPercurso[8][ff] = 1;//IDEM NOTA ANTERIOR

                }//END IF

            }//END FOR

        }//END IF

  //ATE AQUI ESTA CERTO NA PRIMEIRA PASSAGEM


        NovaColuna = 0;



//-------------------------------------------------------------------------------------
```

```
-----------------
          if (coluna == x || coluna == 6)
          {

                  //se chegou ao fim da coluna, tem que saltar para o menor valor(possivel)
verificado ate agora. Isto e' para escolher
                  //o novo caminho
                  //vai andando dentro do array acabado de carregar, para escolher o menor
valor na linha dos acumulados


                  int pp = 1;
                  int aa = 0;
                  int cc = 0;
              t = 1;
              for (t = 1; t<=n-1; t++)
              {
                  if(VerificaPercurso[8][t]==0)//8 EXCEDE A DIM DO ARRAY (CORRIGIR).........
                  {

                      if( aa == 0)
                      {
                                valorescolhido = VerificaPercurso[5][t];
                                linha = VerificaPercurso[2][t];
                                ValorAcAnterior = VerificaPercurso[5][t];
                                aa =1; // com a passagem da variavel a 1, garante que so
acontece uma vez a passagem neste if
                                NColunainicial = t;
                                NovaColuna = 1;
                                d = 1;
                                cc = 1;

                      }//END IF


                      if (VerificaPercurso[5][t]< valorescolhido)
                      {

                                  linha = VerificaPercurso[2][t];
                                  ValorAcAnterior = VerificaPercurso[5][t];
                                  valorescolhido = VerificaPercurso[5][t];
                                  Ncoluna = t;
                                  NovaColuna = 1;
                                  d = 1;
                                  cc = cc + 1;
                                  //para memorizar as coordenadas actuais
                                  linhaAnterior = VerificaPercurso [1][t];
                                  colunaAnterior = VerificaPercurso[2][t];
                      }//END IF

                  }//END IF


                      if (cc == 1)
                      {
                          VerificaPercurso[8][NColunainicial]=1;
                      }//END IF
```

```
                    if (cc > 1)
                    {
                      VerificaPercurso[8][NColunainicial]=0;
                      VerificaPercurso[8][Ncoluna]=1;
                    }//END IF


                  //serve para marcar o no' ja escolhido
                  zz = zz +1;


              }//END FOR


          }//END IF

//-------------------------------------------------------------------------------------------
--------

          int Z = 0;



            if (coluna < 6)

            {
                coluna = coluna +1;
            }//END IF


            if (NovaColuna == 1)
            {
                coluna = 1;
            }//END IF


  }//END WHILE

  //ATE AQUI ESTA A FUNCIONAR PARA TODOS OS CICLOS



//-------------------fim da leitura dos percursos----------------------
//-------------------------------------------------------------------


//ESCOLHA DO MELHOR TRAJECTO-----------------------------------------

  h = 1;
  hh = 1;

  while (VerificaPercurso[2][h] != fim)
  {
      h = h +1;
```

```
  }//END WHILE
          menorValorFinalDoNo = VerificaPercurso[5][h];
          PercursoFinal[1][1]=VerificaPercurso[1][h];
          PercursoFinal[2][1]=VerificaPercurso[2][h];

  for (h = 1; h<=14; h++)
  {

      if (VerificaPercurso[5][h]< menorValorFinalDoNo && VerificaPercurso[2][h] == fim)
      {
          menorValorFinalDoNo = VerificaPercurso[5][h];

          pp= h;

      }//END IF

  }//END FOR
  //na saida do for, temos o valor da coluna onde se encontra o valor mais baixo (no valor h)


  PercursoFinal[1][hh] = VerificaPercurso[1][pp];
  PercursoFinal[2][hh] = VerificaPercurso[2][pp];
  hh = hh + 1;
  PercursoFinal[1][hh]= VerificaPercurso[6][pp];

  PercursoFinal[2][hh]= VerificaPercurso[7][pp];

  pp = pp -1;


  while (pp > 0)
    {

        if (PercursoFinal [1][hh] == VerificaPercurso[1][pp] && PercursoFinal[2][hh] ==
VerificaPercurso[2][pp] && VerificaPercurso[6][pp]!=0)
        {
            hh= hh+1;
            PercursoFinal[1][hh] = VerificaPercurso[6][pp];
            PercursoFinal[2][hh] = VerificaPercurso[7][pp];

        }//END IF

        kk = kk -1;
        pp = pp -1;

    }//FIM WHILE



//-------------------------------------------------------------------------------------------
-----------------------


switch (PercursoFinal[1][hh])
{
   case 1:
      if (PercursoFinal[2][hh] == 2)//segue em frente
```

```
        {
        motor[motorC]=20;
        motor[motorB]=20;
      }


      if (PercursoFinal[2][hh] == 3)//180 graus
        {
        motor[motorC]=20;
        motor[motorB]=-20;
      }


       if (PercursoFinal[2][hh] == 6)//90 graus esquerda
        {
        motor[motorC]=20;
        motor[motorB]=0;
      }


  break;



   case 2:
     if (PercursoFinal[2][hh] == 4)//segue em frente
        {
        motor[motorC]=20;
        motor[motorB]=20;
      }


      if (PercursoFinal[2][hh] == 5)//90 graus direita
        {
        motor[motorC]=0;
        motor[motorB]=20;
      }


       if (PercursoFinal[2][hh] == 1)//180 graus
        {
        motor[motorC]=20;
        motor[motorB]=-20;
      }


   break;


   case 3://QUANDO ESTA NO 3, O ROBOT ESTA VIRADO PARA O 5
     if (PercursoFinal[2][hh] == 5)//segue em frente
        {
        motor[motorC]=20;
        motor[motorB]=20;
      }
```

```
      if (PercursoFinal[2][hh] == 4)//90 graus direita
        {
        motor[motorC]=0;
        motor[motorB]=20;
      }


       if (PercursoFinal[2][hh] == 1)//180 graus
        {
        motor[motorC]=20;
        motor[motorB]=-20;
      }


  break;


 case 4:
    if (PercursoFinal[2][hh] == 6)//segue em frente
        {
        motor[motorC]=20;
        motor[motorB]=20;
      }


      if (PercursoFinal[2][hh] == 2)//90 graus esquerda
        {
        motor[motorC]=20;
        motor[motorB]=0;
      }



  case 5:
    if (PercursoFinal[2][hh] == 6)//segue em frente
        {
        motor[motorC]=20;
        motor[motorB]=20;
      }


      if (PercursoFinal[2][hh] == 2)//90 graus esquerda
        {
        motor[motorC]=20;
        motor[motorB]=0;
      }


       if (PercursoFinal[2][hh] == 3)//180 graus
        {
        motor[motorC]=20;
        motor[motorB]=-20;
      }


  break;
```

```
    case 6://NO CASO DO NO' 6, COMO NAO TEM SUPERIOR, O ROBOT ESTA VIRADO PARA O NO' MAIS
BAIXO, O 1
      if (PercursoFinal[2][hh] == 1)//segue em frente
          {
          motor[motorC]=20;
          motor[motorB]=20;
        }


        if (PercursoFinal[2][hh] == 3)//90 graus direita
          {
          motor[motorC]=0;
          motor[motorB]=20;
        }


        if (PercursoFinal[2][hh] == 4)//90 graus esquerda
          {
          motor[motorC]=20;
          motor[motorB]=0;
        }




         if (PercursoFinal[2][hh] == 5)//180 graus
          {
          motor[motorC]=20;
          motor[motorB]=-20;
        }


   break;

  }//FIM DO SELECT


   //NO CORRESPONDE AO NUMERO DE COR
   int NO_COR[7];
   NO_COR[1]=6;
   NO_COR[2]=9;
   NO_COR[3]=5;
   NO_COR[4]=2;
   NO_COR[5]=10;
   NO_COR[6]=3;

   //INICIO DO MOVIMENTO DO ROBOT
   hh= hh -1;//PORQUE O PRIMEIRO LANCO JA FOI EFECTUADO ACIMA

   while (hh>0)//PARA DAR INICIO AO MOVIMENTO DO ROBOT, INICIA A LEITURA DO FIM PARA O
INICIO DO ARRAY POR
              //TER SIDO CARREGADO DE FORMA INVERSA
   {


     //CODIGO A ACRECENTAR DEPOIS DE TESTES NAS SALA DE APRESENTACAO,
```

```
      //E NECESSARIO TESTAR OS SENSORES


    }
}//end task
```