UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Talita Timbó Vilela

MÉTODO HEURÍSTICO PARA TOMADA DE DECISÃO EM SISTEMAS DE ROTEIRIZAÇÃO DE MATERIAIS POR VEÍCULOS AUTOMATICAMENTE GUIADOS

Manaus
2015

Talita Timbó Vilela

MÉTODO HEURÍSTICO PARA TOMADA DE DECISÃO EM SISTEMAS DE ROTEIRIZAÇÃO DE MATERIAIS POR VEÍCULOS AUTOMATICAMENTE GUIADOS

Trabalho de Conclusão de Curso submetido à Coordenação do curso de Engenharia de Controle e Automação da Universidade do Estado do Amazonas como parte dos requisítos necessários para a obtenção do grau de Engenheiro.

Orientador Msc. Charles Luiz Silva de Melo

Manaus
2015

Talita Timbó Vilela

MÉTODO HEURÍSTICO PARA TOMADA DE DECISÃO EM SISTEMAS DE ROTEIRIZAÇÃO DE MATERIAIS POR VEÍCULOS AUTOMATICAMENTE GUIADOS

Trabalho de Conclusão de Curso submetido à Coordenação do curso de Engenharia de Controle e Automação da Universidade do Estado do Amazonas como parte dos requisítos necessários para a obtenção do grau de Engenheiro.

Aprovado em de de 2015.

BANCA EXAMINADORA

Msc. Charles Luiz Silva de Melo
Orientador

Professor
Convidado 1

Professor
Convidado 2

Professor
Convidado 3

Professor
Convidado 4

Manaus
2015

Agradecimentos

Agradeço primeiramente a Deus pelo dom precioso da vida e por ter abençoado minha jornada até aqui. Aos meus amados pais, Amaury e Walda, pela educação que me propiciaram, pelo incentivo contínuo, pela paciência e pelo apoio em minhas decisões.E aos meus irmãos Luís e Paulo pelo companheirismo, e a todos os meus familiares que me apoiaram. Aos amigos do Grupo de Robótica, GEAR, por todos os bons momentos, pela amizade, e por compartilharem comigo tanto de seus conhecimentos. Em especial aos amigos Cristiano Coimbra, Henrique Vieira, João Hidaka e Thales Araújo e professor Almir Kimura, que me ajudaram e incentivaram tanto durante o desenvolvimento do projeto. Aos amigos cativados durante o curso, por tantos momentos de descontração e pelo companheirismo e que nunca se indispuseram a compartilhar algum conhecimento. Especialmente a Lucas Bonetti, Dianny Machado, Renan Baima, Luiz Júnior, Antônio Medeiros, Eiji Maeda, Isabelle Stoco, Rafael Goulart, Izoneide Leite, Filipe Máximo, Nilteomar Gabay, Rodrito Santa Rita, Dilermando Ferreira. Não poderia deixar de agradecer aos amigos da Fundação Nokia, que acompanham minhas batalhas desde o ensino médio, em especial Adria Brito, Diego Freitas, Jerocílio Júnior, Adriano Encarnação, Simone Zagonel, Laion Nogueira, Sérgio William, Kaio Klinger e Thyago Ale. E ainda às amigas Karina Melo e Danyelle Gouveia. Agradeço ainda à Universidade do Estado do Amazonas, à coordenação de Engenharia de Controle e Automação e todos os seus professores, que tiveram fundamental importância neste projeto. Agradeço especialmente ao meu orientador, Msc. Charles Melo.

Resumo

O Veículo Guiado Automaticamente é um tipo de robô amplamente utilizado em ambientes fabris na movimentação de materiais. Neste trabalho será apresentada a implementação do método de busca A Estrela para encontrar a rota mais curta entre o ponto inicial e um ponto de destino, simulando a tomada de decisão pelo robô em sistema de roteirização. Uma interface para comunicação entre o usuário, que irá fornecer os pontos de entrada, e o robô será apresentada, assim como a comunicação sem fio entre os dispositivos, por meio da tecnologia ZigBee. O robô Pololu 3pi, que é um robô autônomo guiado por referência no chão, será utilizado no projeto como modelo do AGV. Por meio de simulações realizadas com o algoritmo desenvolvido e testes com o robô, observou-se o comportamento do modelo em várias situações, conferindo que o robô consegue chegar ao ponto de destino pelo caminho mais curto, uma vez que recebe os comandos certos e os executa no momento certo.

Palavras-chaves: AGV. Pololu 3pi. algoritmo A*. decisão. ZigBee.

Abstract

The Automatic Guided Vehicle is a type of robot widely used in factory environment for handling material. In this work, implementation of the search method A-Star will be presented to find the shortest path between the starting point and the goal, simulating the decision made by the robot in the routing system. An interface for communication between the user, which will provide entry points, and the robot will be presented, as well as a wireless communication between the devices through ZigBee technology. The Pololu 3pi robot, which is an autonomous robot guided by reference on the ground, will be used in the project as a prototype of the AGV. Through simulations with the developed algorithm and testing the robot, it was observed the model behavior in various situations, checking that the robot can reach the destination point by the shortest route, once it gets the right commands and runs it at the right time.

Key-words: AGV. Pololu 3pi. A* algorithm. finding. ZigBee.

Lista de ilustrações

Lista de tabelas

Sumário

1 INTRODUÇÃO

Nos últimos tempos o panorama industrial se configura de forma que as empresas devem atualizar-se às exigências do mercado de forma contínua. A produção desempenha um importante papel no auxílio ao cumprimento dos objetivos estratégicos da empresa e sua gestão da produção é conduzida de forma sistemática (MORAIS, 2010).

Os veículos automaticamente guiados (AGV – *Automated Guided Vehicle*) são robôs que seguem marcas ou fios no chão, e são utilizados para auxiliar a produção. Este tipo de robô é muito utilizado em ambientes fabris, na movimentação de materiais nas instalações da planta e em depósitos. Os AGVs começaram a ser mais amplamente utilizados no final do século XX(MORAIS, 2010).

Quando se trabalha com robôs móveis, e especialmente robôs autônomos, a localização e a navegação dos mesmos são as tarefas mais importantes, segundo Braunl (2006). Deve se saber onde o robô está, e um plano deve ser traçado a fim de se movimentar até o ponto de destino. Estes problemas estão interligados, pois se o robô não sabe sua exata posição no inicio da trajetória, ele encontrará problemas para chegar ao destino. Dentre os vários métodos de localização e navegação, mais adiante será mencionado o algoritmo A-Estrela (A*: *A-Star*).

Um problema clássico de roteirização é do caixeiro viajante (*Traveling Salesman Problem*), onde um caixeiro viajante procura encontrar o melhor percurso, o mais curto, a ser seguido a fim de se visitar várias cidades, dispostas aleatoriamente em um plano, porém cada cidade só pode ser visitada uma única vez. Não se sabe quando e quais os primeiros estudos realizados a fim de se solucionar esse problema, mas pode-se utilizar métodos heurísticos para se encontrar uma boa solução para este problema (REEVES, 1993).

Este trabalho trata do problema de roteirização de materiais por veículos automaticamente guiados em um ambiente industrial, será apresentada a utilização de um robô guiado por referencia no chão em um sistema complexo de roteirização e será mostrada a implementação da tomada de decisão por método de busca heurística, a fim de se percorrer a menor trajetória possível entre os pontos.

1.1 Problemática

A movimentação de materiais representa uma parte significativa dos custos operacionais, sendo assim, de grande importância em processos industriais (SULE, 2009). A roteirização de materiais pode ser realizada por meio do uso de veículos automaticamente guiados, porém seu baixo nível de tomada de decisão limita o uso desses veículos.

Assim sendo, este trabalho apresenta um método de busca que pode ser implementado para determinar o menor percurso a ser seguido entre dois pontos pelos veículos mencionados em ambientes fabris.

1.2 Justificativa

O uso de veículos automaticamente guiados contribui em processos industriais em vários pontos, podem ser citados o aumento da produtividade, a redução de mão de obra, a redução de danos aos produtos por manuseio, a redução de custos, o aumento na eficiência do produto acabado e a otimização do espaço fabril, além de ser possível a utilização dos veículos onde se fizer útil (LANGER, 2007).

Este trabalho apresenta uma proposta para otimizar os sistemas de roteirização de materiais por AGVs em ambientes fabris, fazendo com que cada veículo seja mais amplamente utilizado na movimentação de materiais, por meio da implementação de um sistema que encontre rotas curtas entre dois pontos.

1.3 Objetivo Geral

Construir, no laboratório da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, um modelo de um sistema de roteirização, utilizando o robô Pololu 3pi, uma pista, que será construída para simular o percurso, e uma Interface para comunicação do usuário com o robô, aplicando um método heurístico para tomada de decisão do melhor percurso a ser seguido pelo robô.

1.4 Objetivos Específicos

- Estudar os métodos heurísticos e escolher um método para solução de busca por menor percurso.
- Definir o ambiente, mapa, para ser utilizado no trabalho.
- Desenvolver o algoritmo de busca para encontrar a menor rota entre dois pontos.
- Criar uma interface de comunicação, por meio do *software* LabVIEW.
- Implementar a comunicação sem fio, utilizando a tecnologia ZigBee.
- Fazer a programação do robô Pololu 3pi, para executar os comandos enviados pelo computador.
- Desenvolver um ambiente de teste para validação do modelo.

1.5 Metodologia

Realizar um estudo de métodos heurísticos e suas aplicações em problemas de busca por menor percurso, por meio de pesquisa em referenciais bibliográficos.

Desenvolver um ambiente para servir como base para a simulação e teste do projeto, mapa, que possua vários caminhos possíveis entre os pontos existentes.

Criar uma interface de comunicação entre o usuário, o sistema e o robô. Para que o usuário possa fornecer as informações de entrada para o sistema, os pontos inicial e final.

Escrever um algoritmo para realizar a busca da melhor rota entre dois pontos do mapa, levando-se em consideração a menor distância a ser percorrida entre os pontos. O algoritmo deve ser baseado em métodos heurísticos para retornar um bom resultado.

Implementar a comunicação sem fio entre o computador e o robô, para que o robô possa receber os comandos de direção e seguir o percurso escolhido pelo algoritmo de busca.

Desenvolver o código de programação do robô Pololu 3pi, para que o mesmo execute os comandos recebidos.

Construir uma pista, em quadro de fundo branco e com caminhos representados por linha preta.

Simular em ambiente virtual, o algoritmo desenvolvido e testar em ambiente real a movimentação do veículo, utilizando o robô Pololu 3pi e a pista construída.

1.6 Organização do Trabalho

Este trabalho está dividido em cinco capítulos descritos da seguinte maneira.

No Capítulo 1 é exposta uma breve introdução sobre veículos automaticamente guiados em indústrias, e são apresentados a problemática do trabalho, justificativa, objetivos e um resumo da metodologia do trabalho.

O Capítulo 2 apresenta um referencial teórico das principais tecnologias utilizadas no desenvolvimento do projeto, como veículos automaticamente guiados, tecnologias de comunicação sem fio e o robô Pololu 3pi é apresentado, são apresentados também quatro métodos heurísticos.

A metodologia do projeto é descrita em detalhes no Capítulo 3, é feita a relação entre os materiais e métodos utilizados no desenvolvimento do projeto, descrevendo os principais passos da implementação do método heurístico para encontrar o melhor percurso entre dois pontos.

Em seguida, no Capítulo 4 são apresentados os resultados de simulações realizadas com o algoritmo de busca desenvolvido, assim como discussão e análise dos resultados obtidos.

Enfim, o Capítulo 5 são apresentadas as principais considerações finais.

2 REFERENCIAL TEÓRICO

De acordo com Sule (2009), a movimentação de materiais pode representar entre 30% e 75% dos custos operacionais totais em uma fábrica típica. Essa faixa mostra o potencial impacto da movimentação de materiais nos custos operacionais totais, mostrando quão importante essa atividade é, ainda que não agregue valor ao processo produtivo. O arranjo físico da fábrica é impactado pela forma de movimentação e manuseio, sendo necessárias ruas, corredores, espaço entre áreas de trabalho, entre outros. Durante o projeto de uma nova instalação o arranjo físico da fábrica e a movimentação de materiais são pontos que devem ser analisados em conjunto.

Um sistema de movimentação de materiais que não é capaz de realizar o fornecimento de peças e componentes nas linhas de produção, na quantidade correta e momento certo, prejudica o processo de produção, gerando descontinuidade do processo, elevação dos estoques em processo, ociosidade de equipes de produção e de máquinas e até mesmo perda de prazos de entrega (BOZER; YEN, 1996).

Em seu livro Baudin (2004), resume os meios de transporte utilizados para se alimentar linhas de produção, em especial em indústrias automobilísticas ou em indústrias de bens de consumo e produção em massa, citando paleteiras, carrinhos manuais, transportadores contínuos, empilhadeiras, composições dotadas de rebocadores e AGVs, conforme figuras 1, 2, 3 e 4.

Figura 1 – Paleteira

Fonte: http://goo.gl/56atXr

Figura 2 – Carrinho manual

Fonte: http://goo.gl/56atXr

Figura 3 – Empilhadeira

Fonte: http://goo.gl/C4JqQq

Figura 4 – Veículo Automaticamente Guiado

Fonte: http://goo.gl/1U2c9C

2.1 Veículos automaticamente guiados

De acordo com Sezen (2003), um AGV típico é constituído de chassi, baterias, sistema elétrico, unidade de acionamento, direção, unidade de parada de precisão, controlador a bordo, unidade de comunicação, sistema de segurança e plataforma de trabalho. Esses

veículos precisam de um sistema de orientação, que é o método por meio do qual seus percursos são definidos e os AGVs são controlados para seguir os percursos.

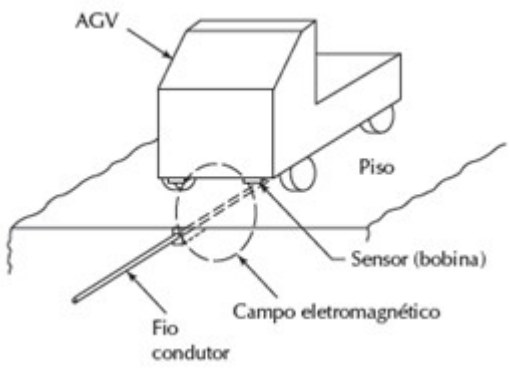
Segundo Field e Kasper (1989), os AGVs podem ser inflexíveis ou complexos. Sistemas inflexíveis de orientação impossibilitam a utilização generalizada deste tipo de robô, por outro lado, sistemas complexos os tornam caros e não confiáveis. Como resultado, o uso de AGVs tem sido limitado a aplicações especificas. O AGV mais inflexível é o guiado indutivamente por fio acoplado ao chão. Esses sistemas possuem um elevado custo de instalação e são de difícil adaptação, quando há necessidade de mudança na rota do robô. Outro método de orientação para o AGV é o guiado por marcas, no chão ou em paredes, em lugares específicos. Neste tipo de sistema o robô utiliza sensores para detectar as marcas e assim, o percurso a ser seguido. Groover (2011) cita três tecnologias utilizadas para orientação de veículos: condutores embutidos, faixas pintadas e veículos guiados automaticamente.

2.1.1 Condutores Embutidos

Neste método fios elétricos são colocados em um pequeno canal cortado na superfície do piso, o canal é então coberto por cimento, para que a superfície do piso seja contínua. Quando o fio é conectado a um gerador de frequência, um campo magnético é produzido ao longo do caminho. Os veículos possuem sensores que detectam este campo magnético e segue o caminho determinado, conforme ilustrado na figura 5.

Dois sensores, bobinas, são montados no veículo, um em cada lado do fio condutor. A intensidade do campo magnético medida por cada bobina é igual quando o veículo está posicionado de forma que o fio condutor esteja diretamente entre as duas bobinas. Se o veículo se afastar para qualquer um dos lados, ou se o percurso do fio condutor mudar de direção, a intensidade do campo magnético nos dois sensores não será igual. Essa diferença é usada para controlar o motor de direção, fazendo as mudanças necessárias na direção do veículo, a fim de igualar os dois sinais dos sensores, ou seja, para seguir o fio condutor.

Figura 5 – Veículo orientado por condutores embutidos



Fonte: Groover (2011)

2.1.2 Faixas Pintadas

Outro método de orientação é o de faixas pintadas para definir o percurso. O veículo possui sensores óticos capazes de detectar as faixas, que podem ser feitas com fitas, com tinta spray ou pintadas no chão. Esse método é útil em ambientes onde há muito ruído elétrico, porém as faixas devem ser mantidas limpas e substituídas periodicamente, pois se deteriora como tempo. Ainda assim, esse método é mais flexível que o método de condutores embutidos, pois novas faixas podem ser pintadas facilmente.

Figura 6 – AGV orientado por marcas no chão

Fonte: http://goo.gl/hTR9Pu

2.1.3 Veículos Guiados Automaticamente

Esse é o método que não necessita de percursos continuamente definidos no piso, pois ele utiliza uma combinação de orientação por cálculo e balizas localizados por toda a planta e que podem ser detectados e identificados por sensores do veículo. Baseado nas posições das balizas, o computador de navegação do veículo calcula as posições por triangularização. A maior vantagem do uso deste método é a sua flexibilidade, pois os percursos são definidos em software e a rede de percursos pode ser mudada simplesmente inserindo-se os dados no computador de navegação, sem grandes alterações na instalação da planta.

2.2 Métodos Heurísticos

Segundo Silva et al. (2013), as heurísticas são algoritmos que não garantem uma solução ótima, que seria a melhor solução para o problema, mas é capaz de retornar uma solução em tempo adequado para as necessidades da aplicação. Essa subjetividade, ou

falta de precisão dos métodos heurísticos, não se trata de uma deficiência, mas de uma particularidade que pode ser comparada à inteligência humana.

Entre os métodos mais utilizados, destacam-se os Algoritmos Genéticos, Busca Tabu, o Simulated Annealing e o Algoritmo A-Estrela.

2.2.1 Algoritmos Genéticos

Algoritmos genéticos é um método inspirado no processo Darwiniano de seleção natural dos seres vivos, onde os indivíduos mais aptos sobrevivem. Este método foi proposto por Holland (1975), e mostra a aplicação do processo evolucionário na busca de solução para vários problemas de otimização.

Segundo Prestes (2006), há um custo para cada indivíduo e o algoritmo transforma uma população criando uma nova geração de indivíduos, por meio de operadores de reprodução, cruzamento e mutação. Uma coleção de indivíduos representa o espaço de busca do problema a ser solucionado e o método propõe encontrar um indivíduo com o melhor material genético no espaço de busca (uma boa solução).

A melhoria das soluções é feita por meio do cruzamento entre as melhores soluções recentes e por suas mutações. O algoritmo não fica preso a um ótimo local, pois as operações de mutação fazem com que o algoritmo explore pontos diferentes no espaço de soluções.

2.2.2 Busca Tabu

De acordo com Benevides (2012), a Busca Tabu guia um algoritmo de busca local na exploração contínua dentro de um espaço de busca, evitando a convergência local em problemas de otimização. Este método cria uma lista de boas soluções que foram visitadas recentemente e só visita essa soluções novamente, após um determinado tempo. Ao se encontrar um ponto máximo ou mínimo, a busca segue para um ponto distante do atual, pois a permanência ou retorno para o ponto máximo ou mínimo não é permitida. Essa lógica faz com que a área do espaço de soluções visitadas seja maior.

A Busca Tabu utiliza estruturas flexíveis de memória para armazenar conhecimento sobre os espaços percorridos, afim de evitar que a busca retorne soluções previamente visitadas. Esta busca estabelece regras que administram uma heurística de refinamento na procura de soluções (FRAGA, 2006).

A Busca se inicializa a partir de uma solução inicial qualquer, procurando em toda a vizinhança da solução por seu melhor vizinho. Quando o melhor vizinho é encontrado ele passa a ser a solução corrente, mesmo que ele seja pior que a solução anterior. O processo é repetido até que algum critério de parada seja atingido.

2.2.3 Simulated Annealing

Segundo BUENO (2009), o método heurístico *simulated annealing* foi criado baseado no processo de tratamento térmico que visa alterar a estrutura cristalina de metais, a fim de lhes conferir características mecânicas e estruturais desejadas. Esse processo consiste em aquecer continuamente metais até certa temperatura e depois resfria-los de forma controlada, pois resfriamentos muito rápidos levam à imperfeições nos cristais metálicos e resfriamentos muito lentos ocasionam a formação de cristais muito grandes.

Neste método parte-se de uma solução viável de um problema e soluções vizinhas são aceitas, a princípio a probabilidade de qualquer solução vizinha ser aceita é alta, mas no decorrer do resfriamento a probabilidade de melhores soluções serem aceitas aumenta.

2.2.4 Método A*

De acordo com Cazenave (2006), *pathfinding* (busca de caminho) é algo muito importante em jogos comerciais e navegação de robôs. Em jogos é importante se utilizar um algoritmo de *pathfinding* otimizado pois os recursos do CPU também são necessários para outros algoritmos e porque muitos jogos são em tempo real.

A* é um método heurístico de busca admissível, usado para encontrar o menor caminho entre dois pontos. Conforme Luger (2013), um algoritmo de busca é dito admissível se ele conseguir encontrar um caminho mínimo até o objetivo, se essa solução existir.

O método de busca A* utiliza uma função heurística rápida que não sobrestima o comprimento do caminho até o ponto final, isto é, uma heurística otimista normalmente denominada h. Essa heurística, $h(n)$, é a estimativa do custo para se mover do nó atual, n, ao ponto de destino.

A heurística auxilia o cálculo da função de avaliação do método A*, f(n) , que é calculado conforme equação 2.1.

$$f(n) = g(n) + h(n) \tag{2.1}$$

Onde,

n é o nó estudado;

$g(n)$ é o custo para se mover do ponto de início ao nó em questão;

$h(n)$ é o custo estimado para se mover do nó até o destino final.

O custo real de movimentação entre o nó e o ponto final ainda não é conhecido, uma vez que podem haver bloqueios entre os pontos estudados, então é feita uma estimativa do custo para se mover entre os pontos, por isso $h(n)$ é frequentemente referida como heurística, uma vez que não é exata.

De acordo com Coppin (2012), se $h(n)$ for sempre uma superestima da distância de um nó a um nó objetivo, então o algoritmo A* será ótimo, ou seja, é garantido encontrar o caminho mais curto até um estado objetivo. A* é descrito como sendo otimamente eficiente, no sentido de que, para encontrar o caminho até o nó objetivo, ele expandirá o mínimo de caminhos possível. Mais uma vez, essa propriedade depende de $h(n)$ ser sempre uma subestimativa.

O custo de $h(n)$ pode ser calculado por meio da distância Manhattan ou distância Euclidiana, por exemplo. Distância Euclidiana é a distância reta entre dois pontos, o teorema de Pitágoras é normalmente utilizado para se calcular distâncias euclidianas, enquanto que a distância Manhattan é a soma das distâncias, horizontal e vertical, entre os pontos.

Abaixo segue a representações matemática dessas duas medidas.

Distância Euclideana:

$$\sqrt{\left((x_1 - x_2)^2 + (y_1 - y_2)^2\right)} \tag{2.2}$$

Distância Manhattan:

$$|x_1 - x_2| + |y_1 - y_2| \tag{2.3}$$

Em um mapa onde existam bloqueios entre os pontos inicial e final, sendo então necessário se contornar os obstáculos, tanto a distância euclidiana quanto a distância Manhattan serão menores que o verdadeiro custo de se movimentar entre o nó e o destino (ou no máximo de igual custo), significando que o cálculo das heurísticas por meio das equações 2.2 e 2.3 retorna uma heurística admissível.

Como $h(n)$ é uma estimativa do custo de movimentacao entre o nó n e o ponto de destino, a função de avaliação, $f(n)$, também é uma estimativa do custo total do caminho, partindo do ponto inicial, passando por n e chegando ao ponto de destino.

A lógica do método A* pode ser dividida em três partes, que serão apresentadas nas secções 2.2.4.1, 2.2.4.2 e 2.2.4.3 a seguir, e foi utilizada como base para o desenvolvimento do algoritmo de busca do projeto.

2.2.4.1 Iniciando a Busca

Primeiramente deve ser feita a simplificação da área de procura. Cada posição do mapa é analisado e seu estado é registrado como passável ou não passável. A procura do caminho é realizada levando em consideração somente as posições passáveis, as posições não passáveis, bloqueios, são ignoradas e não são consideradas. Aplica-se então uma logica para encontrar o caminho mais curto.

A procura é realizada a partir de um ponto de início. São analisados os nós adjacentes, expandindo cada vez mais a busca, até se chegar ao ponto de destino.

Os primeiros passos do método A Estrela, para buscar o menor caminho, são os seguintes:

- Passo 1:

Começa-se do ponto, nó, inicial, e a posição referente a este ponto é acrescentada a uma 'lista aberta', de posições a serem consideradas. Neste momento inicial só haverá um elemento na lista aberta, porém outros serão inseridos posteriormente. Esta lista contém elementos referentes às posições que poderão, ou não, fazer parte do caminho mais curto, encontrado por meio do método A*. Basicamente, a lista aberta é uma lista de nós que devem ser verificados e que fazem fronteira com o nó que está sendo analisado atualmente.

- Passo 2:

É feita a observação das posições alcançáveis, que são os nós vizinhos ao ponto que está sendo estudado atualmente. Somente as posições passáveis são observadas, bloqueios são ignorados. Os nós referentes às posições alcançáveis observadas são acrescentados à lista aberta. Para cada um dos nós observados, o nó atual deve ser salvo como seu nó pai ou os nós observados deve ser salvos como filhos do nó atual. O fato de relacionar o nó a seu ancestral é importante para que seja possível traçar o caminho posteriormente.

- Passo 3:

O nó atual deve então ser removido da lista aberta e acrescentado a uma 'lista fechada', onde são guardados os nós que não precisam ser procurados ou observados novamente.

Neste momento os nós passáveis próximos ao nó sendo analisado, têm como nó pai o nó atual, 'anterior'. O nó atual na lista fechada e todos os nós filhos estão na lista aberta, para serem conferidos.

O próximo passo é escolher um nó da lista aberta para ser analisado e se repetir quase o mesmo processo anterior (passos 1, 2 e 3). A escolha do nó a ser analisado em seguida leva em consideração a função de avaliação do método A*, equação 2.1.

O caminho é gerado passando-se repetidamente pela lista aberta, escolhendo nós com menores custos $f(n)$, processo que será explicado a seguir.

2.2.4.2 Continuando a Busca

Para a continuação da procura, o nó com menor custo $f(n)$ da lista aberta é escolhido para ser estudado.

Então são executados os seguintes passos para esse nó:

- Passo 4:

O nó é retirado da lista aberta e acrescentado à lista fechada.

- Passo 5:

Os nós vizinhos ao nó estudado atualmente são conferidos. Ignoram-se os nós que já estejam na lista fechada e posições não passáveis, bloqueios, os demais nós vizinhos ao atual são acrescentados à lista aberta, se ainda não estiverem nela. Então é feita a relação entre o nó atual e os nós vizinhos que acabaram de ser adicionados à lista aberta, os nós vizinhos são salvos como filhos do nó atual e o nó atual é salvo como pai destes nós vizinhos.

- Passo 6:

Se um nó vizinho já estiver na lista aberta, deve-se analisar se o novo caminho até o nó em questão é melhor que o caminho anterior. Pode-se fazer esta análise comparando os custos, $g(n)$, para se mover até o nó estudado pelos dois caminhos diferentes, o anterior e o novo. Se o $g(n)$ deste novo caminho não for menor, nenhuma ação é tomada.

- Passo 7:

Porém se o custo do novo caminho até o nó analisado, $g(n)$, for menor que o custo $g(n)$ anterior até este mesmo nó, é feita uma nova relação de parentesco entre os nós. Deve-se trocar o pai anterior do nó vizinho estudado pelo nó atual, agora este nó estudado é filho do nó atual. E então os custos $g(n)$ e $f(n)$ devem ser recalculados para o nó estudado.

Após a realização dos passos acima, provavelmente novos nós foram incluídos na lista aberta, e esses nós devem ser reordenados crescentemente, conforme a função de avaliação $f(n)$.

2.2.4.3 Finalizando da Busca

O nó com menor custo $f(n)$ na lista aberta é então estudado, ele é retirado da lista aberta e adicionado à lista fechada, passo 4. Se este nó coincidir com o ponto de destino, a busca é finalizada, pois o ponto final foi encontrado. Se não coincidir, então a busca continua, e dá-se prosseguimento à repetição dos passos 5, 6 e 7, nesta ordem, voltando à secção anterior 2.2.4.2.

Quando a busca é finalizada, por meio da lista fechada se tem acesso ao caminho encontrado pelo algoritmo, que deve corresponder ao menor caminho entre os pontos inicial e final. Pela relação de parentesco entre os nós da lista fechada, é possível organizar os nós na sequência correta em que devem ser seguidos para se percorrer o caminho do ponto inicial ao final.

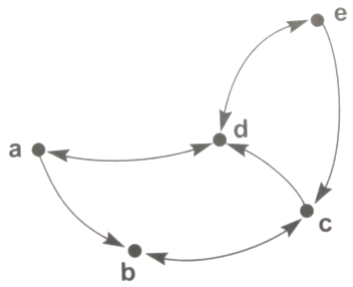
2.2.5 Representação por espaço de estados

As possibilidades de caminhos existentes entre dois pontos pode ser representada por meio de grafos de espaços de estados.

Um caminho ao longo de um grafo conecta uma sequência de nós através de arcos sucessivos. Um caminho é representado por uma lista ordenada que registra os nós na

ordem em que eles ocorrem. Por exemplo, na figura 7, [a, b, c, d] representa o caminho através dos nós *a, b, c* e *d*, nessa ordem.

Figura 7 – Exemplo de grafo



Fonte: Luger (2013)

Segundo Luger (2013), um grafo consiste em um conjunto de nós, que não precisa ser finito, um conjunto de arcos que conectam pares de nós e os caminhos. Um caminho que contém qualquer nó mais do que uma vez são considerados ciclos, ou laços.

Na representação de um problema por espaço de estados, os nós de um grafo correspondem a estados de soluções parciais do problema, e os arcos correspondem a passos de um processo de solução de um problema. A busca em espaço de estados caracteriza a solução de um problema como o processo de procurar um caminho de solução partindo do estado inicial até um objetivo.

A tarefa de um algoritmo de busca é encontrar um caminho de solução que leva de um nó inicial a um nó objetivo. Uma das características gerais de um grafo, e um dos problemas que surgem no desenvolvimento de um algoritmo de busca em grafo, é que às vezes os estados podem ser alcançados por diferentes caminhos. Observando novamente a figura 7, existe mais de um caminho que pode ser feito do estado a para o estado d, por exemplo, pode-se tomar o caminho $[a, b, c, d]$ ou simplesmente o caminho $[a, d]$. Por isso é importante escolher o melhor caminho entre dois pontos, de acordo com a necessidade do problema. Além disso, vários caminhos a um estado podem levar a laços ou ciclos em um caminho, que impedem que o algoritmo alcance um objetivo. Por exemplo, ainda na figura 7, uma busca cega para o estado objetivo e poderia buscar a sequência de estados $[a, b, c, d, a, b, c, d, a, b, c, d, ...]$, infinitamente.

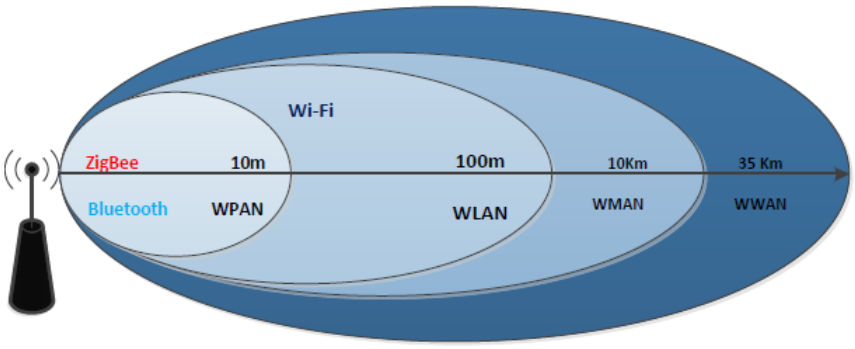
2.3 Comunicação Sem Fio

A comunicação sem fio é a transferência de informação entre dois ou mais pontos, que não estão ligados fisicamente. Esse tipo de comunicação (*wireless*) já está inserida na sociedade em redes como WPANs, WLANs, WMANs e WWANs, são divididos conforme o alcance do sinal e áreas de aplicação. Essas redes têm como objetivo a transferência de

grandes volumes de dados a altas velocidades e a substituição de cabos, possibilitando assim a flexibilidade nas comunicações(FERNANDES, 2012).

A figura 8 ilustra a classificação das redes de comunicação sem fio.

Figura 8 – Redes de comunicação sem fio



Fonte: Fernandes (2012)

Wireless Personal Area Network (WPAN) são as redes sem fio que podem atingir 10 metros de alcance. Redes locais que cobrem uma área maior, como um quarto, um edifício ou até mesmo um campus, porém atingem uma área geográfica limitada, são chamadas *Wireless Local Area Network* (WLAN), essas redes podem atingir 100 metros de alcance. Redes que abrangem uma área metropolitana, permitindo ligações sem fios entre diferentes localizações, como vários edifícios de escritórios em uma cidade ou em um espaço universitário, são chamadas *Wireless Metropolitan Area Network* (WMAN). E finalmente, redes sem fios que abrangem uma grande área geográfica, como um estado ou país, são chamadas *Wireless Wide Area Network* (WWAN) e são possíveis com a utilização de antenas em vários locais ou sistemas de satélite.

A flexibilidade e mobilidade fornecidas pelas redes de comunicação sem fio explicam o rápido crescimento desta tecnologia, além da possibilidade de formar redes dinamicamente, com baixo custo e fácil implementação.

Geralmente, as redes de comunicação sem fios são asseguradas por três tecnologias: Wi-Fi, Bluetooth e ZigBee.

2.3.1 Tecnologia Bluetooth®

De acordo com Miller (2001), Bluetooth® é uma tecnologia que permite a comunicação sem fio entre dispositivos eletrônicos de curto alcance, com baixo consumo de energia e custo. A banda de operação desta tecnologia é a ISM (*Industrial, Scientific and Medical*) 2.4 GHz e utiliza o padrão IEEE 802.15.1.

Para se estabelecer a comunicação nos sistemas Bluetooth®, primeiro o dispositivo deve identificar dispositivos na vizinhança e, em seguida, tem que haver um circuito

pré-estabelecido. A conexão entre dois dispositivos para comunicação pode até acontecer sem uma explícita intervenção manual do usuário.

A popularização desta tecnologia é identificada por sua disponibilidade em quase todos os celulares modernos, sendo bastante utilizada para eliminar o uso de cabos para conectar periféricos ao computador, como por exemplo, mouses, teclados, joysticks e impressoras. Além disso, hoje em dia o Bluetooth® está presente nos automóveis, nas casas inteligentes e até na robótica móvel.

2.3.2 Tecnologia ZigBee®

Segundo Šafarić e Malarić (2006), o ZigBee® é um padrão global para comunicação sem fios, que tem foco na padronização e permite interoperabilidade de produtos. O nome veio da analogia ao modo como as abelhas se movimentam entre as flores e trocam informações com outras abelhas, sobre onde encontrar alimento.

A ZigBee Alliance é um consórcio industrial que promove e desenvolve redes sem fio para fins de controle e monitoramento industrial, que podem também ser utilizados como redes domésticas. Visando o desenvolvimento de uma tecnologia para criar um padrão de baixo consumo de energia, baixo custo, segurança, confiabilidade e com funcionamento em rede sem fios baseado em uma norma aberta global, a ZigBee Alliance, em conjunto com a IEEE (*Institute of Electrical and Eletronics Engineers*), desenvolveu a tecnologia ZigBee®.

Esta tecnologia utiliza o protocolo 802.15.4 para aplicações de rede e permite comunicações robustas, operando com frequência ISM, não necessitando de licença para funcionamento na maioria dos países, como no Brasil.

As redes ZigBee® oferecem uma excelente imunidade contra interferências, e a capacidade de hospedar milhares de dispositivos em uma rede (mais de 60.000), com taxas de transferências de dados a 250Kbps.

Existem dois conjuntos de recursos desta tecnologia, o ZigBee® PRO e o ZigBee®. Os dois operam em redes em malha, porém o ZigBee® PRO é otimizado para apoiar redes com milhares de dispositivos enquanto que o ZigBee® é destinado às redes com menor número de dispositivos.

2.3.3 Tecnologia Wi-Fi®

A *Wireless Fidelity* (Wi-Fi®) é uma tecnologia de redes locais sem fios (WLAN) com base nas especificações da IEEE 802.11. Wi-Fi® é uma marca registrada originalmente pela Wi-Fi Alliance, que foi fundada em 1999 por cinco companhias: 3Com, Aironet, Lucent Tecnologies, Nokia e Symbol Tecnologies (FERNANDES, 2011)

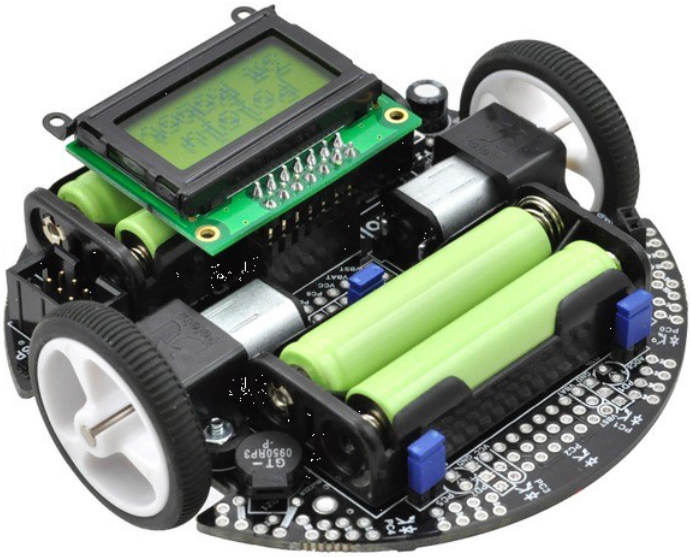
Essa tecnologia permite que o usuário tenha acesso à internet em qualquer lugar, contanto que haja uma rede Wi-Fi®. Sua utilização no acesso à internet é grande devido a altas taxas de dados quando comparadas com as tecnologias Bluetooth® e ZigBee®. Como

essa tecnologia possui um elevada mobilidade, é muito utilizada na criação de redes em hotéis, faculdades, bibliotecas, lojas, hospitais entre outros.

2.4 Robô Pololu 3pi

Segundo Rainwater (2009), o robô Pololu 3pi foi o robô mais exaustivamente testado em campo por sua equipe do que qualquer outro robô. Esse robô é muito utilizado em competições de robótica, seguidores de linha e resolução de labirintos. Ele representa uma ampla plataforma para pessoas com experiência em programação C para aprender robótica, e é um ambiente divertido para iniciantes ambiciosos em programação C. Este robô foi projetado especialmente para seguir linhas e resolver labirintos.

Figura 9 – Pololu 3pi



Fonte: Pololu Corporation (2014)

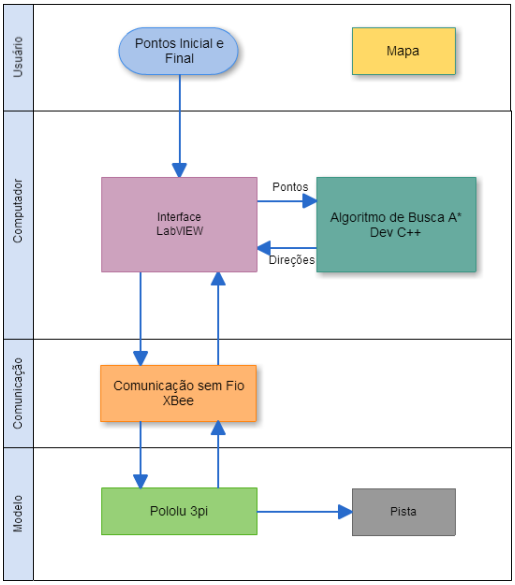
O Pololu 3pi é um robô pequeno, apresentado na figura 9, de alto desempenho e autônomo. Ele é alimentado por quatro pilhas AAA e um sistema único de potência que fornece 9,25V para dois motores, o robô atinge a velocidade de 100 cm/s enquanto executa curvas e rotações com precisão, não sofrendo influência da variação de tensão das baterias, resultando em alto desempenho nas atividades repetitivas, mesmo com as baterias em níveis baixos. O Pololu 3pi vem totalmente montado com dois motores DC com caixa de redução, cinco sensores de refletância, um display LCD 8x2, entre outros, tudo conectado a um microcontrolador AVR (POLOLU CORPORATION, 2014).

Possui um microcontrolador ATmega328p utilizando um clock de 20MHz para realizar o controle. O 3pi possui *hardware* e *software* gratuitos, fornecidos pelo fabricante *Pololu Corporation.*

3 MATERIAIS E MÉTODOS

Neste capítulo será descrita a metodologia utilizada no desenvolvimento do projeto, mostrando como foi realizada a conexão entre os elementos envolvidos para se obter as condições de funcionamento do sistema. O fluxograma ilustrado na figura 10, resume de forma prática a metodologia.

Figura 10 – Metodologia do projeto



Fonte: Autora

O usuário deve fornecer as informações de entrada para o sistema, os pontos inicial e final, de onde e para onde o robô deverá se movimentar. Por meio de uma interface de comunicação desenvolvida no software LabVIEW®, que será abordada em detalhes na secção 3.2.1, o usuário pode se comunicar com o sistema, enviando os pontos necessários para se realizar a busca da melhor rota.

A busca da melhor rota é realizada por um algoritmo que foi escrito, e deve ser compilado no software Dev-C++®. O algoritmo desenvolvido é baseado no método A*, e encontra o melhor caminho entre os pontos inicial e final. A lógica do algoritmo, e seu funcionamento, são descritos na secção 3.2.2.

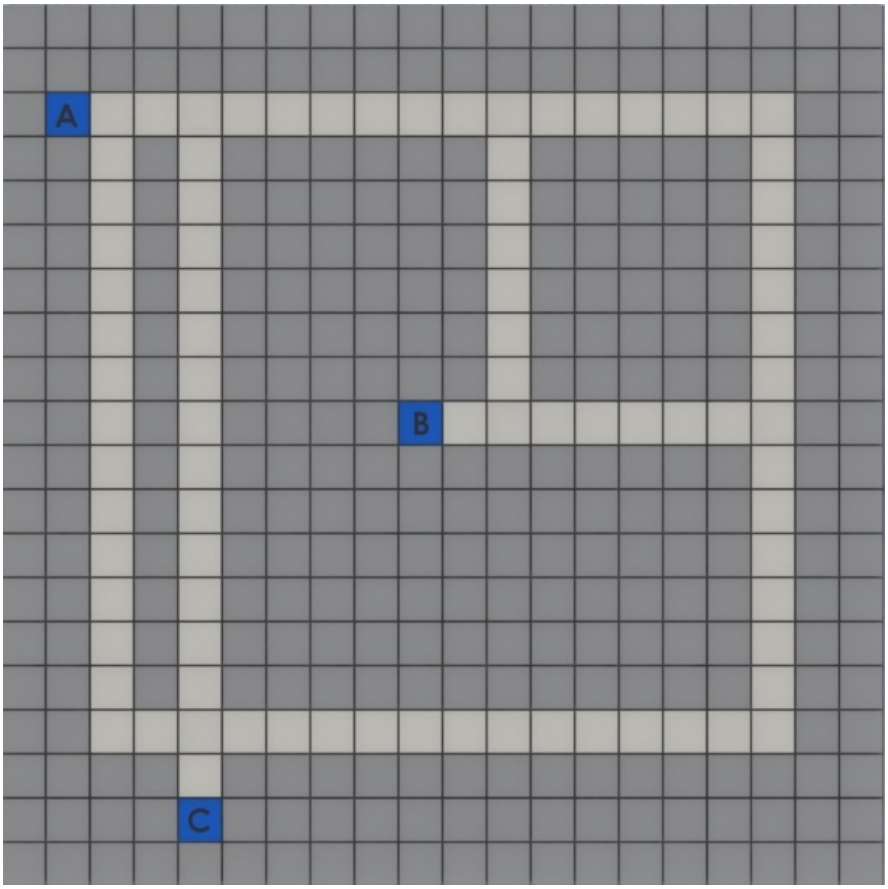
A interface desenvolvida envia os dados, que foram inseridos pelo usuário, para o compilador. Após a execução do algoritmo, o Dev-C++ retorna um vetor com comandos de direções. Este vetor poderá ser visualizado na própria interface e é também mandado para o robô, por meio de uma comunicação sem fio, possível com o uso da tecnologia ZigBee. Na secção 3.3 a comunicação sem fio entre o computador e o robô é abordada em detalhes.

3.1 Usuário

A partir de um conhecimento prévio do ambiente, o usuário deve cadastrar os pontos inicial e final, de onde e para onde o veículo deve se movimentar, por meio da interface que será apresentada na secção 3.2.1. Para exemplificar a escolha dos pontos, foi necessário fazer o desenvolvimento de um ambiente para simular o percurso do AGV.

Uma das primeiras etapas do trabalho foi o desenvolvimento do ambiente, neste trabalho refere-se a esse ambiente como Mapa, na figura 11 pode-se observar este ambiente. Dois mapas serão apresentados neste trabalho, um Mapa e um Mapa virtual. Os dois mapas são equivalentes, ou seja, possuem os mesmos pontos, que estão dispostos também em posições equivalentes, e possuem os mesmos caminhos entre os pontos do mapa.

Figura 11 – Ambiente para simular o percurso do AGV



Fonte: Autora

O ambiente desenvolvido, Mapa, possui três pontos A, B e C, que podem ser relacionados a pontos estratégicos de uma indústria, como almoxarifado, uma linha de produção e estoque de produto final.

Com esses três pontos, o usuário pode escolher um ponto para ser o início da rota, de onde o veículo irá sair, e um ponto para ser o destino da rota, para onde o veículo deve se movimentar.

Vários caminhos foram desenhados entre os pontos, a fim de se conectar cada par de pontos mais de uma vez. Isso criou a possibilidade de se escolher, entre várias rotas, a rota mais adequada entre dois pontos. Este é um ponto importante, pois o objetivo do trabalho é decidir qual a melhor rota a ser tomada, conforme será explicado na secção 3.2.2.

A diferença entre o Mapa virtual e o Mapa é que o primeiro foi construído no computador e o segundo foi construído fisicamente, como será explicado na secção 3.4.2.

3.2 Computador

Nesta secção serão apresentados o desenvolvimento da interface de comunicação entre o usuário e o robô e o algoritmo de busca desenvolvido para a tomada de decisão. O computador atua como um ambiente central, responsável por compilar o algoritmo no *software* Dev-C++, para calcular o percurso a ser seguido, e controlar a comunicação do usuário com o robô.

3.2.1 Interface de Comunicação

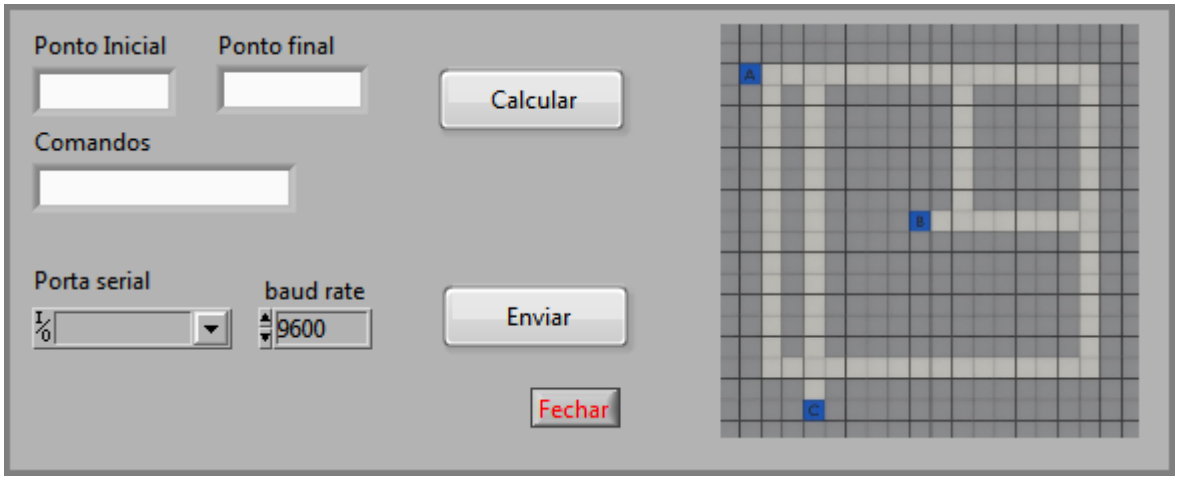
O *software* NI *(National Instruments)* LabVIEW é utilizado para uma grande variedade de aplicações e setores da indústria. Ele é um ambiente de desenvolvimento altamente produtivo, para a criação de aplicações customizadas e que interagem com os dados ou sinais do mundo real, em áreas como ciência e engenharia.

O LabVIEW utiliza uma linguagem de programação gráfica, criando aplicações utilizando ícones e não linhas de texto. Utiliza assim o fluxo de dados para determinar a execução de ações. Essa linguagem, denominada "G", é um modelo de programação bastante intuitivo, semelhante a um fluxograma, porém possui a mesma potencialidade de uma linguagem textual, como C ou Pascal, por exemplo.

Para o desenvolvimento da interface do projeto foi utilizado o *software* LabVIEW 2012, na edição para estudantes.

A interface desenvolvida para comunicação do usuário com o robô abrange a tela de interface, disponível para o usuário, e funções para o controle do cálculo da rota e da comunicação serial. A figura 12 apresenta a tela de interface desenvolvida.

Figura 12 – Tela de interface de comunicação



Fonte: Autora

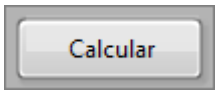
Nos campos 'Ponto Inicial' e 'Ponto Final' o usuário pode inserir os pontos de onde e para onde o robô deverá se movimentar, conforme figura 13. Os pontos Inicial e Final que devem ser inseridos nos campos apresentados devem ser A, B ou C, conforme Mapa apresentado na tela de interface.

Figura 13 – Inserção dos pontos inicial e final

Fonte: Autora

O usuário deve então selecionar o botão Calcular, para que o percurso seja calculado, figura 14.

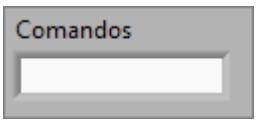
Figura 14 – Botão para iniciar o cálculo dos comandos



Fonte: Autora

Neste momento o LabVIEW chama um aplicativo gerado para executar o algoritmo de busca A*, desenvolvido no projeto para encontrar o menor caminho entre os dois pontos inseridos e que será apresentado na secção 3.2.2, e este executável gera um arquivo de texto contendo os comandos equivalentes ao menor caminho entre os pontos. Esta rota pode então ser visualizada na tela de interface, figura 15.

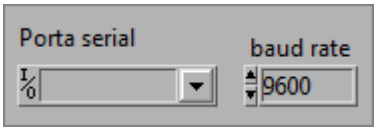
Figura 15 – Apresentação dos comandos



Fonte: Autora

Para realizar a comunicação entre o computador e o XBee, que deve estar conectado ao computador, a Porta serial e a *baud rate* (taxa de transmissão de dados) devem ser configuradas, nos campos apresentados na figura 16.

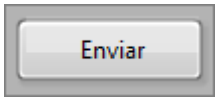
Figura 16 – Configuração da comunicação serial



Fonte: Autora

Assim é possível enviar os comandos para o XBee conectado ao computador, que em seguida enviará para o XBee conectado ao robô, para que o mesmo possa executar os comandos e então se movimentar em direção ao ponto de destino. O envio dos comandos ocorre mediante o acionamento do botão Enviar, figura 17.

Figura 17 – Botão para envio dos comandos



Fonte: Autora

A programação gráfica da interface foi desenvolvida em dois estados, Calcular e Enviar, executando ações quando algum botão for acionado. Os diagramas de blocos para os dois estados mencionados estão disponíveis no apêndice A.

3.2.2 Algoritmo de Busca A*

Primeiramente foram estudados alguns caminhos possíveis entre os pontos do Mapa, foi desenvolvida uma busca de caminhos para cada situação possível no projeto. E então foi realizada a implementação do algoritmo A*. E finalmente foi executada a transformação do vetor rota em comandos para o robô.

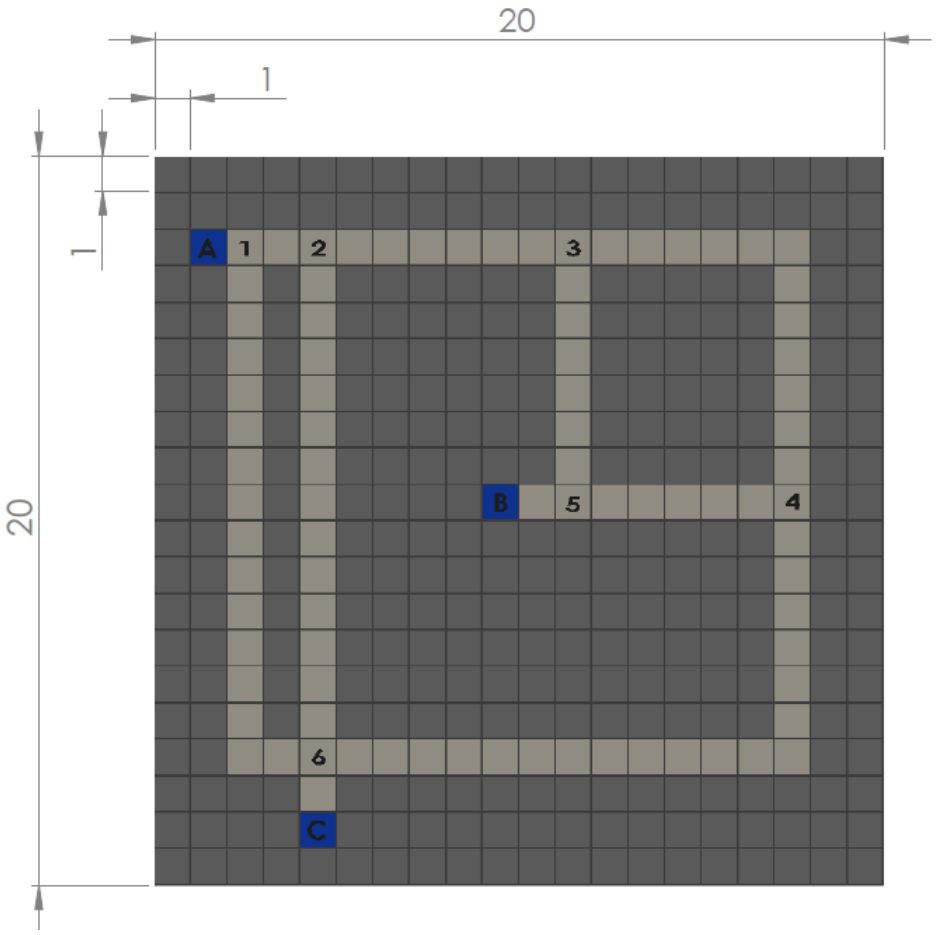
3.2.2.1 Busca por caminhos possíveis

Para a realização dos estudos dos caminhos do Mapa, o mesmo foi desenhado em uma área quadriculada e nós foram atribuidos a alguns pontos do Mapa. Às posições do Mapa,

onde existe a possibilidade de se seguir por mais de um caminho, foram atribuídos nós. No total o Mapa possui seis nós, e na figura 18 podem ser observadas suas localizações.

Para efeito de cálculo, foi convencionado que cada quadrado do Mapa desenhado corresponde a uma unidade de medida, como pode também pode ser observado na figura 18.

Figura 18 – Mapa em escala



Fonte: Autora

Desta forma, alguns caminhos possíveis puderam então ser representados por espaços de estados, independente de ser ou não o caminho mais curto entre o ponto inicial e o final.

Os caminhos foram traçados a fim de se realizar uma comparação entre os caminhos possíveis entre dois pontos e o caminho encontrado pelo algoritmo desenvolvido no projeto. Foram construidos grafos para as seguintes situações: de A para B, de A para C, de B para A, de B para C, de C para A e de C para B.

Não foi utilizada a busca em profundidade e nem a busca em amplitude, pois a representação dos espaços de estados dessas buscas é extremamente grande, uma vez que existem muitas possibilidades de caminhos entre dois pontos e o objetivo de mostrar uma representação dos espaços de estados é somente para ilustrar alguns caminhos possíveis.

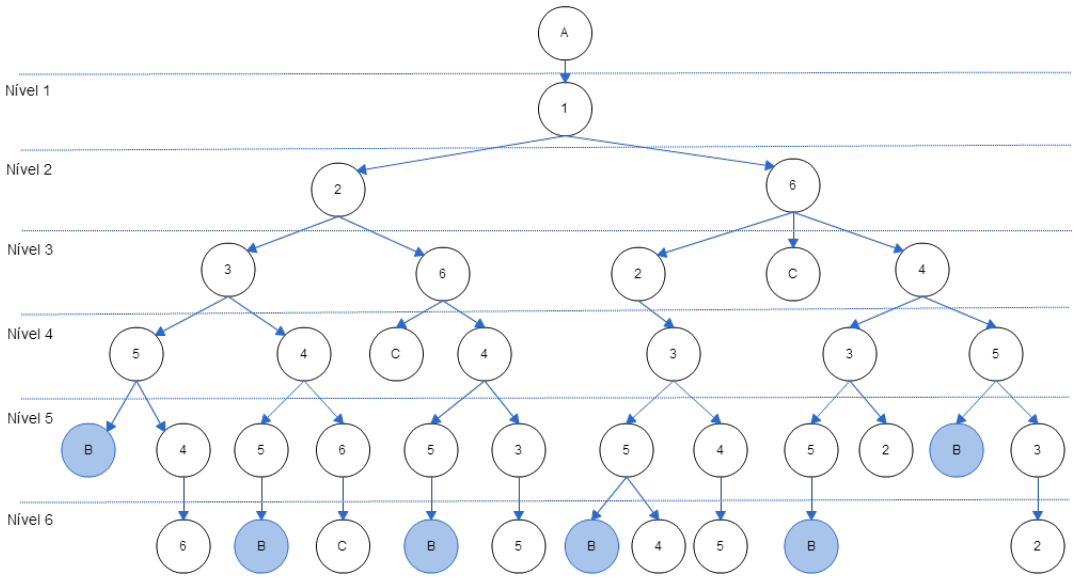
Algumas situações foram evitadas, como a expansão de nós que foram expandidos anteriormente, no mesmo caminho, por formarem laços. Esses nós repetitivos, não foram incluídos nos grafos apresentados neste trabalho.

A seguir serão mostradas as representações em espaços de estados para cada uma das situações citadas acima, e alguns caminhos encontrados.

- Situação 1: de A para B

Por meio da figura 19 podemos visualizar seis possíveis caminhos a se tomar para sair do ponto A e chegar ao ponto B.

Figura 19 – Representação por espaço de estados de A para B



Fonte: Autora

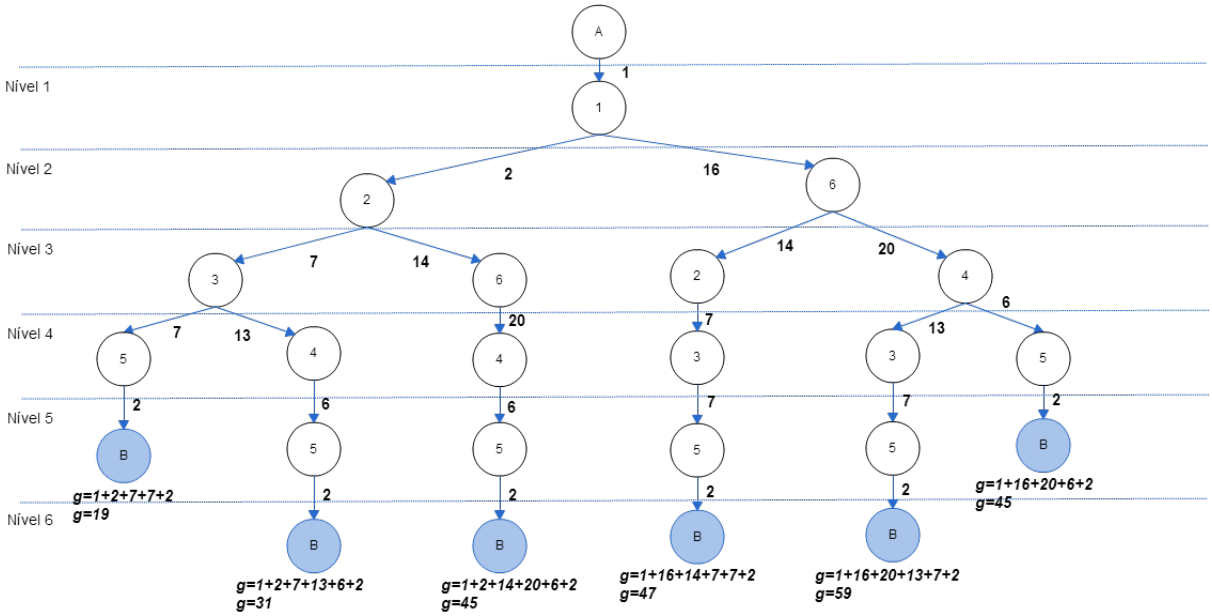
Uma vez encontrados alguns caminhos do ponto inicial ao ponto final, podemos estudar em detalhes esses caminhos.

Para isso faremos os cálculos de custo dos caminhos. Essa ainda não é a aplicação do algoritmo A* para busca do menor caminho, e sim o estudo individual de cada caminho encontrado por meio da representação em espaço de estados.

Os arcos, que são representados pelas setas, mostram as distâncias entre os nós, ou seja, o custo para se mover entre os dois nós conectados pelo arco, e auxiliam no cálculo do custo g de cada caminho. O cálculo de g é feito simplesmente somando-se todos os valores dos arcos seguidos no caminho.

Na figura 20 podem ser observados os cálculos dos custos g para cada caminho encontrado, o custo g é referente ao caminho percorrido desde o ponto inicial até o nó em questão, como foi explicado brevemente na secção 2.2.4. Neste momento, mostraremos apenas o custo g final de cada caminho, que representa a distância real que deve ser percorrida, se o caminho em questão for escolhido.

Figura 20 – Custos para os caminhos de A para B



Fonte: Autora

Com base nas representações por espaço de estados de cada situação, foram criadas tabelas de custo de movimentação, para denominar cada caminho encontrado, atribuir a cada um desses caminhos o custo de movimentação e seu percurso e fazer a comparação entre eles, posteriormente. A seguir, a tabela 1 mostra os custos de movimentação dos caminhos encontrados para se locomover do ponto A ao ponto B.

Tabela 1 – Caminhos de A para B

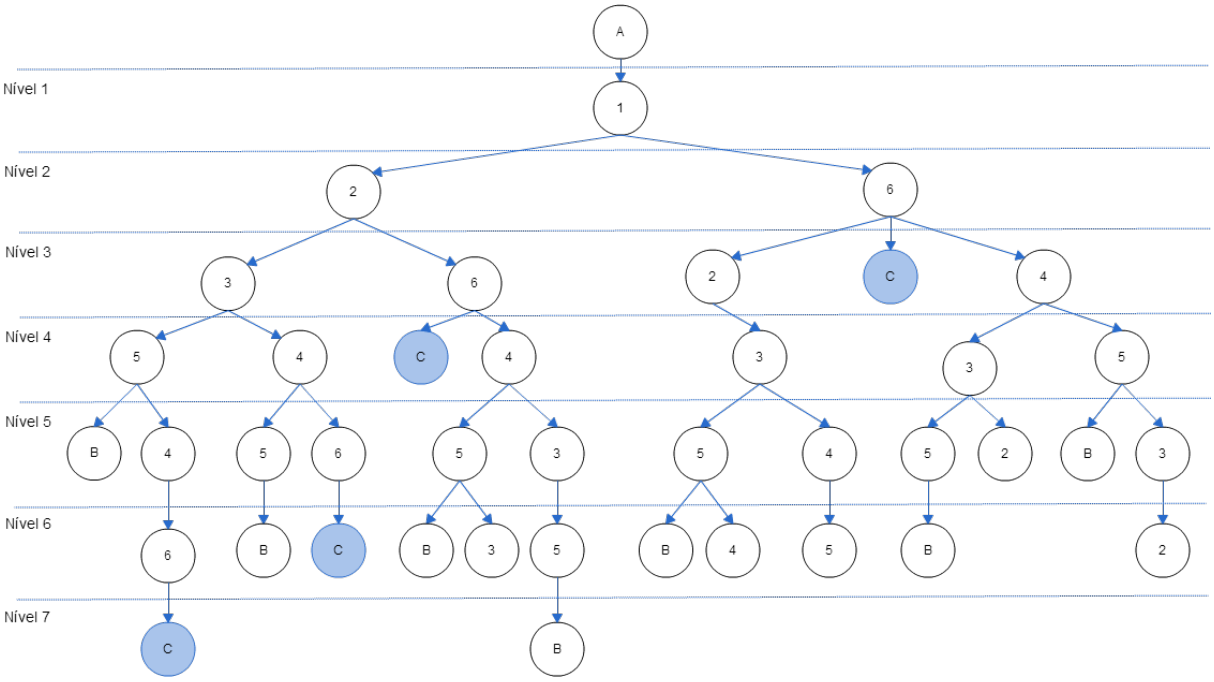
Caminho	Percurso	Custo de movimentação (g)
1	$[A, 1, 2, 3, 5, B]$	19
2	$[A, 1, 2, 3, 4, 5, B]$	31
3	$[A, 1, 2, 6, 4, 5, B]$	45
4	$[A, 1, 6, 2, 3, 5, B]$	47
5	$[A, 1, 6, 4, 3, 5, B]$	59
6	$[A, 1, 6, 4, 5, B]$	45

Fonte: Autora

- Situação 2: de A para C

A figura 21 representa em espaços de estados da busca de caminhos, tendo como início o ponto A e como final o ponto C.

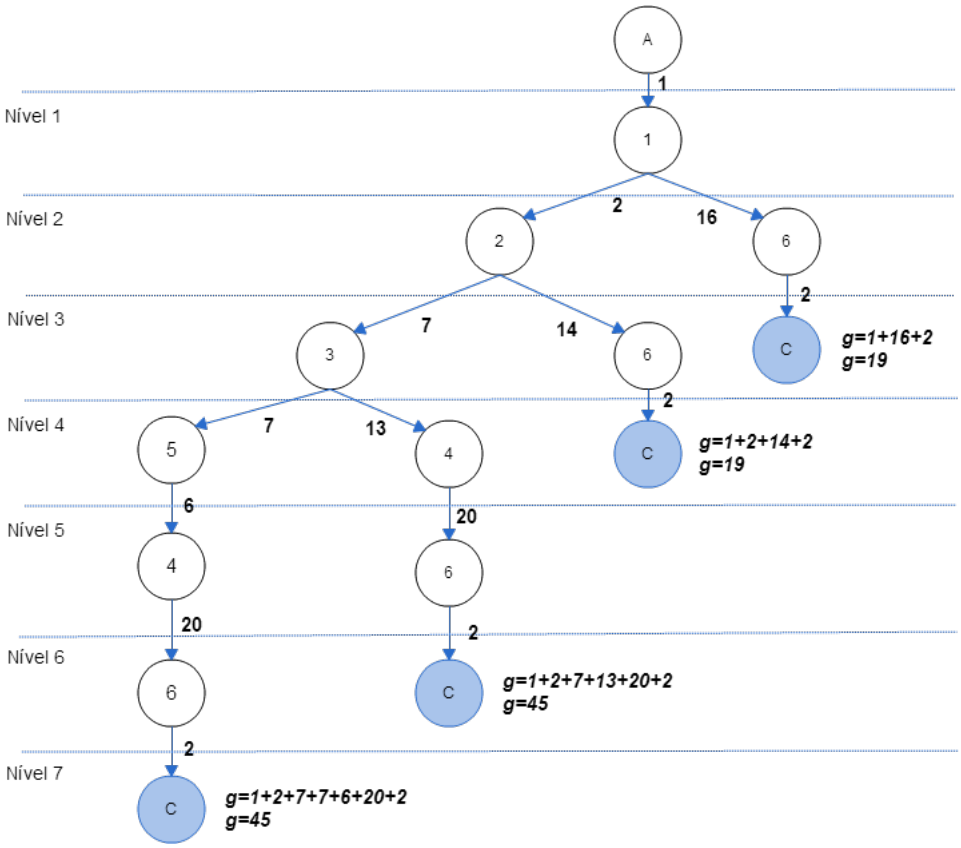
Figura 21 – Representação por espaço de estados de A para C



Fonte: Autora

Os cálculos dos custos g podem ser observados na figura 22, para os caminhos encontrados anteriormente.

Figura 22 – Custos para os caminhos de A para C



Fonte: Autora

A seguir, a tabela 2 mostra os custos de movimentação dos caminhos encontrados para se locomover do ponto A ao ponto C.

Tabela 2 – Caminhos de A para C

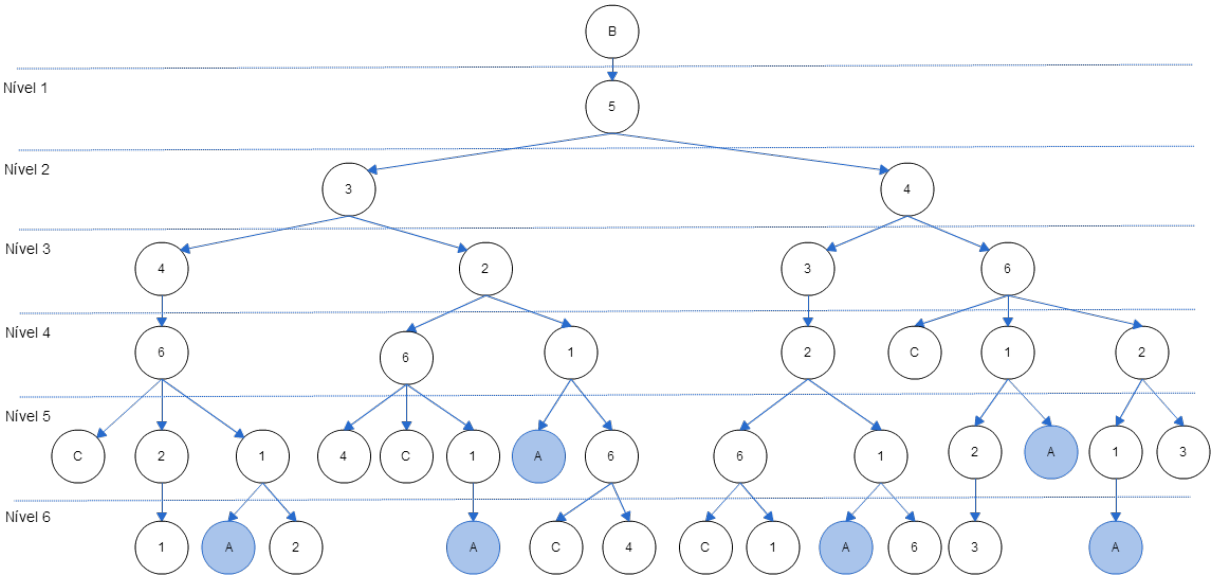
Caminho	Percurso	Custo de movimentação (g)
1	$[A, 1, 2, 3, 5, 4, 6, C]$	45
2	$[A, 1, 2, 3, 4, 6, C]$	45
3	$[A, 1, 2, 6, C]$	19
4	$[A, 1, 6, C]$	19

Fonte: Autora

- Situação 3: de B para A

A figura 23 representa em espaços de estados da busca de caminhos, iniciando no ponto B e tendo como destino o ponto A.

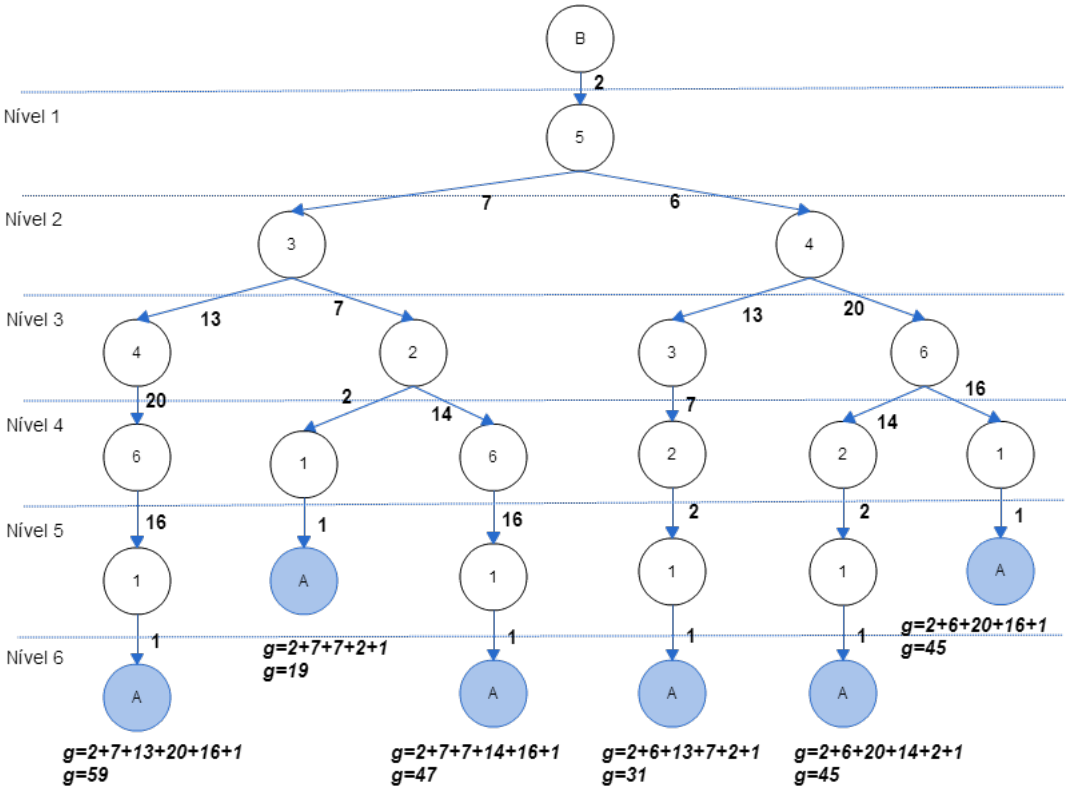
Figura 23 – Representação por espaço de estados de B para A



Fonte: Autora

Os cálculos dos custos g podem ser observados na figura 24, para os caminhos encontrados anteriormente.

Figura 24 – Custos para os caminhos de B para A



Fonte: Autora

A tabela 3 mostra os custos de movimentação dos caminhos encontrados para se locomover do ponto B ao ponto A.

Tabela 3 – Caminhos de B para A

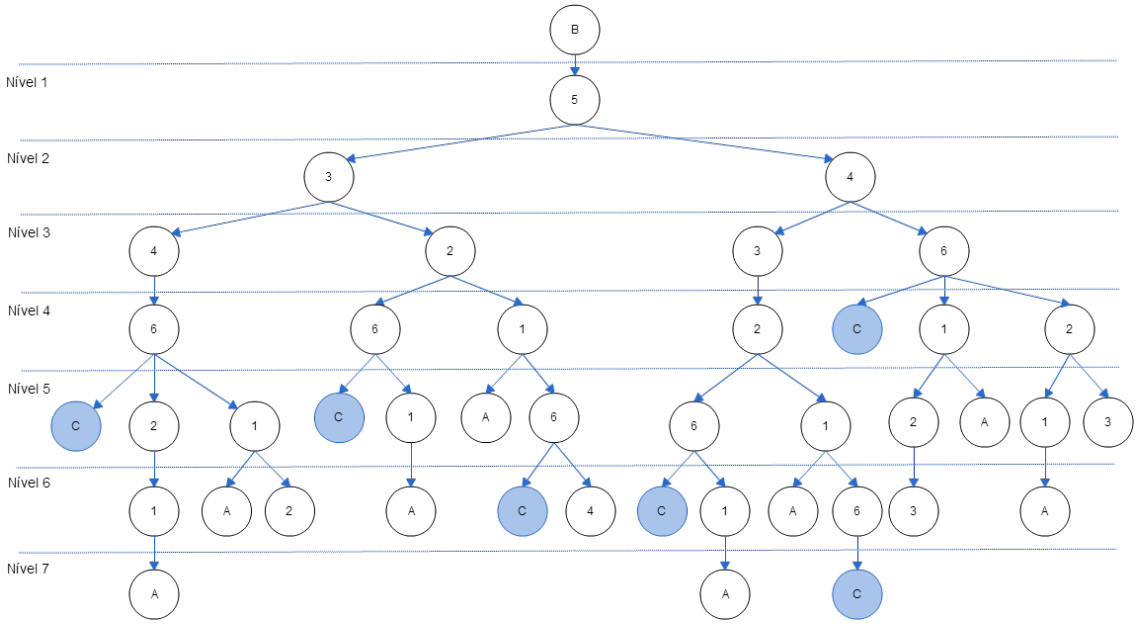
Caminho	Percurso	Custo de movimentação (g)
1	$[B, 5, 3, 4, 6, 1, A]$	59
2	$[B, 5, 3, 2, 6, 1, A]$	47
3	$[B, 5, 3, 2, 1, A]$	19
4	$[B, 5, 4, 3, 2, 1, A]$	31
5	$[B, 5, 4, 6, 2, 1, A]$	45
6	$[B, 5, 4, 6, 1, A]$	45

Fonte: Autora

- Situação 4: de B para C

A representação em espaços de estados da busca de caminhos, partindo do ponto B e chegando no ponto C, pode ser observada na figura 25.

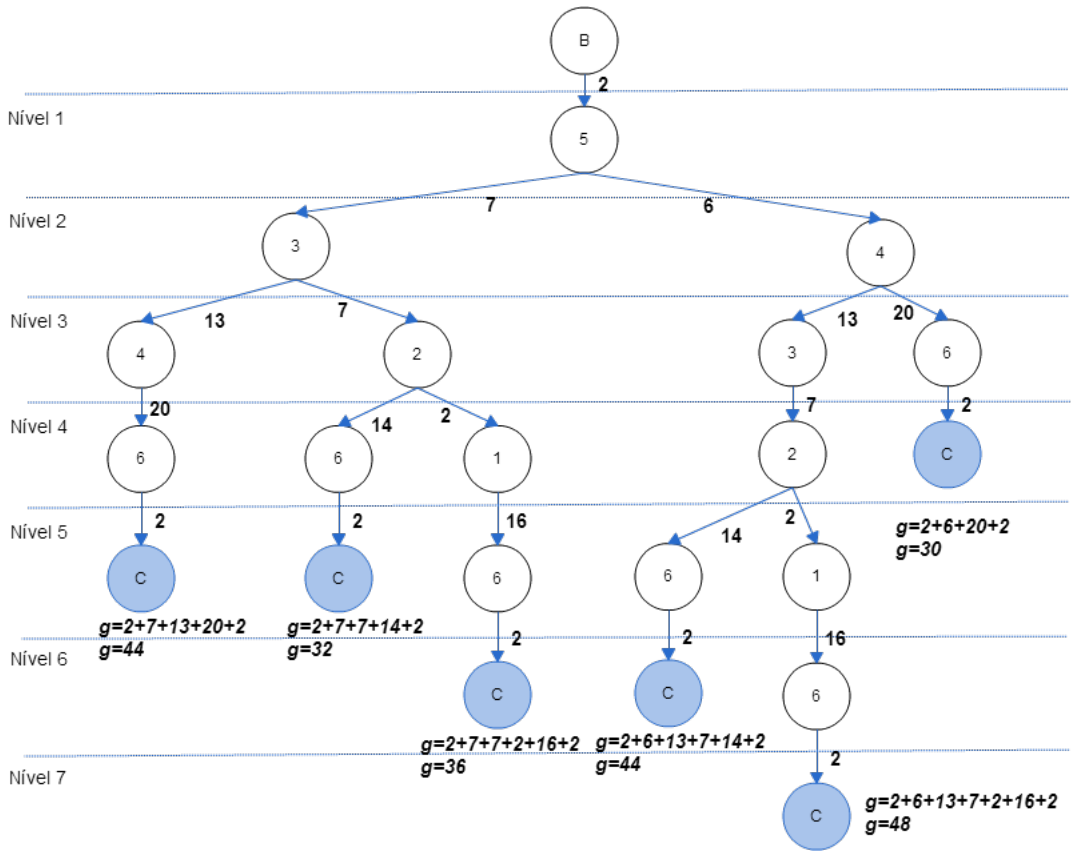
Figura 25 – Representação por espaço de estados de B para C



Fonte: Autora

Os cálculos dos custos g podem ser observados na figura 26, para os caminhos encontrados anteriormente.

Figura 26 – Custos para os caminhos de B para C



Fonte: Autora

A tabela 4 mostra os custos de movimentação dos caminhos encontrados para se locomover do ponto B ao ponto C.

Tabela 4 – Caminhos de B para C

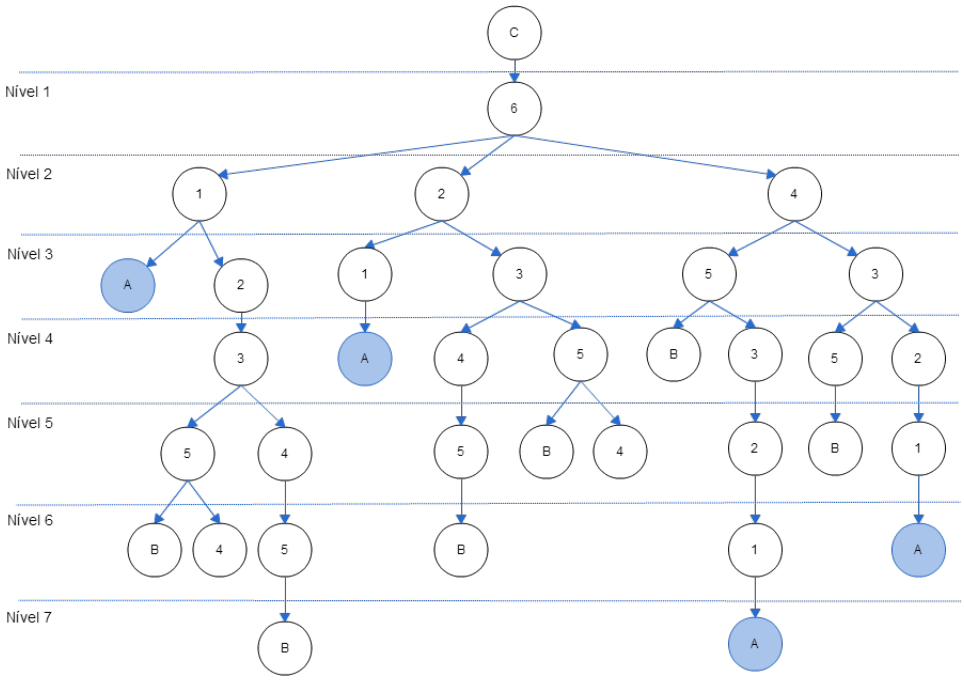
Caminho	Percurso	Custo de movimentação (g)
1	$[B, 5, 3, 4, 6, C]$	44
2	$[B, 5, 3, 2, 6, C]$	32
3	$[B, 5, 3, 2, 1, 6, C]$	36
4	$[B, 5, 4, 3, 2, 6, C]$	44
5	$[B, 5, 4, 3, 2, 1, 6, C]$	48
6	$[B, 5, 4, 6, C]$	30

Fonte: Autora

- Situação 5: de C para A

A figura 27 representa em espaços de estados da busca de caminhos, tendo como início o ponto C e como final o ponto A.

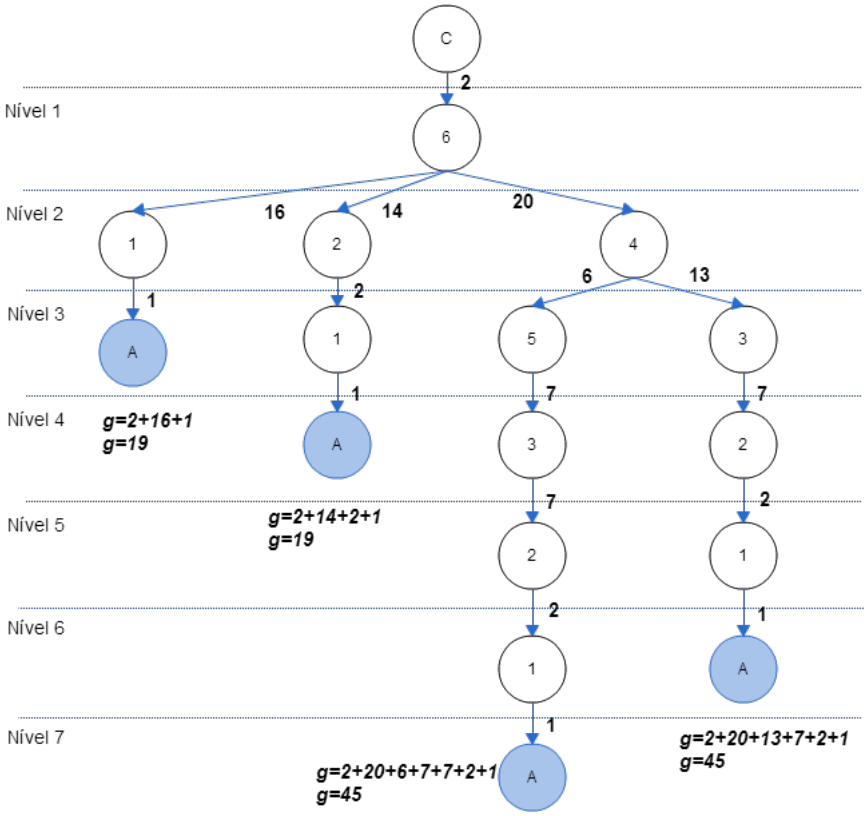
Figura 27 – Representação por espaço de estados de C para A



Fonte: Autora

Os cálculos dos custos g podem ser observados na figura 28, para os caminhos encontrados anteriormente.

Figura 28 – Custos para os caminhos de C para A



Fonte: Autora

A tabela 5 mostra os custos de movimentação dos caminhos encontrados para se locomover do ponto C ao ponto A.

Tabela 5 – Caminhos de C para A

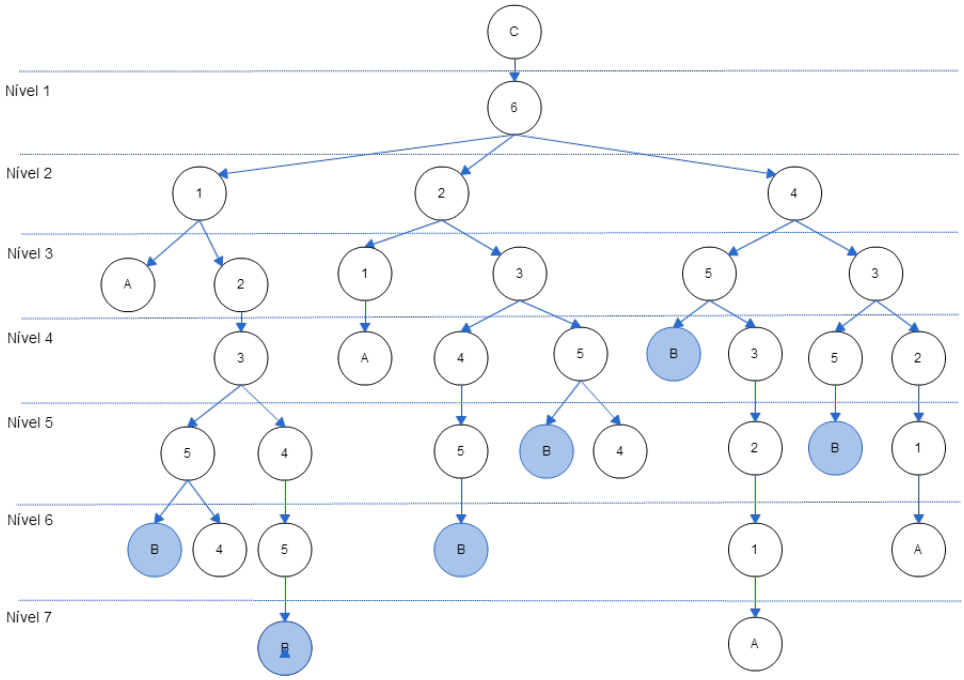
Caminho	Percurso	Custo de movimentação (g)
1	$[C, 6, 1, A]$	19
2	$[C, 6, 2, 1, A]$	19
3	$[C, 6, 4, 5, 3, 2, 1, A]$	45
4	$[C, 6, 4, 3, 2, 1, A]$	45

Fonte: Autora

- Situação 6: de C para B

A representação em espaços de estados da busca de caminhos, partindo do ponto C e chegando no ponto B, pode ser observada na figura 29.

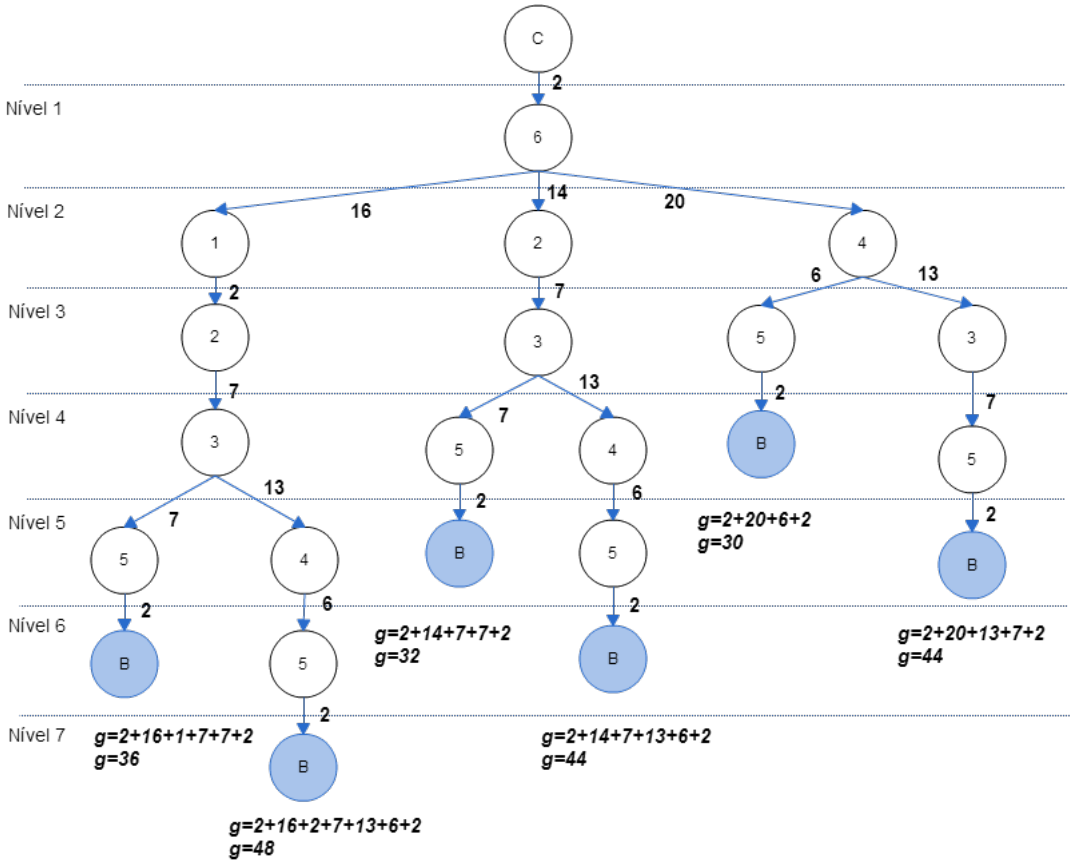
Figura 29 – Representação por espaço de estados de C para B



Fonte: Autora

Os cálculos dos custos g podem ser observados na figura 30, para os caminhos encontrados anteriormente.

Figura 30 – Custos para os caminhos de C para B



Fonte: Autora

A tabela 6 mostra os custos de movimentação dos caminhos encontrados para se locomover do ponto C ao ponto B.

Tabela 6 – Caminhos de C para B

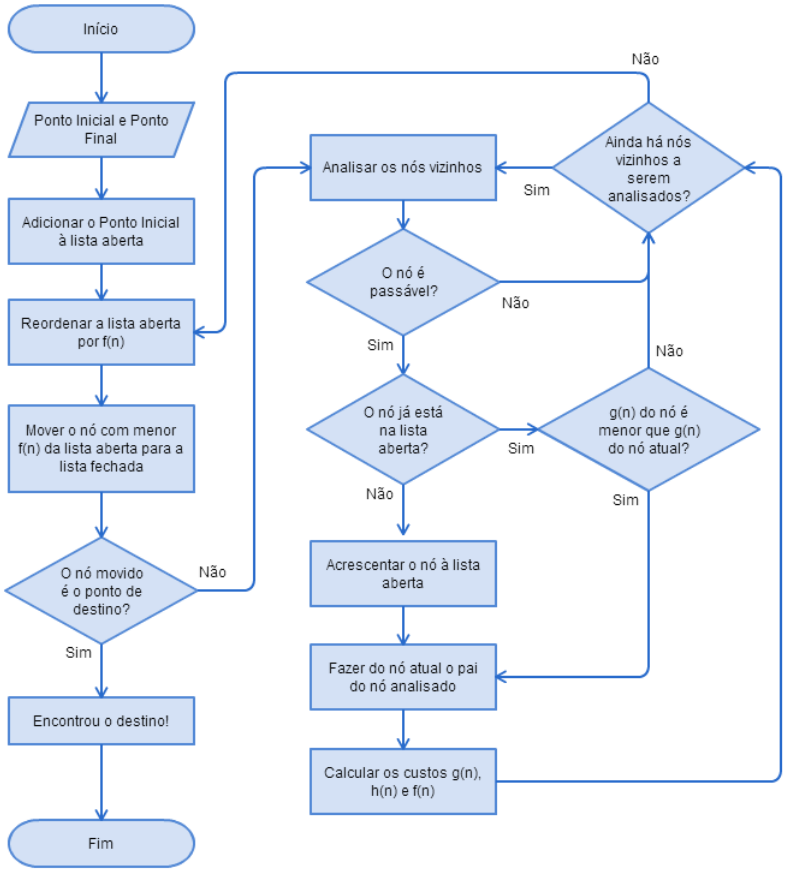
Caminho	Percurso	Custo de movimentação (g)
1	$[C, 6, 1, 2, 3, 5, B]$	36
2	$[C, 6, 1, 2, 3, 4, 5, B]$	48
3	$[C, 6, 2, 3, 5, B]$	32
4	$[C, 6, 2, 3, 4, 5, B]$	44
5	$[C, 6, 4, 5, B]$	30
6	$[C, 6, 4, 3, 5, B]$	44

Fonte: Autora

3.2.2.2 Implementação do Algoritmo A*

A figura 31 resume a lógica do método A*, que serviu de base para o desenvolvimento do algoritmo de busca utilizado neste trabalho para encontrar o menor caminho entre dois pontos.

Figura 31 – Lógica do método A*



Fonte: Autora

Seguindo a lógica de busca resumida pelo fluxograma 31 e explicado mais detalhadamente na secção 2.2.4, foi desenvolvido o algoritmo de busca para o projeto.

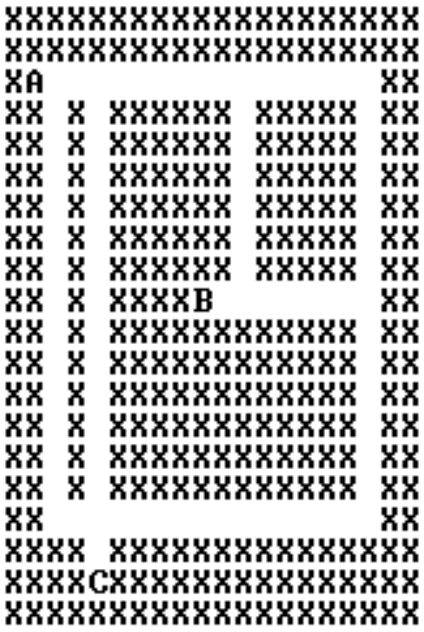
O algoritmo foi implementado em linguagem C++, usando o *software* Dev-C++, que oferece um ambiente de desenvolvimento integrado para o desenvolvimento de aplicações. O Dev-C++ é gratuito e de código aberto, além de possuir todas as funcionalidades padrões necessárias para a escrita, compilação, debugging e execução de programas na linguagem C e C++, como por exemplo a fila de prioridades, *priority queue*.

A fila de prioridades foi utilizada para manter a lista aberta, com os nós que ainda seriam estudados, quando acessada, a fila de prioridades retorna o nó com maior prioridade, ou seja o nó com menor custo heurístico $h(n)$, como já foi explicado anteriormente.

Como foi mencionado na secção 3.1, foi criado um Mapa virtual para simular o cálculo da rota de menor percurso pelo algoritmo A* e para visualizar o comportamento do mesmo.

O Mapa virtual foi construído para simular o cálculo da rota de menor percurso pelo algoritmo A* e visualizar o comportamento do algoritmo desenvolvido neste trabalho. Esse mapa é representado virtualmente por uma matriz 20x20, onde cada posição da matriz pode ser preenchida por um dos seguintes caracteres, 'A', 'B', 'C' e 'X'. Onde 'A', 'B' e 'C' são os pontos que servirão como início ou destino do percurso, de onde ou para onde o AVG irá se movimentar, 'X' representa um espaço geométrico onde o AVG não têm acesso, por se tratar de paredes ou bloqueios já mapeados. O não preenchimento de um elemento da matriz representa um espaço geométrico acessível ao AVG, sendo ele parte de um caminho possível a ser seguido. As posições matriciais sem preenchimento, ou preenchidas por 'A', 'B' e 'C', são consideradas pelo algoritmo na decisão do melhor caminho, pois são vistos pelo algoritmo como estados passáveis, enquanto que as posições em 'X' são ignoradas, pois são estados não passáveis para o algoritmo. Uma representação de Mapa virtual seguindo os princípios citados pode ser vista na figura 32.

Figura 32 – Mapa virtual



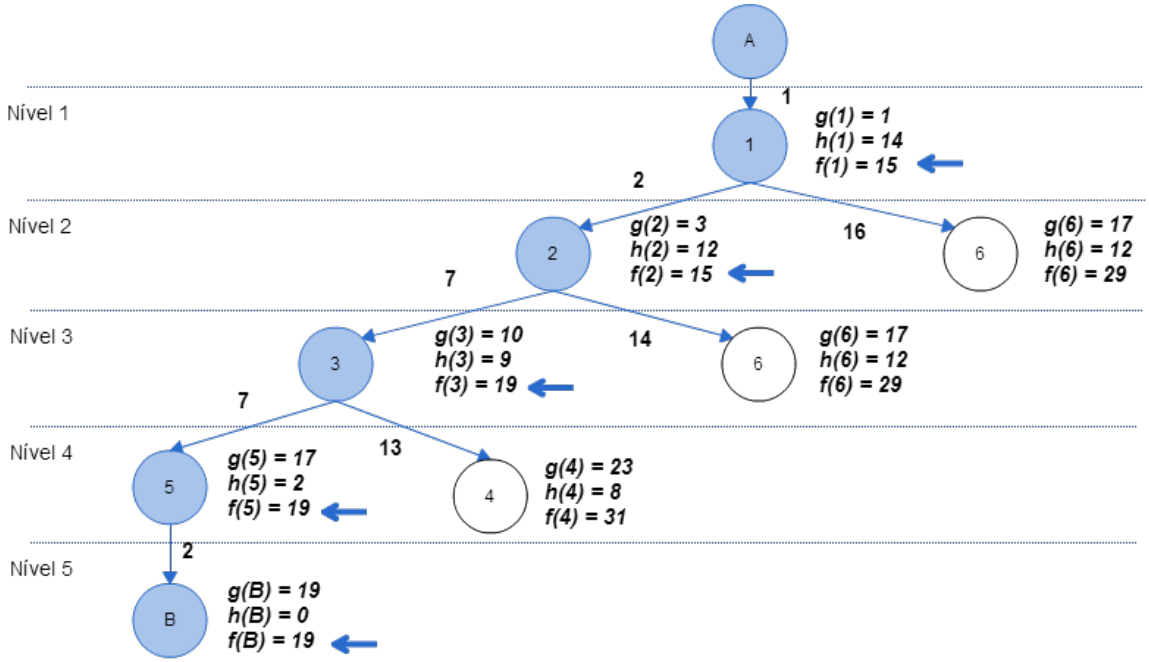
Fonte: Autora

O caminho é encontrado, se escolhendo quais posições da matriz se deve seguir para ir de um ponto inicial a um ponto final, por exemplo, do ponto A ao ponto B, que será o exemplo utilizado para explicar o algoritmo desenvolvido. Uma vez que o caminho é encontrado, pode-se visualizar o Mapa virtual final na simulação, onde o caminho encontrado é simbolizado por pontos.

Neste trabalho foi utilizada a distância Manhattan para realizar os cálculos dos custos heurísticos, $h(n)$, conforme equação 2.3.

A figura 33 ilustra a lógica do algoritmo A*, conforme foi descrito nos passos de 1 a 7, da explicação do método A* na secção 2.2.4. Os nós em cor azul são os nós que foram escolhidos para serem expandidos, eles foram escolhidos, pois possuíam os menores valores da função de avaliação, $f(n)$, no momento em que estavam sendo estudados. Todos os nós mostrados na figura passaram pela lista aberta, porém os nós em branco não foram expandidos, ou seja, seus vizinhos não foram estudados, pois possuiam maiores valores de $f(n)$, e então continuaram na lista aberta, enquanto que os nós em cor azul, foram expandidos e movidos da lista aberta para a lista fechada, no momento em que estavam sendo estudados.

Figura 33 – Representação da lógica do algoritmo A*



Fonte: Autora

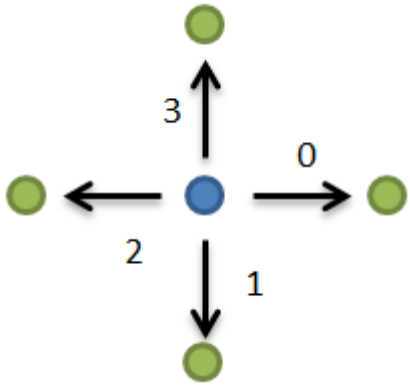
Quando a busca é finalizada, por meio da lista fechada se tem acesso ao caminho encontrado pelo algoritmo, que deve corresponder ao menor caminho entre os pontos inicial e final. Pela relação de parentesco entre os nós da lista fechada, é possível organizar os nós na sequência correta em que devem ser seguidos para se percorrer o caminho do ponto inicial ao final. No exemplo utilizado o caminho encontrado foi$[A, 1, 2, 3, 5, B]$, como também pode ser visto na figura 33.

Quando compilado no Dev-C++, o algoritmo desenvolvido recebe como valores de entrada os pontos inicial e final, executa a lógica, realizando o cálculo da rota, e retorna um vetor com as direções que devem ser seguidas para se mover do ponto inicial ao final, pelo menor caminho.

3.2.2.3 Transformação da rota em comandos

Após efetuado o cálculo do percurso pelo algoritmo, ele retorna um vetor, como uma lista de direções a serem seguidas para se traçar o caminho do ponto inicial ao final, no Mapa virtual. Os caracteres utilizados pelo algoritmo para representar as direções, neste primeiro momento são: ‘0’,’1’,’2’ e ‘3’ e significam que a próxima posição a se assumir no Mapa virtual está a direita, abaixo, a esquerda ou acima da posição atual, respectivamente. A figura 34 ilustra esses comandos e suas direções.

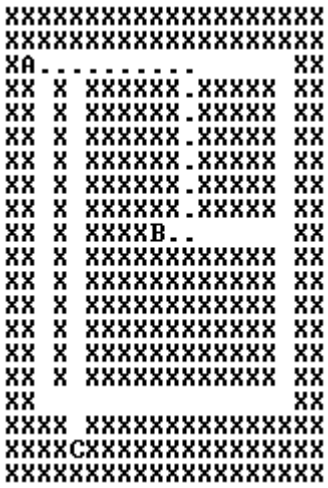
Figura 34 – Comandos de direção para o Mapa virtual



Fonte: Autora

Porém esses comandos não levam em consideração a orientação do robô no momento em que ele irá realizar o comando solicitado. Por exemplo, quando se compila o algoritmo para encontrar o melhor percurso entre os pontos A e B, ele nos retorna o caminho que pode ser observado na figura 35.

Figura 35 – Simulação do algoritmo de A para B



Fonte: Autora

Em cada interseção, ou seja, em cada nó, o algoritmo deve tomar uma decisão e escolher uma direção a ser seguida. Essa decisão irá gerar um comando de direção: para esquerda, para direita, para baixo ou para cima. Porém como no Mapa virtual a posição ou orientação do robô não importam, por se tratar apenas de uma simulação, os comandos de direção são calculados conforme a posição, (x, y), do nó anterior.

Assim sendo, no primeiro nó do exemplo mostrado na figura 35, o comando enviado será '0', representando uma movimentação, no Mapa virtual, para a direita, assim como o comando enviado para a segunda interseção, nó 2, também será '0', para a direita, na terceira interseção, nó 3, o comando enviado será '1', significando que o caminho traçado

neste momento segue para baixo. E finalmente na quarta interseção, nó 5, o comando enviado será '2', pois o caminho então segue para a esquerda.

Resumindo os comandos, se tem a rota $[0, 0, 1, 2]$, para as interseções, ou seja, os nós. Esses comandos significam: 'direita', 'direita', 'baixo' e 'esquerda'. E é chamada de 'rota' o primeiro vetor que o algoritmo retorna, com os comandos acima, por exemplo.

Por esse motivo, foi implementada uma lógica para transformar o vetor rota, gerado pelo algoritmo para simulação no Mapa virtual, em um vetor de comandos, contendo os comandos exatos a serem executados pelo robô no Mapa. As possíveis direções que o robô pode tomar são: frente, direita ou esquerda, representadas pelas letras 'F', 'D' e 'E', respectivamente.

A lógica implementada foi estudada e testada diversas vezes, até atender ao requisito de retornar os comandos corretamente, levando em consideração a orientação do robô. No exemplo acima, onde a rota é $[0, 0, 1, 2]$, após a implementação da lógica mencionada, se obteve o seguinte vetor de comandos: $[F, F, D, D]$, então o robô pode se movimentar corretamente no Mapa, uma vez que sua posição inicial é saindo do ponto A em direção ao nó 1.

O vetor de comandos pode então ser enviado ao robô, que seguirá corretamente o caminho, uma vez que seus comandos levam em consideração a orientação do robô em cada interseção do Mapa.

3.3 Comunicação

Baker, apud Pothuganti e Chitneni (2014), estudou as vantagens e desvantagens do uso das tecnologias ZigBee® e Bluetooth® em aplicações industriais. Ele afirma que o ZigBee® pode atender a uma maior variedade de necessidades industriais que o Bluetooth®, devido à sua operação de longa duração da bateria, ao seu maior alcance útil, à flexibilidade em um número de dimensões e à segurança de sua arquitetura de rede em malhas.

3.3.1 XBee

O módulo utilizado no projeto é o XBee, especificamente o XBee Série 1 (S1), fabricado pela Digi Internacional, e oferece ligações sem fios por rádio frequência (RF) entre os módulos, e utiliza como protocolo de rede a norma IEEE 802.15.4, tecnologia ZigBee. Na figura 36 pode ser visto o XBee utilizado no desenvolvimento deste projeto e a tabela 7 informa as principais características do XBee utilizado.

Figura 36 – Módulo XBee Série 1

Fonte: http://goo.gl/l5d6yW

Tabela 7 – Principais características do XBee S1

Características	XBee S1
Norma/Protocolo	IEEE 802.504
Tipo de dispositivo lógico	Coordenador/ End-devices
Topologia de rede	Topologia em estrela
Alcance em áreas internas ou urbanas	30m
Alcance em linha de visão (campo aberto)	100m
Taxa de transferência	250Kbps
Tensão de alimentação	2.8 – 3.4 V
Frequência	ISM 2.4 GHz
Temperatura de operação	-40 a 85ºC (industrial)
Opções de endereçamento	PAN ID, canal e endereços

Fonte: http://goo.gl/MHRV

3.3.1.1 Configuração do XBee

A configuração dos módulos XBee pode ser feita utilizando o *software* X-CTU, disponibilizado gratuitamente e online pela Digi Internacional. Pode-se configurar os parâmetros do módulo para a formação de uma rede utilizando módulos XBee, como o PAN ID e o *Scan Channel*, conforme será mostrado mais adiante.

O módulo faz um varrimento na área onde se encontra para ver quais endereços de PAN e canais estão disponíveis, para então ser configurado.

Devem ser selecionados um canal de operação, que não esteja sendo utilizado, e um identificador (ID PAN) para a rede, para que possa haver comunicação entre os módulos XBee, eles devem estar configurados no mesmo canal. Então, quando iniciados, os módulos XBee podem se ligar à rede

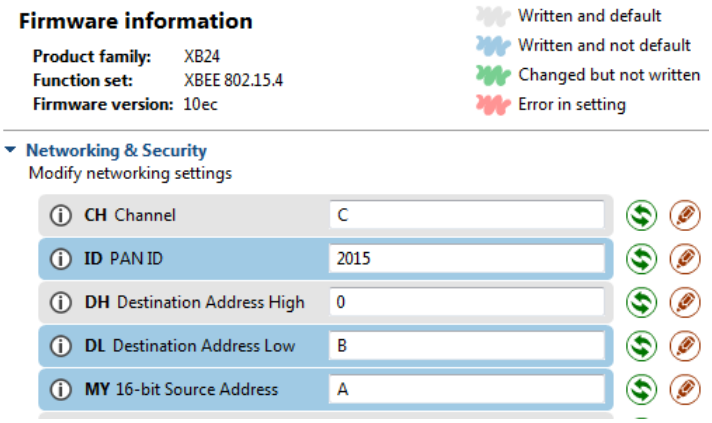
Dentre todos os campos de configuração dos módulos, dois campos foram essencialmente importantes para a comunicação entre dois módulos XBee durante o desenvolvimento do projeto, que são o DL (*Destination Address Low*) e o MY (*Source Address*). DL é o endereço de destino, é para onde os dados enviados pelo módulo irão, enquanto que MY é o endereço do próprio módulo.

Para transmitir uma mensagem para outro dispositivo é necessário configurar o endereço de destino, DL, apenas. Porém para criar uma relação recíproca, de envio e recebimento de dados, entre dois módulos, o endereço DL de um deve ser o mesmo que o endereço MY do outro e vice e versa. Assim, o destino de um módulo XBee será o endereço do outro XBee e vice e versa.

Para melhor o entendimento do desenvolvimento do projeto, o XBee conectado ao computador foi denominado XBee#1, enquanto que o XBee conectado ao robô foi denominado XBee#2.

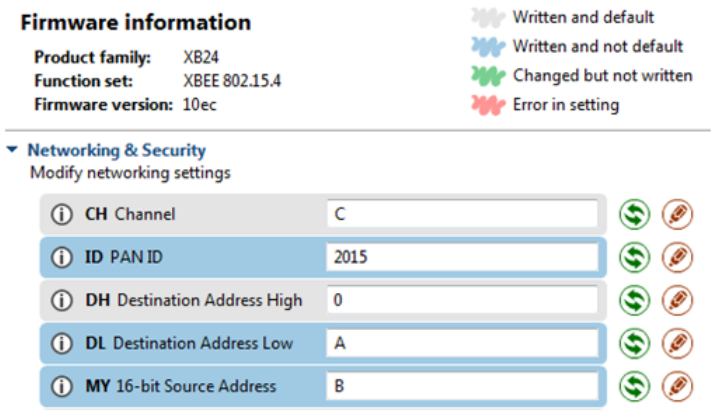
Pode-se observar nas figuras 37 e 38 as configurações realizadas para os dois módulos utilizados no projeto, por meio do *software* X-CTU.

Figura 37 – Configuração do XBee#1



Fonte: Autora

Figura 38 – Configuração do XBee#2

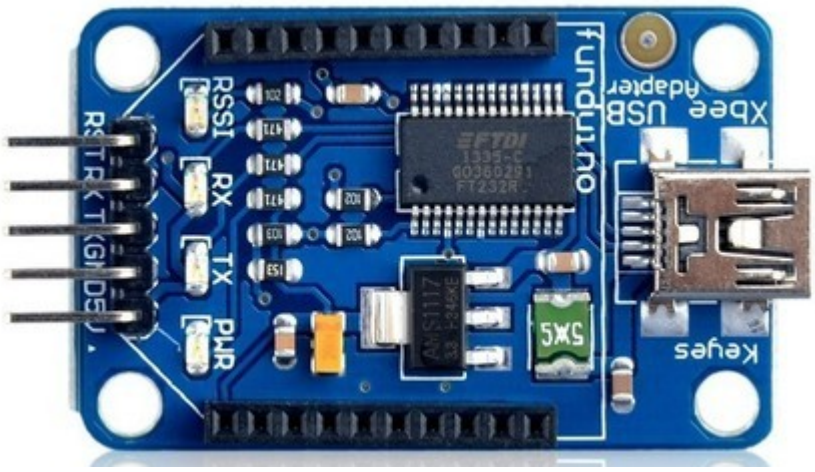


Fonte: Autora

3.3.2 Adaptador USB para o XBee

No projeto, para se realizar a conexão física entre o XBee e os outros dispositivos, computador e robô, foi necessária a utilização de um adaptador, XBee Explorer USB Adapter. Este adaptador é usado para a configuração de parâmetros do módulo XBee, facilitando a transmissão de dados sem fio e seu formato pode ser observado na figura 39.

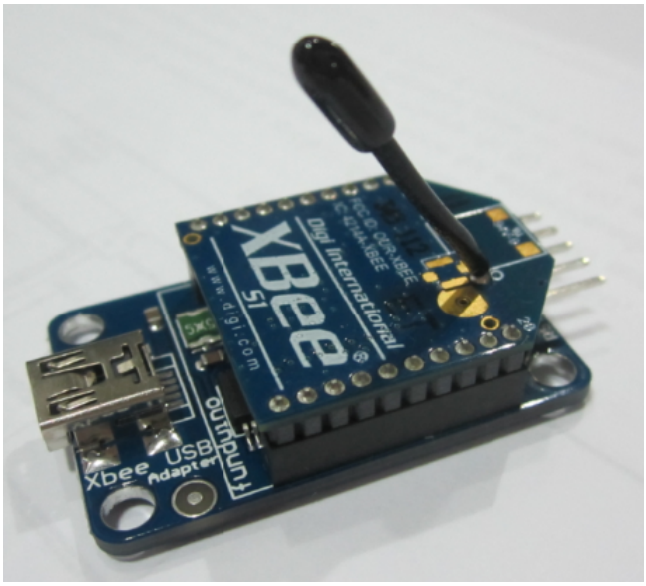
Figura 39 – XBee Explorer USB Adapter



Fonte: http://www.filipeflop.com/index.html

O módulo XBee é acoplado ao XBee Explorer USB Adapter por ligações próprias para o efeito, como pode ser observado na figura 40.

Figura 40 – XBee acoplado ao adaptador



Fonte: Autora

A conexão entre o adaptador e os demais dispositivos pode ser realizada de duas formas, a primeira, e mais simples, é a conexão do dispositivo ao adaptador por meio de um

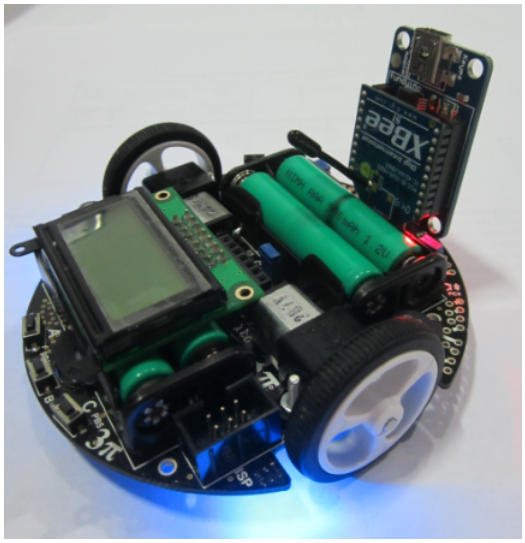
cabo USB mini, essa opção foi utilizada para realizar a conexão física entre o XBee e o computador, como pode ser observado na figura 41.

Figura 41 – XBee#1 conectado ao laptop

Fonte: Autora

A segunda forma de se conectar um dispositivo ao adaptador é utilizando a pinagem externa do mesmo. O adaptador externa os cinco principais pinos do XBee, que são: RST, RX, TX, GND e 5V, nesta ordem. Relacionando essa pinagem com a pinagem do robô Pololu, pode-se realizar a conexão física entre os dois dispositivos, conforme figura 42.

Figura 42 – XBee#2 conectado ao robô



Fonte: Autora

A tabela 8 mostra a relação entre os pinos do adaptador do XBee e os pinos do Pololu 3pi.

Tabela 8 – Conexão entre o adaptador do XBee e o robô Pololu

Adaptador do XBee	Robô Pololu
RST	-
RX	PD0 (RXD)
TX	PD1 (TXD)
GND	GND
5V	VCC

Fonte: Autora

As adaptações que tiveram que ser realizadas no *hardware* do robô para que esta conexão fosse possível podem ser verificadas na secção 3.4.1.2.

3.3.3 Comunicação entre os módulos XBee

Uma vez que os módulos XBee estão devidamente configurados, no mesmo canal e o DL do XBee#1 correspondendo ao MY do XBee#2, e vice e versa, e conectados fisicamente aos dispositivos, o sistema estará pronto para iniciar a comunicação sem fio entre os dispositivos.

A rede ZigBee é criada automaticamente pelos módulos XBee, quando os mesmos são inicializados. Quando os dispositivos são alimentados, os XBee também são e começam a funcionar, criando uma rede WPAN.

Grosso modo, os dados do computador são enviados, por meio de uma porta serial USB, ao XBee#1, onde ficam armazenados na memória temporária de entrada do módulo. O XBee#1 envia então os dados ao XBee#2, por meio da rede ZigBee criada entre eles.

Os dados recebidos pelo XBee#2 ficam em sua memória temporária de entrada até serem enviados para o robô, por comunicação serial.

3.4 Modelo

Por apresentar características essencialmente de um seguidor de linha, o Pololu 3pi representa um modelo de um AGV industrial, principalmente devido a presença de sensores de refletância, utilizados para detectar a presença de uma linha demarcada na superfície de deslocamento.

3.4.1 Pololu 3pi

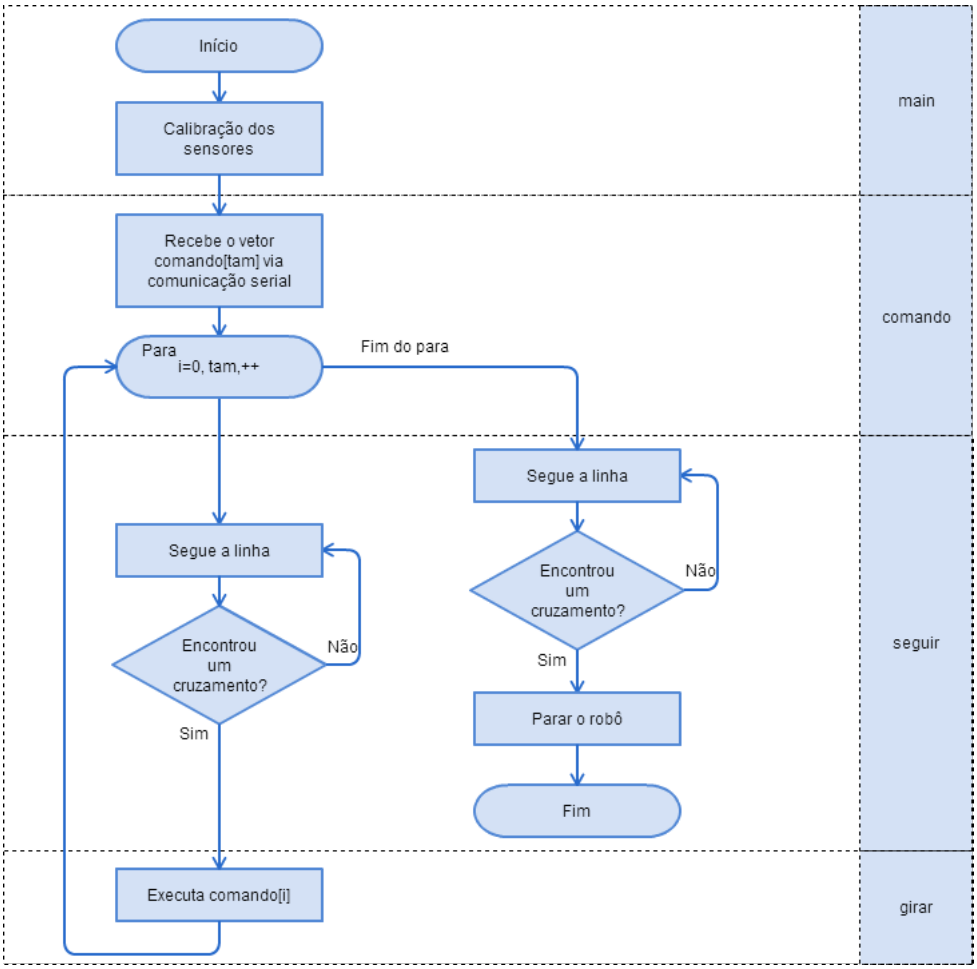
Foi desenvolvido um código de programação para o robô Pololu 3pi, para que ao receber o vetor com os comandos a serem executados, o robô siga corretamente o caminho escolhido no mapa, na pista construída, que será apresentada na secção 3.4.2.

3.4.1.1 Programação do robô Pololu 3pi

Para a programação do robô Pololu 3pi, a *Pololu Corporation* recomenda a utilização do *software* Atmel® Studio 6, gratuito, que é a plataforma de desenvolvimento integrada (IDP: *integrated development platform*) para desenvolver executar o código.

Devido à complexibilidade e extensão do código, este foi dividido em funções e arquivos. A utilização de vários arquivos facilita a visualização do código e os organiza de forma que se torna mais fácil modificá-los quando necessários.

Figura 43 – Código do Pololu 3pi



Fonte: Autora

Conforme pode ser observado na figura 43, a lógica do código foi dividida em principal, comando, seguir e girar. O fluxograma resume a lógica do código desenvolvido para o controle do robô Pololu 3pi separando os passos de acordo com a função que o executa.

O arquivo principal controla o comportamento do robô, com auxílio das funções desenvolvidas em outros arquivos. As funções são responsáveis por partes específicas do controle do robô, como por exemplo, receber informações via comunicação serial, seguir a linha, reconhecer um cruzamento, verificar o comando a ser realizado, orientar o robô na direção correta, entre outros.

A função principal inicializa o robô e faz a calibração de seus sensores. A calibração dos sensores pode levar a leituras substancialmente mais confiáveis. Durante a fase de calibração, todos os sensores devem ser expostos às leituras mais claras e mais escuras que podem ser encontradas no trajeto do robô.

Quando inicializado, o robô deve estar posicionado em cima de um segmento de linha. A primeira atividade realizada pelo robô, antes de seguir a linha e executar os comandos da rota encontrada, é girar no sentido horário e no sentido anti-horário (em seu próprio eixo), assim cada sensor pode realizar a leitura de quão escura é a linha e quão claro é o fundo do quadro.

A função comando recebe um vetor comando[tam], via comunicação serial, que contém os comandos que o robô deve executar ao encontrar cada cruzamento (vide secção 3.2.2.3).

O número de cruzamentos equivale ao número de comandos no vetor recebido (tam), pois em cada cruzamento deve ser executada uma ação de comando. Porém a função comando não identifica os cruzamentos e tão pouco executa o comando. Essa função só armazena e controla qual será o próximo comando do vetor a ser executado.

O seguimento da linha pelo robô e a detecção de cruzamentos, outras linhas, são realizados pela função seguir. A função seguir é chamada pela função comando, e faz com que o robô siga a linha até encontrar um cruzamento, quando a função seguir retorna à função anterior. Um cruzamento é detectado quando um dos sensores de extremidade detectam a presença de outra linha.

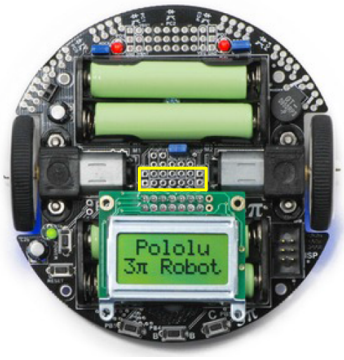
Em seguida, a função girar é chamada. Essa função é responsável por girar o robô no sentido correto (direita, esquerda ou em frente, não girando o robô), conforme o comando atribuído ao cruzamento em questão.

O código entra então em um laço de repetição, seguindo a linha (função seguir) e executando o comando (função girar), até que todos os comandos do vetor sejam realizados. Então o código entra, pela última vez, na função seguir e quando um cruzamento é detectado, o robô para e o código é finalizado, pois o robô chegou ao ponto final.

3.4.1.2 Adaptações no hardware do Pololu 3pi

O Pololu 3pi disponibiliza uma área com vários pinos de conexão para expansão opcional das funções do robô, essa área foi marcada de amarelo na figura 44.

Figura 44 – Pinos de conexão para expansão opcional



Fonte: Pololu Corporation (2014)

Entre os pinos disponíveis estão os pinos PD0 e PD1 do microcontrolador, que correspondem às funções RX (recebimento de dados) e TX (transmissão de dados), conforme pode ser observado na tabela 9.

Tabela 9 – Funções PD0 e PD1

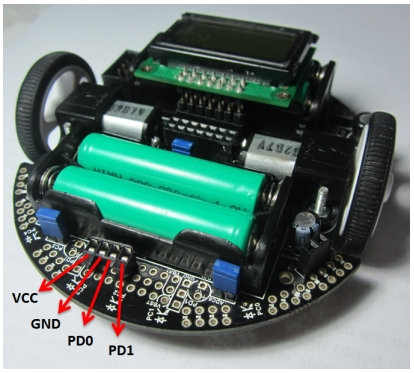
Pino ATmega328	Função
PD0	RX
PD1	TX

Fonte: Pololu Corporation (2014)

Para se realizar a conexão do adaptador do XBee ao robô Pololu, como foi mencionado na secção 3.3.2, é preciso ter acesso aos seguintes pinos do robô: PD1, PD0, GND e VCC, preferencialmente nesta ordem.

Foi então desenvolvido um modo de conexão direta do adaptador ao robô. Foi feita uma ligação dos pinos necessários a uma área livre na extremidade frontal do Pololu 3pi, conforme pode ser observado na figura 45.

Figura 45 – Adaptação do Pololu 3pi



Fonte: Autora

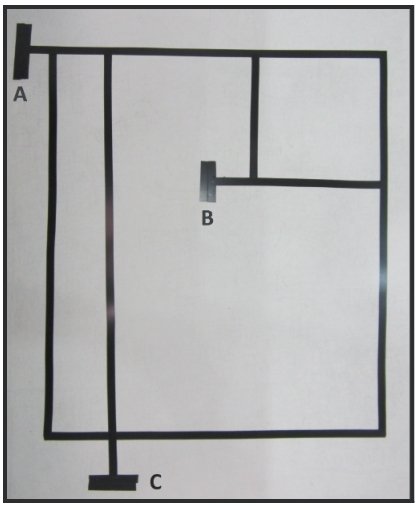
O diagrama esquemático simplificado do robô Pololu 3pi, com as ligações entre os principais componente do robô, como por exemplo, o microcontrolador, pode ser observado no anexo A.

3.4.2 Pista

A pista foi construída baseada no Mapa desenvolvido para o projeto e pode ser visualizada na figura 46. Foi utilizado um quadro de fundo branco, para representar o fundo do Mapa, e as linhas foram construídas com fita isolante preta. É importante que haja o maior contraste possível entre o fundo do quadro e a linha, pois quanto maior o contraste, maior também será o desempenho do robô em encontrar os segmentos de linha e interseções.

O robô foi programado para seguir a linha preta, então ele interpreta o fundo branco do quadro como bloqueios, ou estados não passáveis, e procura sempre pelos caminhos representados pela linha preta, que representam os estados passáveis.

Figura 46 – Pista para o robô Pololu 3pi



Fonte: Autora

4 RESULTADOS E DISCUSSÕES

Neste capítulo serão apresentados os principais resultados, obtidos por testes e simulações, a partir da metodologia abordada no trabalho.

4.1 Simulação do Algoritmo

Por meio de simulações realizadas com o algoritmo desenvolvido, foram obtidos vários resultados, podendo-se observar o comportamento do mesmo em várias situações. As situações simuladas foram as mesmas descritas na secção 3.2.2.1. Nas secções de 4.1.1 a 4.1.6 podem ser observados os Mapas virtuais, com os caminhos encontrados pelo algoritmo A*, para cada situação simulada. As simulações do algoritmo A* apresentadas a seguir foram realizadas por meio do *software* Dev-C++.

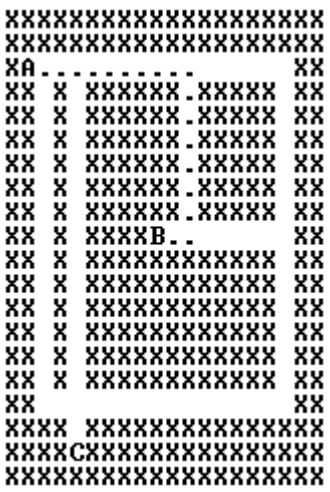
Podem ser observados também as rotas e os comandos retornados pelo algoritmo durante as simulações.

As tabelas de 1 a 6, apresentadas na secção 3.2.2.1, foram organizadas em ordem crescente de valor do custo de movimentação, g, e serão apresentadas em cada situação, a fim de se realizar a comparação entre o caminho encontrado pelo algoritmo com os caminhos possíveis estudados anteriormente, para cada situação.

4.1.1 Situação 1: de A para B

Na simulação desta situação foi obtido como resposta o Mapa virtual apresentado na figura 47.

Figura 47 – Caminho encontrado pelo algoritmo para ir de A para B



Fonte: Autora

O algoritmo retornou também a seguinte rota: $[0, 0, 1, 2]$. A rota foi transformada em comandos pelo próprio algoritmo, que retornou ainda os seguintes comandos: $[F, F, D, D]$.

A tabela 10 apresenta seis possíveis caminhos que podem ser escolhidos para se mover de A até B, os caminhos estão ordenados pelo custo de movimentação, do mais curto ao mais longo.

Tabela 10 – Caminhos de A para B, ordenados por g

Caminho	Percurso	Custo de movimentação (g)
1	$[A, 1, 2, 3, 5, B]$	19
2	$[A, 1, 2, 3, 4, 5, B]$	31
3	$[A, 1, 2, 6, 4, 5, B]$	45
6	$[A, 1, 6, 4, 5, B]$	45
4	$[A, 1, 6, 2, 3, 5, B]$	47
5	$[A, 1, 6, 4, 3, 5, B]$	59

Fonte: Autora

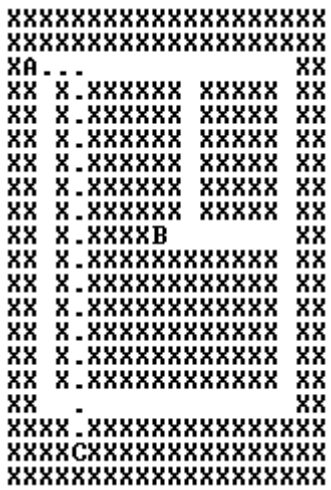
Comparando o caminho encontrado pelo algoritmo A*, com os percursos apresentados na tabela 10, pode-se dizer que o caminho encontrado pelo algoritmo corresponde ao caminho 1, cujo custo de movimentação , g, é 19.

Lembrando-se que os nós são referentes às interseções do Mapa, onde mais de um caminho pode ser seguido, e observando-se que os pontos A, B e C não são considerados nós. Então, quando se observa o número de nós percorridos pelo caminho 1 e a quantidade de comandos que devem ser executados pelo robô para percorrer este caminho no mapa, pode-se dizer que o número de nós é equivalente à quantidade de comandos.

4.1.2 Situação 2: de A para C

Na simulação desta situação foi obtido como resposta o Mapa virtual apresentado na figura48.

Figura 48 – Caminho encontrado pelo algoritmo para ir de A para C



Fonte: Autora

O algoritmo retornou também a seguinte rota: $[0, 1, 1]$, que foi transformada nos seguintes comandos: $[F, D, F]$.

A tabela 11 apresenta quatro possíveis caminhos que podem ser escolhidos para se mover de A até C, os caminhos estão ordenados pelo custo de movimentação, do mais curto ao mais longo.

Tabela 11 – Caminhos de A para C, ordenados por g

Caminho	Percurso	Custo de movimentação (g)
3	$[A, 1, 2, 6, C]$	19
4	$[A, 1, 6, C]$	19
1	$[A, 1, 2, 3, 5, 4, 6, C]$	45
2	$[A, 1, 2, 3, 4, 6, C]$	45

Fonte: Autora

Comparando o caminho encontrado pelo algoritmo A*, com os percursos apresentados na tabela 11, pode-se dizer que o caminho encontrado pelo algoritmo corresponde ao caminho 3, cujo custo de movimentação, g, é 19.

Analisando os custos de movimentação, g, dos caminhos possíveis entre A e C, apresentados na tabela, pode-se observar que os caminhos 3 e 4, apesar de apresentarem dois percursos diferentes, possuem igual custo de movimentação, $g = 19$, e são os caminhos mais curtos para se percorrer do ponto A ao C.

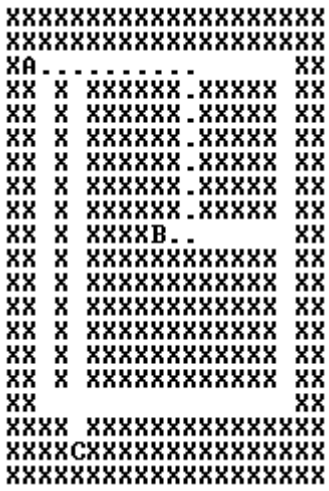
Pode-se observar um acontecimento similar nos caminhos 1 e 2, que também possuem igual custo de movimentação, $g = 45$, apesar de apresentarem dois percursos diferentes. Porém esses caminhos são os mais longos para se percorrer do ponto A ao C.

Ao se observar o número de nós percorridos pelos caminhos 3 e 4, que são 3 nós e 2 nós, respectivamente, e observando também a quantidade de comandos que devem ser executados pelo robô para percorrer o caminho encontrado pelo algoritmo, 3 comandos, pode-se dizer que o número de nós do caminho 3 é equivalente à quantidade de comandos que o algoritmo utiliza.

4.1.3 Situação 3: de B para A

Na simulação desta situação foi obtido como resposta o Mapa virtual apresentado na figura49.

Figura 49 – Caminho encontrado pelo algoritmo para ir de B para A



Fonte: Autora

O algoritmo retornou também a seguinte rota: $[3, 2, 2, 2]$, que foi transformada nos seguintes comandos: $[E, E, F, F]$.

A tabela 12 apresenta seis possíveis caminhos que podem ser escolhidos para se mover de B até A, os caminhos estão ordenados pelo custo de movimentação, do mais curto ao mais longo.

Tabela 12 – Caminhos de B para A, ordenados por g

Caminho	Percurso	Custo de movimentação (g)
3	$[B, 5, 3, 2, 1, A]$	19
4	$[B, 5, 4, 3, 2, 1, A]$	31
5	$[B, 5, 4, 6, 2, 1, A]$	45
6	$[B, 5, 4, 6, 1, A]$	45
2	$[B, 5, 3, 2, 6, 1, A]$	47
1	$[B, 5, 3, 4, 6, 1, A]$	59

Fonte: Autora

Comparando o caminho encontrado pelo algoritmo A*, com os percursos apresentados na tabela 12, pode-se dizer que o caminho encontrado pelo algoritmo corresponde ao caminho 3, cujo custo de movimentação , g, é 19.

Ao se observar o número de nós percorridos pelo caminho 3 e a quantidade de comandos que devem ser executados pelo robô para percorrer este caminho no mapa, pode-se dizer que o número de nós é equivalente à quantidade de comandos.

Pode-se observar ainda que os caminhos encontrados pelo algoritmo A* para se mover do ponto B ao A e do ponto A ao B são equivalentes, pois percorrem os mesmos nós, 1, 2, 3 e 5, porém em ordem inversa. A tabela 13 mostra a comparação entre as duas situações.

Outra coisa que pode ser observada é que apesar de os percursos encontrados para se mover de A para B e de B para A serem um o inverso do outro, os comandos e as rotas

Tabela 13 – Comparação entre os caminhos encontrados de A para B e de B para A

	De A para B	De B para A
Rota	$[0, 0, 1, 2]$	$[3, 2, 2, 2]$
Comandos	$[F, F, D, D]$	$[E, E, F, F]$
Percurso	$[A, 1, 2, 3, 5, B]$	$[B, 5, 3, 2, 1, A]$
Custo g	19	19

Fonte: Autora

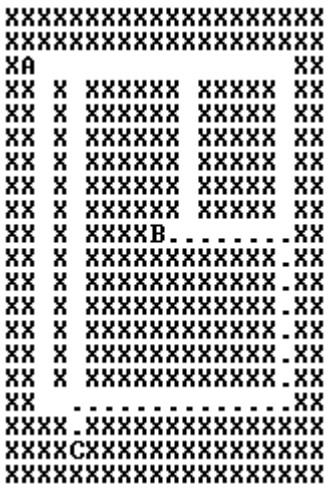
retornadas para ir de A a B e para ir de B a A não são simplesmente um o inverso do outro, como pode ser observado na tabela 13.

Para os outros caminhos encontrados de B para A, pode-se observar a mesma ocorrência de inversão de percurso, em relação aos caminhos encontrados de A para B. Comparando-se os percursos dos caminhos, com o auxílio das tabelas 12 e 10, pode-se dizer que os percursos dos caminhos 2, 3, 6, 4 e 5, de A para B são inversos aos percursos dos caminhos 4, 5, 6, 2 e 1, de B para A, respectivamente.

4.1.4 Situação 4: de B para C

Na simulação desta situação foi obtido como resposta o Mapa virtual apresentado na figura50.

Figura 50 – Caminho encontrado pelo algoritmo para ir de B para C



Fonte: Autora

O algoritmo retornou também a seguinte rota: $[0, 1, 1]$, que foi transformada nos seguintes comandos: $[F, D, E]$.

A tabela 14 apresenta seis possíveis caminhos que podem ser escolhidos para se mover de B até C, os caminhos estão ordenados pelo custo de movimentação, do mais curto ao mais longo.

Tabela 14 – Caminhos de B para C, ordenados por g

Caminho	Percurso	Custo de movimentação (g)
6	$[B, 5, 4, 6, C]$	30
2	$[B, 5, 3, 2, 6, C]$	32
3	$[B, 5, 3, 2, 1, 6, C]$	36
1	$[B, 5, 3, 4, 6, C]$	44
4	$[B, 5, 4, 3, 2, 6, C]$	44
5	$[B, 5, 4, 3, 2, 1, 6, C]$	48

Fonte: Autora

Comparando o caminho encontrado pelo algoritmo A*, com os percursos apresentados na tabela14, pode-se dizer que o caminho encontrado pelo algoritmo corresponde ao caminho 6, cujo custo de movimentação, g, é 30.

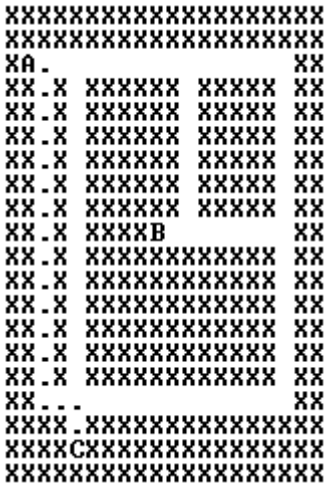
Analisando os custos de movimentação, g, dos caminhos possíveis entre B e C, apresentados na tabela, pode-se observar que os caminhos 1 e 4, apesar de apresentarem dois percursos diferentes, possuem igual custo de movimentação, $g = 44$, porém não representam o caminho mais curto para se percorrer do ponto B ao C.

Ao se observar o número de nós percorridos pelo caminho 6 e a quantidade de comandos que devem ser executados pelo robô para percorrer este caminho no mapa, pode-se dizer que o número de nós é equivalente à quantidade de comandos.

4.1.5 Situação 5: de C para A

Na simulação desta situação foi obtido como resposta o Mapa virtual apresentado na figura 51.

Figura 51 – Caminho encontrado pelo algoritmo para ir de C para A



Fonte: Autora

O algoritmo retornou também a seguinte rota: $[2, 2]$, que foi transformada nos seguintes comandos: $[E, E]$.

A tabela 15 apresenta quatro possíveis caminhos que podem ser escolhidos para se mover de C até A, os caminhos estão ordenados pelo custo de movimentação, do mais curto ao mais longo.

Tabela 15 – Caminhos de C para A, ordenados por g

Caminho	Percurso	Custo de movimentação (g)
1	$[C, 6, 1, A]$	19
2	$[C, 6, 2, 1, A]$	19
3	$[C, 6, 4, 5, 3, 2, 1, A]$	45
4	$[C, 6, 4, 3, 2, 1, A]$	45

Fonte: Autora

Comparando o caminho encontrado pelo algoritmo A*, com os percursos apresentados na tabela 15, pode-se dizer que o caminho encontrado pelo algoritmo corresponde ao caminho 1, cujo custo de movimentação , g, é 19.

Analisando os custos de movimentação, g, dos caminhos possíveis entre C e A, apresentados na tabela, pode-se observar que os caminhos 1 e 2, apesar de apresentarem dois percursos diferentes, possuem igual custo de movimentação, $g = 19$, e são os caminhos mais curtos para se percorrer do ponto C ao A.

Ao se observar o número de nós percorridos pelos caminhos 1 e 2, que são 2 nós e 3 nós, respectivamente, e observando também a quantidade de comandos que devem ser executados pelo robô para percorrer o caminho encontrado pelo algoritmo A*, 2 comandos, pode-se dizer que o número de nós do caminho 1 é equivalente à quantidade de comandos que o algoritmo utiliza.

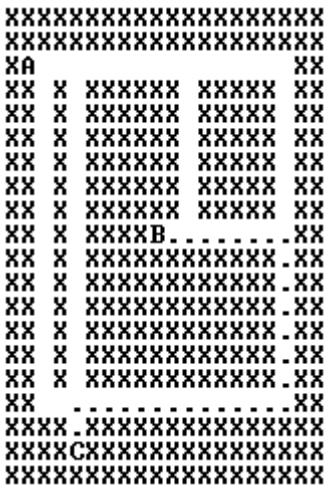
Pode-se observar ainda que o caminho encontrado pelo algoritmo para se mover do ponto C até o ponto A não é equivalente ao caminho encontrado pelo algoritmo para se mover do ponto A ao ponto C, embora os dois possuam o mesmo custo de movimentação.

Para os outros caminhos encontrados de C para A, pode-se observar uma ocorrência de inversão de percurso, em relação aos caminhos encontrados de A para C. Comparando-se os percursos dos caminhos, com o auxílio das tabelas 15 e 11, pode-se dizer que os percursos dos caminhos 4, 3, 1 e 2, de A para C são inversos aos percursos dos caminhos 1, 2, 3, e 4, de C para A, respectivamente.

4.1.6 Situação 6: de C para B

Na simulação desta situação foi obtido como resposta o Mapa virtual apresentado na figura52.

Figura 52 – Caminho encontrado pelo algoritmo para ir de C para B



Fonte: Autora

O algoritmo retornou também a seguinte rota: $[0, 2, 2]$, que foi transformada nos seguintes comandos: $[D, E, F]$.

A tabela 16 apresenta seis possíveis caminhos que podem ser escolhidos para se mover de C até B, os caminhos estão ordenados pelo custo de movimentação, do mais curto ao mais longo.

Tabela 16 – Caminhos de C para B, ordenados por g

Caminho	Percurso	Custo de movimentação (g)
5	$[C, 6, 4, 5, B]$	30
3	$[C, 6, 2, 3, 5, B]$	32
1	$[C, 6, 1, 2, 3, 5, B]$	36
4	$[C, 6, 2, 3, 4, 5, B]$	44
6	$[C, 6, 4, 3, 5, B]$	44
2	$[C, 6, 1, 2, 3, 4, 5, B]$	48

Fonte: Autora

Comparando o caminho encontrado pelo algoritmo A*, com os percursos apresentados na tabela 16, pode-se dizer que o caminho encontrado pelo algoritmo corresponde ao caminho 5, cujo custo de movimentação, g, é 30.

Analisando os custos de movimentação, g, dos caminhos possíveis entre C e B, apresentados na tabela, pode-se observar que o caminho mais curto é também o caminho 5. E que os caminhos 4 e 6, apesar de apresentarem dois percursos diferentes, possuem igual custo de movimentação, $g = 44$.

Ao se observar o número de nós percorridos pelo caminho 5 e a quantidade de comandos que devem ser executados pelo robô para percorrer este caminho no mapa, pode-se dizer que o número de nós é equivalente à quantidade de comandos.

Tabela 17 – Comparação entre os caminhos encontrados de B para C e de C para B

	De B para C	De C para B
Rota	$[0, 1, 1]$	$[0, 2, 2]$
Comandos	$[F, D, E]$	$[D, E, F]$.
Percurso	$[B, 5, 4, 6, C]$	$[C, 6, 4, 5, B]$
Custo g	30	30

Fonte: Autora

Pode-se observar ainda que os caminhos encontrados pelo algoritmo A* para se mover do ponto C ao B e do ponto B ao C são equivalentes, pois percorrem os mesmos nós, 6, 4 e 5, porém em ordem inversa. A tabela 17 mostra a comparação entre as duas situações.

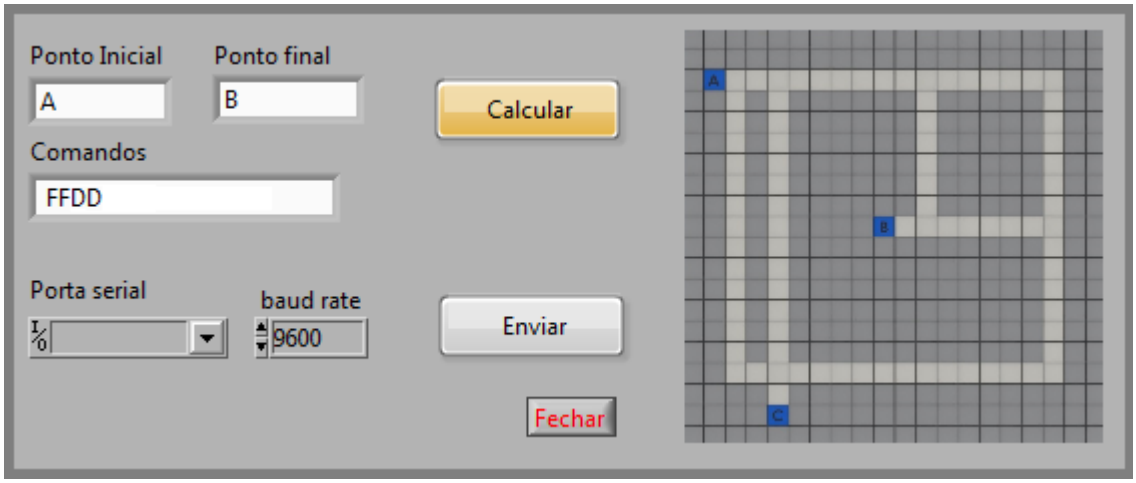
Outra coisa que pode ser observada é que apesar de os percursos encontrados para se mover de B para C e de C para B serem um o inverso do outro, os comandos e as rotas retornadas para ir de B a C e para ir de C a B não são simplesmente um o inverso do outro, como pode ser observado na tabela 17.

Para os outros caminhos encontrados de C para B, pode-se observar a mesma ocorrência de inversão de percurso, em relação aos caminhos encontrados de B para C. Comparando-se os percursos dos caminhos, com o auxílio das tabelas 16 e 14, pode-se dizer que os percursos dos caminhos 2, 3, 4, 1 e 5, de B para C são inversos aos percursos dos caminhos 3, 1, 4, 6 e 2 de C para B, respectivamente.

4.2 Interface

Foram realizados vários testes da interface de comunicação, para várias situações diferentes. A figura 53 apresenta a tela de interface durante um dos testes realizados.

Figura 53 – Teste da interface de comunicação



Fonte: Autora

Pode-se relacionar o teste apresentado à situação 1, onde A é indicado como ponto inicial e B como ponto final. Observando-se os comandos apresentados pela tela de interface e os

comandos retornados pelo algoritmo A*, durante a simulação da situação 1, apresentado na secção 4.1.1, pode ser verificada a equivalência entre os comandos.

4.3 Robô

Após a realização da simulação do algoritmo desenvolvido e da interface de comunicação, foram realizados testes com o robô percorrendo a pista construída.

As seis situações simuladas pelo algoritmo na secção 4.1, foram também testadas no robô.

Foi observado que após o acionamento do envio de dados do computador ao robô, este recebeu os comandos quase instantaneamente.

Uma vez recebidos os dados, verificou-se que o robô aguarda o acionamento de seu botão B, para então começar a seguir a linha e executar os comandos recebidos.

Durante um teste da situação 1, de A para B, por exemplo, verificou-se que o robô executou os comandos $[F, F, D, D]$, passando pelos nós 1, 2, 3 e 5, respectivamente, antes de chegar ao ponto final, B.

5 CONCLUSÕES

Como pode ser observado, a utilização de métodos heurístico é bem válido na redução de custos quando se tratando de movimentação de materiais.

Pode-se dizer que o ambiente definido para a realização do projeto é um sistema complexo de roteirização de materiais, uma vez que para se locomover de um ponto a outro do ambiente, existem mais de um caminho possível.

O fato de o ambiente ser um sistema complexo possibilitou a implementação do algoritmo de busca A*, desenvolvido para decidir qual o caminho mais curto a ser seguido entre dois pontos. Analisando os resultados das simulações do algoritmo e os custos de movimentação apresentados nos resultados do trabalho, pode-se afirmar que o algoritmo A* desenvolvido conseguiu encontrar sempre o percurso ótimo entre dois pontos. Por percurso ótimo entende-se o percurso mais curto, com menor custo de movimentação.

Tendo em mente que o caminho entre dois pontos segue uma sequência de nós que é chamada percurso, verificou-se então que para se movimentar de um ponto a um destino, e então para se voltar ao ponto inicial, utilizando-se um caminho ótimo, encontrado pelo algoritmo A*, o caminho que se percorre na ida é igual ao caminho que se percorre na volta, porem em sentido contrario, claro. Ou seja, o menor caminho a se percorrer para se ir de A a B é também o menor caminho a se percorrer para se voltar de B a A, os dois caminhos são os mesmos, passando pelos mesmos nós e tem o mesmo custo de movimentação, a única diferença é que o percurso de um é o inverso do percurso do outro.

Porém em situações em que não exista somente um caminho mais curto, ou seja existem dois caminhos ‘menores’, ambos com o mesmo custo de movimentação, porém não necessariamente com os mesmos nós em seus percursos, então qualquer que seja o caminho escolhido, dentre esses dois menores, será considerada uma solução ótima.

A lógica do algoritmo A* irá retornar como resposta, caminho ótimo encontrado, o caminho que passar pelo nó que foi estudado e expandido primeiro. Isso explica porque o caminho encontrado pelo algoritmo para ir de A a C é diferente do caminho encontrado pelo algoritmo para ir de C a A. Mas deve-se ressaltar que os dois caminhos possuem o mesmo custo de movimentação, e ambos são soluções ótimas para o deslocamento de A a C.

Observando-se a equivalência entre o número de nós existentes no percurso do caminho encontrado pelo algoritmo A*, com o número de comandos a serem executados para se percorrer o mesmo caminho, pode-se inferir que a cada nó, o robô deverá executar um comando.

Considerando-se os resultados obtidos no projeto, pode-se afirmar que o método heurístico implementado no projeto, método A Estrela, atendeu corretamente as necessidades do projeto, visto que conseguiu encontrar sempre o menor percurso entre dois pontos.

Analisando-se os testes executados no robô pode-se alegar que o Pololu 3pi se comportou de forma esperada, executando os comandos corretos, nas interseções corretas e chegando ao ponto de destino.

Por fim, pode-se dizer que os métodos utilizados no desenvolvimento deste trabalho podem ser muito úteis na aplicação em veículos automaticamente guiados, para utilização em ambientes fabris, na movimentação de materiais. Da mesma forma que a aplicação do sistema funcionou neste modelo, pode também ser implementado em AGV e funcionar da mesma forma em um ambiente de maior escala.

5.1 Trabalhos Futuros

- Desenvolvimento de um sistema de localização do robô por meio do método de triangularização.
- Aplicação de uma heurística de custo variável conforme o terreno do mapa, ou conforme o tráfego de veículos nos caminhos.
- Aplicação do sistema desenvolvido em um ambiente real, com veículos automaticamente guiados em sistemas de movimentação de materiais em uma indústria.
- Desenvolvimento de sistemas de segurança para detecção de obstáculos na pista.

Referências

BAUDIN, M. *Lean logistics: the nuts and bolts of delivering materials and goods*. [S.l.]: Productivity Press, 2004. 14

BENEVIDES, P. F. *Aplicação de heurísticas e metaheurísticas para o problema do caixeiro viajante em um problema real de roterização de veículos*. Dissertação (Mestrado), 2012. 18

BOZER, Y. A.; YEN, C.-k. Intelligent dispatching rules for trip-based material handling systems. *Journal of Manufacturing Systems*, Elsevier, v. 15, n. 4, p. 226–239, 1996. 14

BRAUNL, T. *Embedded Robotics*. [S.l.]: Springer, 2006. 11

BUENO, F. Métodos heurísticos-teoria e implementações. *IFSC. Araranguá*, 2009. 19

CAZENAVE, T. Optimizations of data structures, heuristics and algorithms for path-finding on maps. p. 27–33, 2006. 19

COPPIN, B. *Inteligência Artificial*. [S.l.]: LTC - Livros Técnicos e Científicos Editora LTDA, 2012. 20

FERNANDES, A. S. L. *WirelessTeams: Comparação de Tecnologias Sem Fios em Equipas de Robôs Móveis: Estado da Arte*. [S.l.]: University of Coimbra, 2011. 25

FERNANDES, A. S. L. *Comunicação Ad Hoc em Equipas de Robôs Móveis Utilizando a Tecnologia ZigBee*. Dissertação (Mestrado) — University of Coimbra, 2012. 24

FIELD, B. F.; KASPER, J. G. *Automated guided vehicle*. [S.l.]: Google Patents, 1989. US Patent 4,846,297. 16

FRAGA, M. *Uma metodologia hıbrida: Colônia de formigas–Busca Tabu–Reconexao por caminhos para resoluçao do problema de roteamento de veıculos com janela de tempo*. Dissertação (Mestrado) — Dissertaçao de Mestrado, CEFET-MG, Brasil, 2006. 18

GROOVER, M. P. *Automação industrial e sistemas de manufatura*. 3. ed. São Paulo: Pearson Prentice Hall, 2011. 16

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: U Michigan Press, 1975. 18

LANGER, R. A. *Estudo e Implementação de Métodos para Planejamento de Trajetórias e Controle de Robôs Móveis Não Holonômicos*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Paraná, 2007. 12

LUGER, G. F. *Inteligência Artificial*. 6. ed. [S.l.]: Pearson Education do Brasil Ltda., 2013. 19, 23

MILLER, M. *Discovering Bluetooth*. [S.l.]: Sybex, 2001. 24

MORAIS, J. V. M. Marcia de F. Métodos heurísticos construtivos para redução do estoque em processo em ambientes de produção flow shop híbridos com tempos de setup dependentes da sequência. *Scielo*, v. 17, p. 367–375, 2010. 11

POLOLU CORPORATION. *Pololu 3pi Robot Users Guide*. <http://www.pololu.com/docs/0J21/all>, 2014. 26, 55

POTHUGANTI, K.; CHITNENI, A. *A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi*. [S.l.]: Research India Publications, ISSN, 2014. 47

PRESTES, Á. N. *Uma análise experimental de abordagens heurísticas aplicadas ao problema do caixeiro viajante*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2006. 18

RAINWATER, S. Pololu 3pi: The 10,000 mile review. 2009. 26

REEVES, C. R. *Modern heuristic techniques for combinatorial problems*. [S.l.]: John Wiley & Sons, Inc., 1993. 11

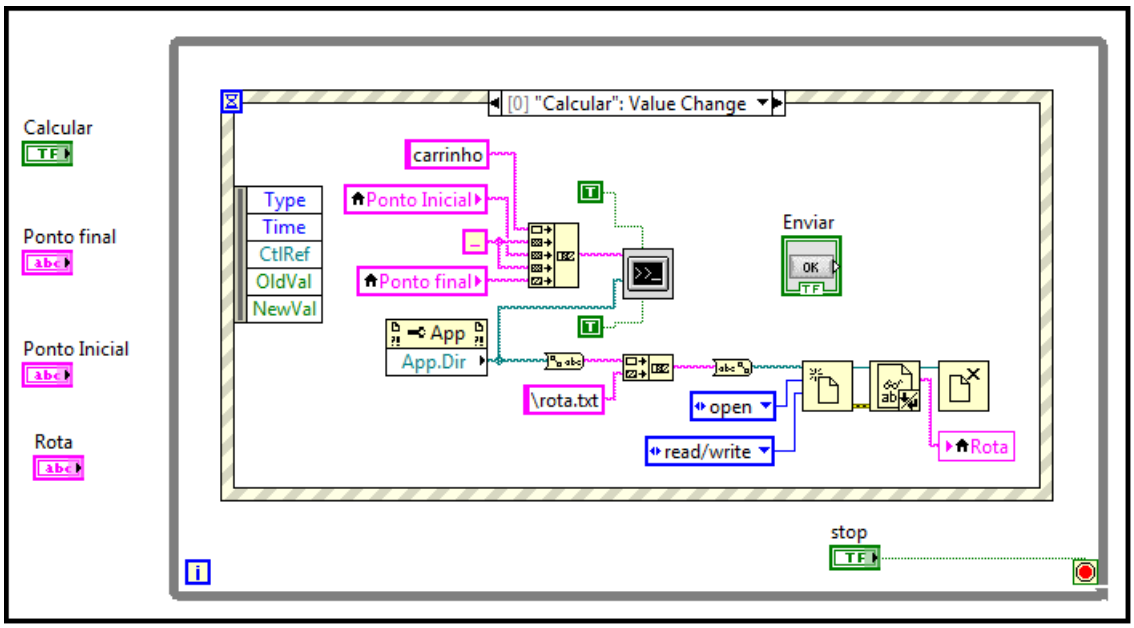
ŠAFARIĆ, S.; MALARIĆ, K. Zigbee wireless standard. In: *Proceedings of 48th International Symposium ELMAR*. [S.l.: s.n.], 2006. p. 259–262. 25

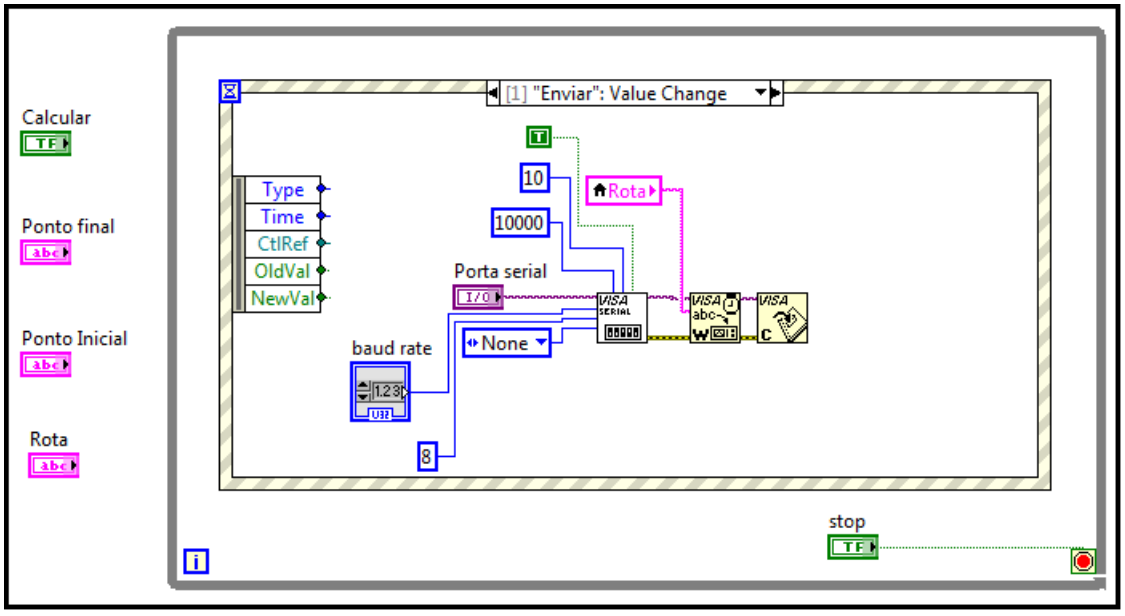
SEZEN, B. Modeling automated guided vehicle systems in material handling. Doğuş Üniversitesi, 2003. 15

SILVA, G. A. N. d. et al. Algoritmos heurísticos construtivos aplicados ao problema do caixeiro viajante para a definição de rotas otimizadas. In: *Colloquium Exactarum*. [S.l.: s.n.], 2013. v. 5, n. 2, p. 30–46. 17

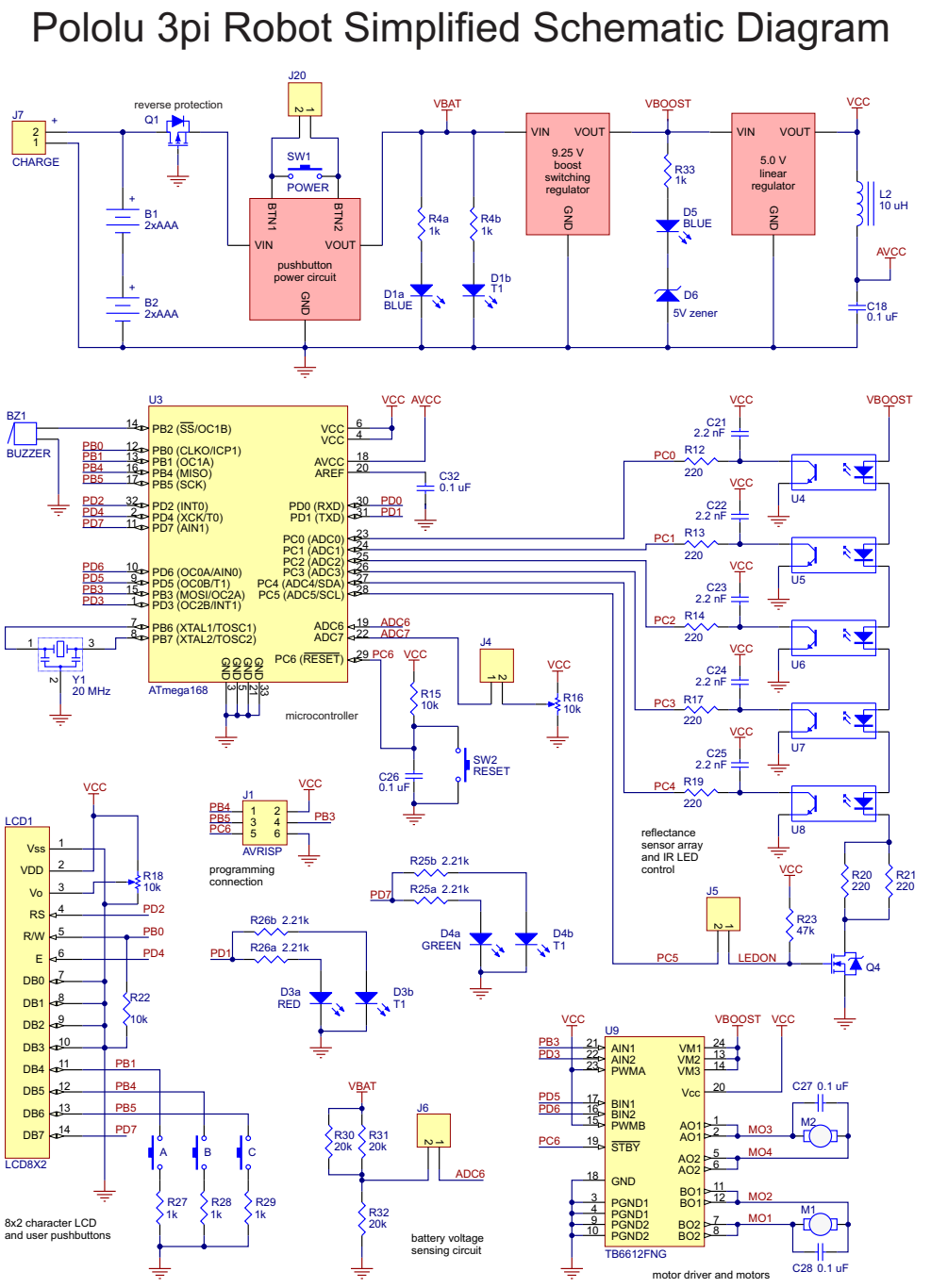
SULE, D. R. D. R. Book; Book/Illustrated. *Manufacturing facilities : location, planning, and design*. 3rd ed. ed. [S.l.]: Boca Raton : CRC Press, 2009. Previous ed.: Boston: PWS Pub. Co., 1994. ISBN 9781420044225. 11, 14

APÊNDICE A – Programação gráfica da interface de comunicação





ANEXO A – Diagrama esquemático simplificado do robô Pololu 3pi



© 2008 Pololu Corporation
http://www.pololu.com/