UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

EDUARDO REBELO

UTILIZAÇÃO DE ALGORITMO DE DIJKSTRA PARA PLANEJAMENTO DE TRAJETÓRIA DE UM PROTÓTIPO DE ROBÔ AUTÔNOMO DESENVOLVIDO EM ARDUINO

CRICIÚMA

2017

EDUARDO REBELO

UTILIZAÇÃO DE ALGORITMO DE DIJKSTRA PARA PLANEJAMENTO DE TRAJETÓRIA DE UM PROTÓTIPO DE ROBÔ AUTÔNOMO DESENVOLVIDO EM ARDUINO

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC

Orientador: Prof. Esp. Sérgio Coral

Coorientador: Dr. Kristian Madeira

CRICIÚMA

2017

EDUARDO REBELO

UTILIZAÇÃO DE ALGORITMO DE DIJKSTRA PARA PLANEJAMENTO DE TRAJETÓRIA DE UM PROTÓTIPO DE ROBÔ AUTÔNOMO DESENVOLVIDO EM ARDUINO

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Automação

Criciúma, 19 de junho de 2017

BANCA EXAMINADORA

Prof. Sérgio Coral - Especialista - (UNESC) - Orientador

Prof. Kristian Madeira - Doutor - (UNESC) - Coorientador

Prof. Christine Vieira - Mestre - (UNESC)

Prof. Paulo João Martins - Mestre - (UNESC)

Dedico este trabalho a todos que de alguma forma contribuíram para a conclusão desta etapa de minha caminhada.

AGRADECIMENTOS

Agradeço aos meus pais por me apoiarem sempre e nunca terem mediram esforços para me em caminhada acadêmica.

Agradeço a minha namorada, Jessica pelo carinho, paciência e compreensão além de me apoiar em toda minha jornada acadêmica.

Agradeço ao meu orientador e coorientador por me apresentarem o conhecimento necessário para o desenvolvimento de todas as etapas do meu projeto.

Por fim, agradeço a todos colegas e professores do curso de Ciência da Computação.

Obrigado a todos vocês!

.

“Existem muitas hipóteses em ciência que estão erradas. Isso é perfeitamente aceitável, eles são a abertura para achar as que estão certas”

Carl Sagan

RESUMO

Neste trabalho é apresentado a utilização de um software, que através da utilização de algoritmo de Dijkstra determina a trajetória de custo mínimo, partindo de um vértice inicial até um vértice de destino. Essas rotas são enviadas ao robô autônomo através de comunicação Bluetooth, com a qual é possível realizar, a movimentação do mesmo através de uma ilustração do grafo utilizado. O robô tem o objetivo de realizar o deslocamento através de linhas pretas na qual representam as arestas do grafo e são detectadas através de sensores de linha. O processamento do algoritmo defini a proposta do trabalho, na qual tem o objetivo de determinar o trajeto de menor caminho entre um ponto de origem e de destino realizando o deslocamento do robô através do caminho gerado. O trabalho também utiliza de técnicas de estruturas de dados, na qual foi aplicado o conceito de Árvore B para armazenamento de informações que determinam as direções do carrinho seguidor de linha. A solução encontrara para realizar a comunicação Bluetooth foi utilizar a biblioteca SerialClass.h, sendo ela utilizada devido a sua compatibilidade de realizar a comunicação através das portas seriais. O trabalho proposto tem a o objetivo de atuar como um estímulo de aprendizado de algoritmos de grafos através da utilização da robótica.

Palavras-chave: Arduino. Protótipo Autônomo. Grafos. Robótica móvel. Algoritmo de Dijkstra. Bluetooth.

ABSTRACT

In this paper is presented the use of a software, which, through the use of Dijkstra algorithm, determines the minimum cost trajectory, starting from an initial vertex to a destination vertex. These routes are sent to the autonomous robot through Bluetooth communication, with which it is possible to carry out the movement of the same through an illustration of the graph used. The robot has the objective to carry out the displacement through the black lines in which they represent the edges of the graph and are detected through line sensors. The algorithm processing define the work proposal, which aims to determine the trajectory of the smallest path between a point of origin and destination by performing the displacement of the robot through the created path. The work also uses techniques of data structures, in which the concept of Tree B was applied to store information that determine the directions of the row follower car. The solution found to perform the Bluetooth communication was to use the SerialClass.h library, being used because of its compatibility of performing the communication through the serial doors. The proposed work has the objective of acting as a stimulus for algorithms learning of graphs through the use of robotics.

Palavras-chave: Arduino. Autonomous prototype. Graphs. Mobile robotics. Dijkstra Algorithm. Bluetooth.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GPS	Global Positioning System
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
RIA	Robotics Industries Association
RPL	Robot Programming Languages
SPSS	Statistical Package for the Social Sciencies
TIC	Tecnologias de Informação e Comunicação

SUMÁRIO

1 INTRODUÇÃO

Uma linha de pesquisa, visando melhorar o processo de estudo é direcionada a utilização de ferramentas e software computacionais como ambiente de estudo. A partir de observações em aulas de graduação, percebe-se um melhor resultado de aprendizado por meio de atividades práticas de desenvolvimento utilizando ferramentas e simuladores didáticos de representação de conceitos abstratos (SANTOS; RODRIGUES; HEITOR, 2005).

Existem problemas computacionais interessantes em termos de grafos. Entretanto, parte dos alunos enfrentam dificuldades na implementação de algoritmos em grafos, devido a necessidade de suporte para implementação. Implementações essas que são compostas por estruturas de dados, ferramentas de visualização e iteração com grafos, diferença de níveis de abstração entre as definições teóricas dos algoritmos e representações computacionais necessárias para implementá-los (SOARES, 2004)

Um deles é o problema de menor caminho, ele é utilizado em muitas aplicações, como por exemplo, em telecomunicações, transporte, correios, entre outros, na qual seja necessário encontrar um caminho de menor custo, ou mais rápido, entre dois pontos. Um dos algoritmos que solucionam esse problema, é o algoritmo de Dijkstra. Este algoritmo funciona apenas em grafos que os pesos das arestas não forem negativos. Ele é aplicado em grafos simples que tem o propósito de determinar o caminho de menor custo entre dois vértices. Quando o grafo não é simples, então o algoritmo procura transformá-lo em um grafo simples eliminando seus laços e eventuais arestas múltiplas deixando apenas com o peso menor (SILVA; FERNANDES; SANCHES ,2009).

O presente trabalho tem por objetivo mostrar ao aluno a utilização do algoritmo de Dijkstra como forma de determinar a rota de caminho mínimo, em que através das rotas geradas, realize a movimentação do robô autônomo a partir de um ponto de origem até um ponto de destino.

1.1 OBJETIVO GERAL

Este trabalho tem por objetivo desenvolver um software que utiliza algoritmo de Dijkstra para determinar o planejamento de trajetória de um robô autônomo desenvolvido em Arduino.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos desta pesquisa são:

a) Compreender teoria de grafos;
b) Investigar conceitos de desenvolvimento de robótica em Arduino;
c) prototipar um robô autônomo;
d) prototipar um software que utiliza algoritmo de Dijkstra para determinar uma trajetória de menor caminho;
e) elaborar a comunicação Bluetooth entre o software e Arduino.

1.3 JUSTIFICATIVA

Uma linha de pesquisa, visando melhorar o processo de estudo é direcionada a utilização de ferramentas e software computacionais como ambiente de estudo (SANTOS; RODRIGUES; HEITOR, 2005).

Segundo Soares (2004), a partir de observações em aulas de graduação, percebe-se um melhor resultado de aprendizado por meio de atividades práticas de desenvolvimento utilizando ferramentas e simuladores didáticos de representação de conceitos abstratos.

De acordo com Soares (2004) existem problemas computacionais interessantes em termos de grafos. Entretanto, parte dos alunos enfrentam dificuldades na implementação de algoritmos em grafos, devido a necessidade de suporte para implementação. Implementações essas que são compostas por estruturas de dados, ferramentas de visualização e interação com grafos, diferença de níveis de abstração entre as definições teóricas dos algoritmos e representações computacionais necessárias para implementá-los.

Segundo Resnick (1998), a construção do conhecimento acontece de forma mais efetiva quando as pessoas estão engajadas em construir algo que seja significativo para si e para as pessoas ao seu redor. Quando o aluno se envolve na implementação e na modelagem dos robôs, ele passa a ter um senso de determinação muito maior na relação com este objeto do que se recebesse o robô pronto. Desta forma, os alunos estão tendentes a explorar e a fazer intensas conexões com conceitos científicos fundamentados em suas atividades.

Diante desses fatos foi implementado um software que utiliza algoritmo de Dijkstra, para determinar a trajetória de menor caminho e auxiliar os alunos no aprendizado da disciplina de teoria dos grafos. Como forma de mostrar aos alunos o funcionamento na prática do algoritmo de Dijkstra, foi implementado um robô autônomo, na qual é capaz de realizar a trajetória através de uma linha sobre uma área planejada, de acordo com o ponto de início até um ponto de destino informado pelo usuário.

1.4 ESTRUTURA DO TRABALHO

O capítulo 2, apresenta as definições de grafos, descreve o problema do menor caminho, além de mostrar a história de algoritmo de Dijkstra, seu funcionamento e suas aplicações.

O capítulo 3, relata-se uma breve história da robótica móvel, descreve-se as definições de robô, também apresenta os sensores utilizados em robótica móvel descrevendo os tipos de sensores e onde pode ser empregado, apresenta as linguagens de programação utilizadas em sistemas robóticos, define o que são essas linguagens de programação para robô e apresenta quais são os três níveis de programação para robôs.

O capítulo 4, trata-se do Arduino, descreve-se suas principais características, além de relatar sobre sua IDE de programação, utilização do Arduino para projetos escolares e descreve as principais *Shilds* utilizadas.

O capítulo 5, são exibidos os trabalhos correlatos que foram utilizados como base de pesquisa para implementação desse projeto.

O capítulo 6, descreve o trabalho desenvolvido. São apresentadas todas as etapas de desenvolvimento começando pelas etapas metodológicas seguindo do desenvolvimento do protótipo e resultados obtidos.

O capítulo 7, apresentada os resultados e discussões bem como testes realizados para chegar no objetivo do trabalho.

Por fim é realizada a conclusão e as considerações finais, além de descrever as propostas para trabalhos futuros.

2 GRAFOS

Conforme Goodrich e Tamassia (2007) grafo é um conjunto de elementos titulados de vértices com uma relação de conexões entre pares de vértices. Eles relatam que um grafo é focalizado na representação e nos algoritmos para lidar efetivamente com mapas onde os objetos são as estradas e nas tabelas de roteamento para a Internet nas quais os objetos são os computadores

Boaventura e Jurkiewicz (2009) exemplificam a utilização de grafos no torneio de vôlei de uma escola. Nesse torneio participam as turmas 6A, 6B, 7A, 8A e 8B. Abaixo a representação dos jogos realizados:

a) 6A jogou com 7A, 7B, 8B

b) 6B jogou com 7A, 8A, 8B

c) 7A jogou com 6A, 6B

d) 7B jogou com 6A, 8A, 8B

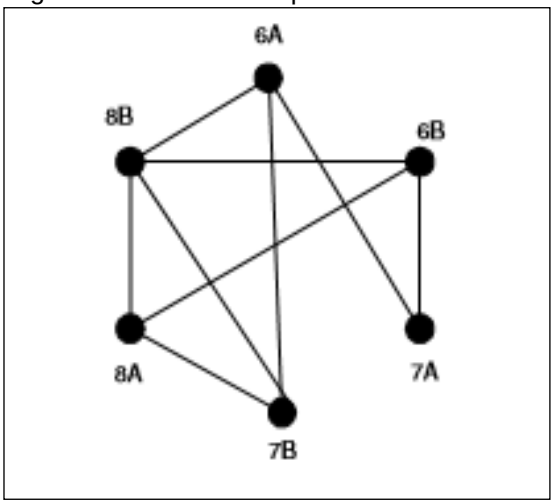
e) 8A jogou com 6B, 7B, 8B

f) 8B jogou com 6A, 6B, 7B, 8A.

Essa listagem pode ser representada a partir de um grafo, tornando assim a representação da visualização e consistência da informação mais eficiente (BOAVENTURA; JURKIEWICZ, 2009).

A figura 1 representa o grafo mencionado acima. Os vértices representam a turma e os jogos são representados pelas arestas.

Figura 1 - Grafo do campeonato de vôlei



Fonte: Boaventura e Jurkiewicz (2009).

Conforme Boaventura e Jurkiewicz (2009) um grafo bem definido possui dois conjuntos:

a) o conjunto V, Vértices - no exemplo é representado pelas turmas.

b) o conjunto A, Arestas - no exemplo é representada pelos jogos.

Um grafo pode ser definido em dirigidos, não-dirigidos ou mistos. Quando as arestas de um grafo não forem dirigidas, ou seja, não tem setas informando qual a direção da mesma, então é dito que é um grafo não-dirigido. Um grafo dirigido, ou dígrafo possui todas as arestas dirigidas. O grafo misto possui um conjunto de arestas dirigidas e não dirigidas (GOODRICH; TAMASSIA, 2007).

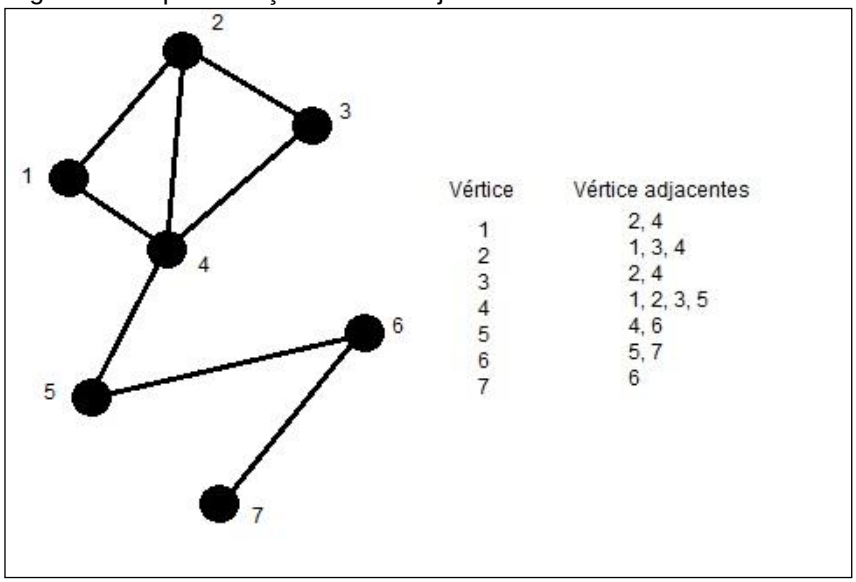
2.1 REPRESENTAÇÃO DO GRAFO

Existem várias maneiras de organizar os dados sobre um grafo de modo que possam ser representados em um computador. As mais utilizadas são as listas de adjacência e matriz de adjacência.

2.1.1 Lista de adjacência

Consiste em dizer, para cada vértice, quais outros vértices estão ligados a ele, ou adjacente a eles (BOAVENTURA; JURKIEWICZ, 2009). A figura 2 é um exemplo da representação do grafo utilizando lista de adjacência.

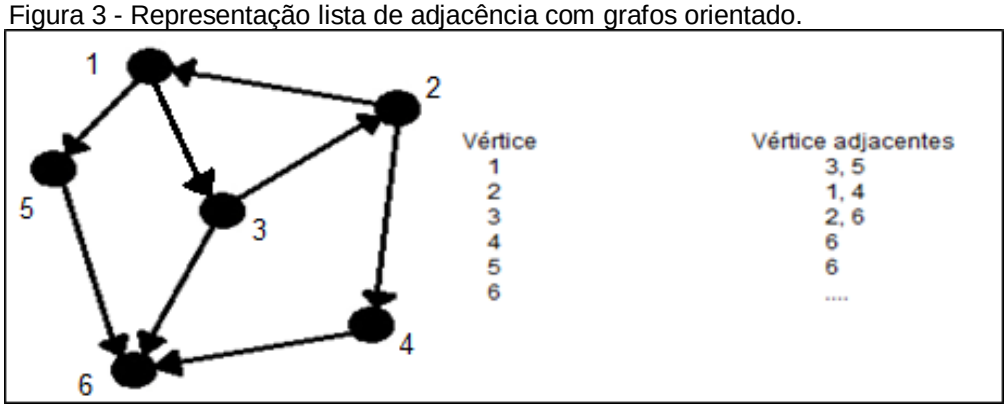
Figura 2 - Representação lista de adjacência.



Fonte: Boaventura e Jurkiewicz (2009).

Existe também a representação de grafos orientados, em que as arestas possuem apenas um sentido. A figura 3 representa um grafo orientado utilizando lista de adjacência:

Figura 3 - Representação lista de adjacência com grafos orientado.



Fonte: Boaventura e Jurkiewicz (2009).

2.1.2 Matriz de adjacência

A partir da matriz de adjacência pode-se localizar qualquer posição em uma matriz por meio de um par de índices. Os índices da matriz correspondem os vértices correspondente, caso o valor contido na matriz na posição dos índices seja 1 então existe ligação entre esses vértices caso for 0 não existe ligação (GOODRICH; TAMASSIA, 2007).

A matriz pode conter valores de pesos, que são considerados como comprimento. Quando um grafo na qual as ligações recebem valores diversos é dito como valorado. Nesse caso a matriz é conhecida como matriz de valores das ligações ou simplesmente matriz de valores (GOODRICH; TAMASSIA, 2007).

As figuras 4 e 5 representam um vetor de matriz de adjacência, onde a figura 3 é de um grafo orientado e a figura 4 representa um grafo que possui pesos em suas arestas.

Figura 4 - Matriz de adjacência

	1	2	3	4	5	6	7
1	0	1	0	1	0	0	0
2	1	0	1	1	0	0	0
3	0	1	0	1	0	0	0
4	1	1	1	0	1	0	0
5	0	0	0	1	0	1	0
6	0	0	0	0	1	0	1
7	0	0	0	0	0	1	0

Fonte: Goodrich e Tamassia (2007).

Figura 5 - Matriz de adjacência com pesos

	1	2	3	4	5	6
1			2		1	
2	4			3		
		1				7
4						6
5						1
6						

Fonte: Goodrich e Tamassia (2007).

Na primeira matriz é definido zero quando não existe relação de adjacência. Para a segunda matriz é definido o valor do peso da aresta quando existe relação de adjacência ou nulo quando não existe.

2.2 HISTÓRIA

Leonard Eule (1736) teria sido o primeiro, segundo Santos (1995), a descrever tal teoria. Contam que o provo da cidade de Königsberg, banhada pelo Rio Pregel, com sete pontes ligando duas ilhas e as margens opostas do rio, o problema consistia em determinar se era possível realizar um passeio atravessando cada uma das pontes uma única vez e voltando ao ponto de partida proposto pelo matemático Leonard Euler.

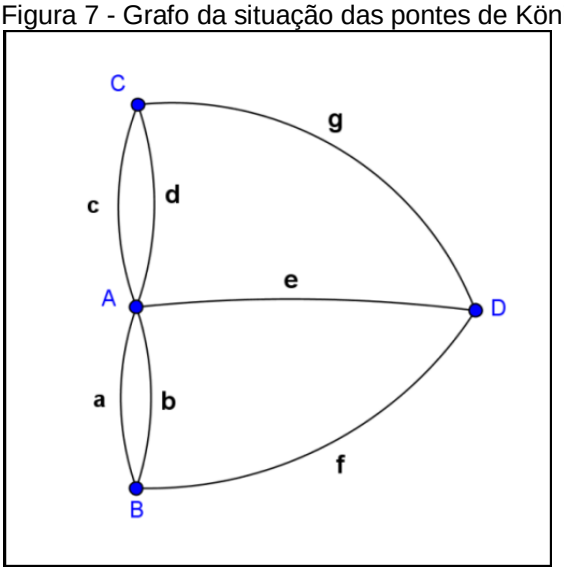
Figura 6 - Pontes de Königsberg

Fonte: Malta (2008).

Segundo Matos (2013) Euler retirou tudo que importante para a resolução do problema, e elaborou um esquema comparável ao da figura 7, com o objetivo de representar quatro partes da cidade através de quatro pontos, representados na figura 1.2 por A, B, C e D, unindo estes pontos através de linhas que representam as pontes.

Euler então eliminou tudo que era acessório para a resolução do problema, e elaborou um esquema análogo ao da figura 7, em que representou quatro partes da cidade através de quatro pontos, representados na figura 1.2 por A, B, C e D, unindo estes pontos através de linhas que representam as pontes.

Figura 7 - Grafo da situação das pontes de Königsberg



Fonte: Matos (2013)

Euler ao descrever sobre o problema das pontes de Königsberg, fomentou e deu início, com decorrer dos anos, cresceu até se transformador no que na atualidade conhecemos por Teoria dos Grafos (MATOS, 2013).

2.3 O PROBLEMA DO CAIXEIRO VIAJANTE (PCV)

O problema do caixeiro Viajante (PCV) é um exemplo clássico de otimização combinatória. Dado um conjunto de n cidades e distância entre elas; o problema consiste em determinar uma rota de custo mínimo passando por todas as cidades percorrendo cada uma delas apenas uma vez e retorne a cidade de destino, de tal forma que a distância total percorrida seja mínima.

O interesse por este problema é devido a sua dificuldade de resolução e também, sua enorme aplicabilidade na engenharia e nas ciências. Vários problemas reais podem ser modelados através de PCV dentre eles roteamento de veículos, programação de tarefas em máquinas, perfuração de placas de circuitos integrados, mapeamento do DNA humano, dentre outras aplicações (LENSTRA; RINNOOY KAN, 1975).

A origem do PCV é creditada a William Rowan Hamilton, que inventou um jogo no qual tinha o objetivo de traçar um roteiro através dos vértices de um dodecaedro (vértices equivalem a cidade) que iniciasse e terminasse no mesmo vértice, realizando apenas uma visita para cada cidade. O Ciclo Hamiltoniano constitui uma solução para o jogo de Hamilton (GOLBARG; LUNA, 1999).

O PCV é chamado simétrico quando a distância entre dois nós (ou cidades) quaisquer i e j independe do sentido, isto é, quando dij = dji; caso contrário o problema é denominado assimétrico. Segundo Helsgaun (2000), problemas assimétricos são, em geral, mais fáceis de serem resolvidos do que problemas simétricos.

Os processos de solução para o PCV podem ser rotulados em exatos e heurísticos. Os métodos exatos na maioria das vezes se baseiam em procedimentos de enumeração implícita em árvore, para os quais têm sido propostas diferentes funções limitadoras. Os métodos exatos possuem aplicação limitada para a solução do PCV, tendo em vista a complexidade combinatória destes problemas. (HELSGAUN, 2000)

Devido a incapacidade dos métodos exatos, métodos heurísticos compõem o principal foco de interesse para a resolução do PCV. Heurísticas são procedimentos de resolução de problema que muitas vezes se apoiam em uma abordagem intuitiva, na qual é explorada de forma inteligente a estrutura particular do problema, para a obtenção de uma solução adequada. Assim, na maior parte dos casos as heurísticas propostas tendem a ser muito específicas e particulares para um determinado problema, necessitando de robustez; isto é, não conseguem dar boas soluções para problemas com características, condicionantes ou restrições pouco diferentes daquelas para as quais foram desenvolvidas (CUNHA, 1997).

2.4 PROBLEMA DO MENOR CAMINHO

O problema de menor caminho é muito conhecido da teoria de grafos, ele é utilizado em muitas aplicações, por exemplo, em telecomunicações, transporte, correios, entre outros, onde seja necessário encontrar um caminho de menor custo, ou mais rápido, entre dois pontos (SILVA; FERNANDES; SANCHES ,2009).

Esse problema consiste em encontrar o menor caminho entre uma origem e um destino, dentro de uma rede, minimizando o custo de travessia de um grafo, entre dois pontos, esses custos são dado pela soma dos pesos de cada aresta percorrida (SILVA; FERNANDES; SANCHES ,2009).

Os algoritmos responsáveis em solucionar o problema do menor caminho são normalmente chamados de "algoritmos de busca de caminhos". O algoritmo que

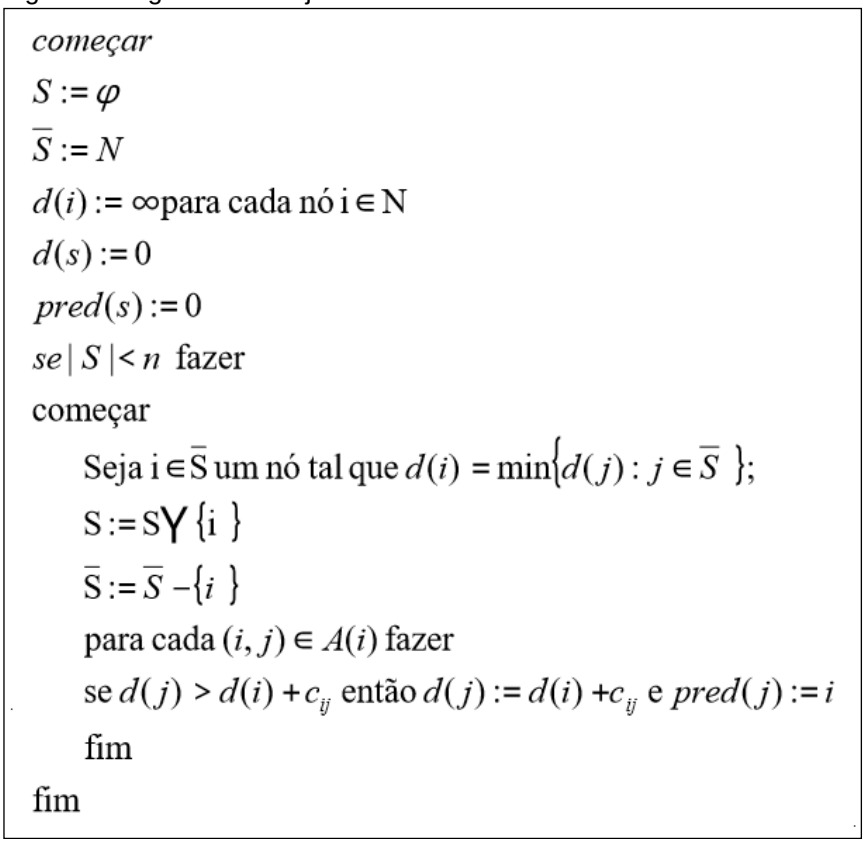
mais se destaca pela sua simplicidade é o algoritmo de Dijkstra (SILVA; FERNANDES; SANCHES ,2009).

2.5 ALGORITMO DE DIJKSTRA

O algoritmo de Dijkstra, foi desenvolvido em 1959 por Edsger Wybe Dijkstra, por isso foi dado esse nome ao algoritmo. Este algoritmo funciona apenas em grafos que os pesos das arestas não forem negativos. Ele é aplicado em grafos simples que tem o propósito de determinar o caminho de menor custo entre dois vértices. Quando o grafo não é simples, então o algoritmo procura transformá-lo em um grafo simples eliminando seus laços e eventuais arestas múltiplas deixando apenas com o peso menor. A complexidade do Algoritmo de Dijkstra é O (n^2), onde n é a quantidade de vértices do grafo (ZAMBONI, 2006).

Segundo Méndez e Gardia (2008) o algoritmo de Dijkstra encontra o menor caminho de um nó fonte até um nó de origem de uma rede orientada para o caso em que todos os pesos dos arcos sejam não negativos. O algoritmo grava a distância rotulada para cada nó inicial e o limite superior do caminho mais curto até o nó de origem. Os nós são divididos em dois grupos: os rotulados permanentemente e rotulados temporariamente. A distância rotulada dos nós permanentes representa a menor distância do nó fonte até um outro nó. No caso dos nós temporários, esta distância representa o limite superior da distância do caminho mais curto até o nó como pode ser visto na representação do algoritmo de Dijkstra na figura 11.

Figura 8 - Algoritmo de Dijkstra



Fonte: Ahuja(1993)

O algoritmo percorre todos os nós sem incluir o nó fonte e rotula permanentemente os nós em ordem das distâncias ao nó fonte. O algoritmo termina quando todos os nós são designados como permanentes

3 ROBÓTICA MÓVEL

O estudo da robótica pertence ao desejo de resumir alguns aspectos da função humana pelo uso de mecanismos, sensores, atuadores e computadores. Neste capitulo será abordado os temas mais importantes de robótica móvel.

3.1 HISTÓRIA ROBÓTICA MÓVEL

Craig e John (2012) relatam que a história da automação industrial se caracteriza por períodos de rápidas mudanças em métodos populares. Segundo ele as técnicas de automação estão intimamente interligadas a economia mundial. O uso de robô industrial se tornou identificável como dispositivo ímpar na década de 1960.

Craig e John (2012) ainda relatam que na América do Norte, foi muito intensa a adoção de equipamentos robóticos no início da década de 1980. Desde então o mercado de robótica vem crescendo.

Hoje em dia a robótica móvel vem relativamente crescendo seu desenvolvendo a cada ano. A partir da década de 50 pesquisadores já se interessavam em desenvolvimento de robôs móveis. Willian Walter construiu vários modelos de robôs móveis na década de 1950, onde os mesmos eram capazes de efetuar tarefas tais como desviar de obstáculos seguir fontes laminosas, utilizando capacitores para o controle do robô (WALTER, 1950).

Nilsson em 1969 desenvolveu o robô SHAKEY em Stanford. O robô empregava uma configuração diferenciada com dois motores de passo para se locomover e era equipado com sensores de distância, câmeras de vídeo e sensores táteis. O robô se conectava ao computador via links de rádio e de vídeo. A função do robô SHARKEY era de desviar de obstáculos e a movimentação de blocos coloridos. O mesmo possuía dificuldades em processar e interpretar as informações dos sensores obtidas do ambiente, e nunca foi capaz de completar uma sequência completa de ações em um ambiente real (NILSSON,1969).

Walter (1950) também construiu na década de 50 robôs denominados como tartarugas, conforme figura 1, com os nome de Elmer e Elsie. Cada tartaruga possua dois comportamentos diferentes: no primeiro a tartaruga se afastava do

objeto ao colidir com um obstáculo, e no segundo, Ela se aproximava da fonte de luz, com exceção se a fonte de luz fosse muito forte, onde também se afastava. Mesmo possuindo limitações de comportamentos eles eram capazes ainda assim, de exibir algumas interpretações complexas com o ambiente e entre si, por exemplo se um deles possuía uma fonte de luz.

Na década de 70 foi desenvolvido o robô móvel HILARE no LAAS em Toulouse. Sendo este um dos primeiros robôs desenvolvidos na Europa. HILARE utilizava câmeras de vídeo, sensores de distância a laser e ultrassom para locomover-se no ambiente. Utilizava-se a representação poligonal do ambiente para planejamento de trajetória. Utilizava os sensores ultrassônico para evitar os obstáculos próximos (BRIOT;TALOU; BAUZIL, 1979)

No início do século XX com a necessidade de aumentar a produtividade industrial e a qualidade dos produtos teve-se a ideia de construir robôs autônomos. Nessa época George Devol pode ser considerado o pai da robótica. O seu primeiro robô, o "Unimate", lançado em 1962, tinha como função automatizar algumas tarefas industriais. Um dos primeiros clientes foi a General Motors, que obteve sucesso em utilizar essa tecnologia (ROSÁRIO, 2007).

3.2 DEFINIÇÃO DE ROBÔ

Segundo a instituição americana *Robotics Industries Association* (RIA) robô é um manipulador reprogramável com múltiplas funcionalidades projetado para manipular materiais, peças, ferramentas ou dispositivos especiais, por meio de movimentos programados para a realização de múltiplas de tarefas (REDEL; HOUNSELL, 2004).

A diferença entre um robô e um computador se dá pela forma que o mesmo interage com o mundo, onde o computador necessita ser inicializado e operado por um humano, já o robô é considerado um mecanismo inteligente pois opera de forma autônoma sem a intervenção humana (MURPHY, 2000).

3.3 DEFINIÇÃO DE ROBÓTICA

A robótica é a área que se relaciona com o desenvolvimento de tais dispositivos. Busca o desenvolvimento e a integração de técnicas e algoritmos para a criação de robôs.

PIO (2006, p. 197) define a robótica como:

> A ligação inteligente entre a percepção e a ação. Trabalhar em robótica significa estudar, projetar e implementar sistemas ou dispositivos que, com a utilização de percepção e de certo grau de inteligência, sejam úteis na realização de uma determinada tarefa, pré-definida ou não, que envolva interação física entre o sistema (ou dispositivo) e meio onde a tarefa está sendo realizada.

A automação industrial e a robótica estão intimamente relacionadas. A robótica é considerada a forma de utilização de tecnologias de robôs na produção. Uma definição formal seria "Uma ciência da engenharia aplicada, tida como uma combinação da tecnologia de máquinas operatrizes e ciência da computação (REDEL; HOUNSELL, 2004).

3.4 SENSORES PARA ROBÔS MÓVEIS

Os robôs móveis autônomos, possuem diferentes configurações de dispositivos de hardware embarcados, de acordo com a tarefa para a qual foi designado. Os principais dispositivos de um robô são seus sensores e atuadores.

A figura 12 mostra os principais dispositivos de sensores mais usados em robótica (DUDEK; JENKIN, 2000).

Na figura 13 são apresentados os principais atuadores utilizados em robótica móvel.

Nas figuras 12 e 13 pode-se observar a complexibilidade de projetar um robô autônomo devido a quantidade de combinações possíveis que um projeto de sistema robótico possui, onde o mesmo envolve especificação e seleção de diferentes componentes, sensores e atuadores de acordo com a necessidade do projeto. Este sistema deve ser projetado de modo a ser dotado de dispositivos

capazes de prover os dados necessários (obtidos através dos seus sensores), para que o sistema de controle robótico inteligente possa planejar e realizar o acionamento dos seus dispositivos de modo a executar a ação desejada.

Figura 9 - Sensores para robôs móveis

Sensor	**Principal função**	**Exemplos**
De Posição e orientação	Determinar a posição absoluta ou direção de orientação do robô	GPS (Sistema de posicionamento global) Bússola Inclinômetro
De obstáculo	Determinar a distância até um objeto ou obstáculo	Sensor infra-vermelho Ultrassom Radar Sensor Laser Sistemas de visão estéreo
De contato	Determinar o contato com um objeto ou posição de contato com marcação	Sensores de contato (bumpers, switches) Antenas (animal whiskers) Marcações (barreiras ópticas e magnéticas)
De deslocamento e velocidade	Medir o deslocamento do robô medidas relativas da posição e orientação do robô	Inercial (Giroscópio, Acelerômetros) Odômetro (encoders: optical, brush) Potenciômetros (angular) Sensores baseados em visão
Para comunicação	Envio e recepção de dados e sinais externos (troca de informação)	Sistemas de visão e sensores óticos Sistemas de comunicação (RF)
Outros tipos	Sensores magnéticos, indutivos, capacitivos, reflexivos, sensores de temperadora, carga (bateria), pressão e força, entre outros. Detectores: detector de movimento, de marcações, de gás/odores	

Fonte: Dudek e Jenkin (2000).

Figura 10 - Atuadores para robôs móveis

Atuador	**Principal tipo/Função**	**Exemplos**
Base fixa	Braço robótico com base fixa	Robôs industriais PUMA
Base Móvel: Rodas	2 Rodas independentes (Diferencial) 3 Rodas (Triciclo, omni-directionais) 4 Rodas (veículos robóticos - ackermann)	Robôs Khepera e Pioneer P3-DX Robô BrainStem PPRK Stanley - Stanford(Darpa Challenge)
Base Móvel: Esteira	Esteira (Slip/Skid locomotion - tracks)	Tanques e veículos militares
Base Móvel: Juntas e Articulações	Bipedes 4 Patas (quadpods) 6 Patas (hexapods)	Robôs humanóides Robôs Sony Aibo, BigDog Robôs Insertos (Lynxmotion Hexapods)
Base Móvel: Propulsão Hélices ou turbinas	Veículos aéreos com hélices Veículos aquáticos com hélices Veículos sub-aquáticos	Aviões, Helicópteros e Dirigíveis Barcos autônomos Submarino autônomos
Outros tipos	Braços manipuladores com base móvel Garras com ou sem feedback sensorial Mecanismo de disparo	Garras (Grippers) embarcadas Mão robótica Disparo do chute (futebol de robôs)

Fonte: Dudek e Jenkin (2000).

Integrar todas as informações obtidas através desses sensores é um dos grandes desafio da robótica, de modo a gerar comandos e controlar os diferentes dispositivos de atuação de um robô, garantindo que a tarefa seja executada de acordo com suas especificações sem colocar em risco tanto o robô quanto aqueles que o cercam. A integridade do robô deve ser preservada bem como a dos elementos presentes no mesmo ambiente do robô. Um robô só deve agir de modo ativo sobre um determinado objeto alvo, se realmente for programada a sua ação sobre este objeto. Esta questão sempre foi alvo de muitas discussões (em propostas como as 3 leis da robótica de Asimov (1968), pois um robô móvel pode ocasionar danos tanto a pessoas como a objetos que o rodeiam, de modo apurado ou não, construindo-se assim de um sistema embarcado crítico, onde seu desenvolvimento deve imperativamente abranger um cauteloso projeto tanto de hardware como de software (WOLF, 2009).

Os sensores individualmente possuem funcionalidades incompleta e sujeita a erros, sendo o papel do sistema de controlar e adquirir as informações dos sensores tratando-as de forma robusta e inteligente. Os comandos de atuação também não são precisos, muitas vezes eles podem sofrer forças externas que podem modificar sua rota e causar problemas de posicionamento ou então uma

instrução pode ser mal interpretada e executada de forma incorreta e imprecisa, pois, está sujeita a erros de posicionamento do robô, de acionamento dos motores, bem como está sujeita também às forças e componentes externos (e.g. fricção, gravidade, aceleração, desaceleração, inércia, colisão com obstáculos e derrapagem das rodas). Cabe ao sistema de controle fornecer técnicas que permitam equilibrar e corrigir estes erros de modo que as tarefas do mesmo possam ser executadas corretamente e em segurança (WOLF, 2009).

3.5 LINGUAGEM E SISTEMAS DE PROGRAMAÇÃO DE ROBÔS

A linguagem e sistemas de programação de robôs é responsável pela interface entre o usuário humano e um robô. É por essa interface que o usuário utiliza de toda a mecânica e dos algoritmos de controle para o domínio do robô (CRAIG, 2012).

Os manipuladores robóticos diferem-se da automação fixa devido a sua capacidade de serem 'flexíveis', o que significa programáveis tendo não apenas os movimentos dos manipuladores programáveis mas também emprego dos sensores de comunicação com outras automações do ambiente, podendo assim se adequar às mudanças no decorrer do cumprimento da tarefa (CRAIG, 2012).

3.6 OS TRÊS NÍVEIS DA PROGRAMAÇÃO DE ROBÔS

Segundo Craig (2012) muitos estilos de interface de usuário já foram desenvolvidos para programação de robôs. Os controladores robóticos pareciam sequenciais simples usadas em geral para o controle de automação fixa, isso antes da rápida proliferação dos microcomputadores.

3.6.1 Ensinar mostrando

De acordo com Craig (2012) os primeiros robôs eram todos programados através de um método que ele descreveu como 'ensinar mostrando', que implica em mover o robô até um local desejado e registrar sua posição em uma memória que um sequenciador leria durante a reprodução. A fase de ensinar o mesmo era

realizada de forma manual, onde o operador movia à mão ou também pela interação com uma caixa de controle (conhecida coom *teach pendant*). Teach pendant são botoeiras portáteis que permitem o controle de cada junta do manipulado.

3.6.2 Linguagem de programação explicitas

Essas linguagens possuem características especiais que se aplicam aos problemas de programar manipuladores e são, portanto, chamadas de linguagens de programação de robôs, do inglês *Robot Programming Languages* (RPL).

Craig (2012) afirma que essas linguagens podem ser divididas em três categorias, são elas:

a) **Linguagem de manipulação especializada**: foi implementada mediante o desenvolvimento de uma linguagem completamente nova que, apesar de ser voltada para áreas robóticas em particular, podem muito bem ser consideradas linguagens genéricas de programação de computador;
b) **biblioteca de robótica para uma linguagem de programação existente**: essas linguagens de programação foram desenvolvidas com base em outras linguagens, como por exemplo pascal, depois é acrescentado a biblioteca de sub-rotinas específica para a robótica. Então o usuário escreve o programa, em pascal e frequentemente chama as sub-rotinas específicas conforme as necessidades robóticas;
c) **biblioteca de robótica para uma nova linguagem de uso geral**: essa linguagens de programação foram desenvolvidas primeiramente criando uma nova linguagem de uso geral e depois acrescentado a biblioteca de sub-rotinas predeterminadas, específicas para robótica.

3.6.3 Linguagem de programação em nível de tarefa

Programação a nível de tarefa consiste em permitir que o usuário administre as submetas da tarefa diretamente em vez de mencionar os detalhes de todas as ações que o robô deverá desempenhar. Nesse tipo de programação o usuário tem a possibilidade de acrescentar instruções no programa da aplicação em

um nível mais alto do que em uma linguagem explícita de programação de robôs (CRAIG, 2012).

Em resumo o programador específico apenas as ações que o robô deve realizar sobre os objetos. Por exemplo, se a instrução for passada para o robô pegar um objeto qualquer em uma superfície, o sistema deve planejar a trajetória para o manipulador evitando colisões com outros objetos e definir qual o melhor local para pegar o objeto em questão (CRAIG, 2012).

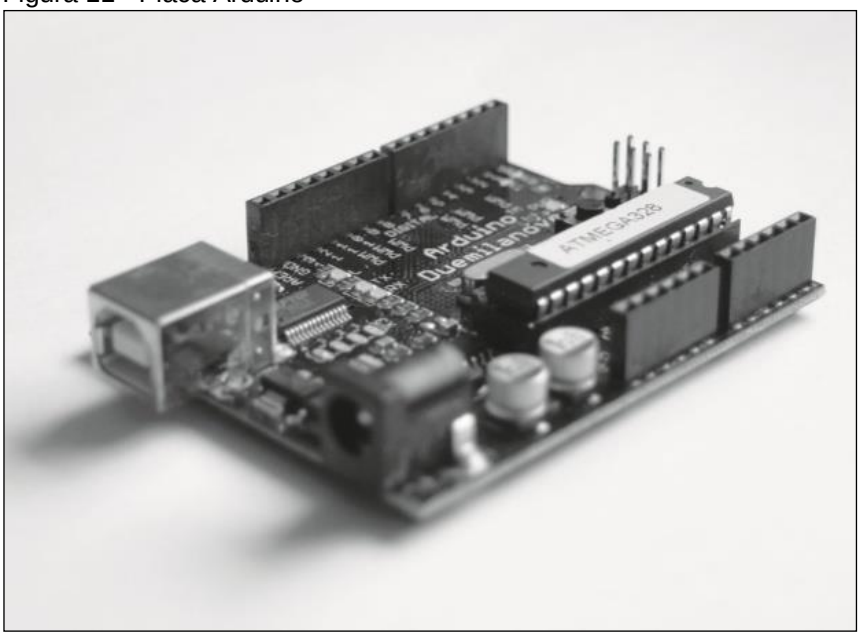
4 ARDUINO

Arduino é um micro controlador de placa única com um conjunto de software para programá-lo. A placa do Arduino possui suporte embutido a entrada/saída de periféricos como sensores e placas externas. Em termos práticos o Arduino é um microcomputador que pode ser programador para processar entradas e saídas entre o dispositivo e componentes externos conectados a ele. O Arduino pode ser chamado de computação física ou embarcada, ou seja, um sistema que possui a capacidade de interagir com o ambiente por meio de hardware e software (MCROBERTS, 2011).

4.1 CARACTERÍSTICAS

A placa do Arduino é composta de um micro controlador Atmel AVR, um cristal ou oscilador que tem a função de enviar pulsos de tempo em uma frequência especificada e um regulador linear de 5 Volts como pode ser visto na figura 14. Alguns modelos também possuem saída de USB, permitindo conectá-lo a um computador para enviar ou recuperar dados (MCROBERTS, 2011).

Figura 11 - Placa Arduino



Fonte: Mcroberts(2011).

A programação do Arduino é realizada utilizando a IDE do Arduino, um software livre que permite escrever o código em linguagem que o mesmo compreende. O IDE possibilita que seja escrito um programa de computador que é um conjunto de instruções passo a passo, tendo como função de manipular os periféricos conectados ao Arduino para realização de uma metaespecífica (MCROBERTS, 2011).

Uma outra opção de extensão do Arduino é utilizando os *Shields que* também são chamados de escudos, na qual são placas de circuitos contendo outros dispositivos como por exemplo GPS, display LCD, módulos de Ethernet entre outros (MCROBERTS, 2011).

4.2 IDE ARDUINO

O IDE do Arduino torna mais fácil para escrever o código e enviá-lo para o console. O software é suportado pelos sistemas Windows, Mac OS x e Linux. O ambiente é escrito em Java e baseado em c/c++. O software é compatível com qualquer placa de Arduino (ARDUINO, 2016).

O IDE é dividido em três partes principais do aplicativo, a Toolbar no topo, o código ou a Sketch Window no centro, e a janela de mensagens na base. O Toolbar na parte superior da Figura 14 possui seis botões que fornecem acesso conveniente as funções mais utilizadas. Os botões são Verificar, carregar, novo, abrir, salvar e monitor serial (ARDUINO, 2016).

.

Figura 12 - Toolbar

Fonte: Arduino (2016).

As funcionalidades de cada botão da IDE do Arduino são:

a) **verificar**: verifica se possui erros no código;
b) **carregar**: faz upload do código para o Arduino;
c) **novo**: cria um novo Sketch em branco, pronto para receber o novo código;
d) **abrir**: mostra uma lista de sketches, que possui armazenado no computador e uma lista de exemplos disponíveis para ser utilizados;
e) **monitor serial**: exige os dados seriais enviados do Arduino.

4.3 POR QUE ARDUINO

O Arduino tem sido utilizado por milhares de projetos em todo o mundo devido a sua facilidade de utilização e preço acessível. O software Arduino é fácil de usar para iniciantes, mas flexíveis o suficiente para usuários avançados (ARDUINO, 2016).

Arduino também simplifica o processo de trabalhar com microcontroladores, além de possuir outras vantagens que o torna o Arduino muito utilizado (ARDUINO, 2016).

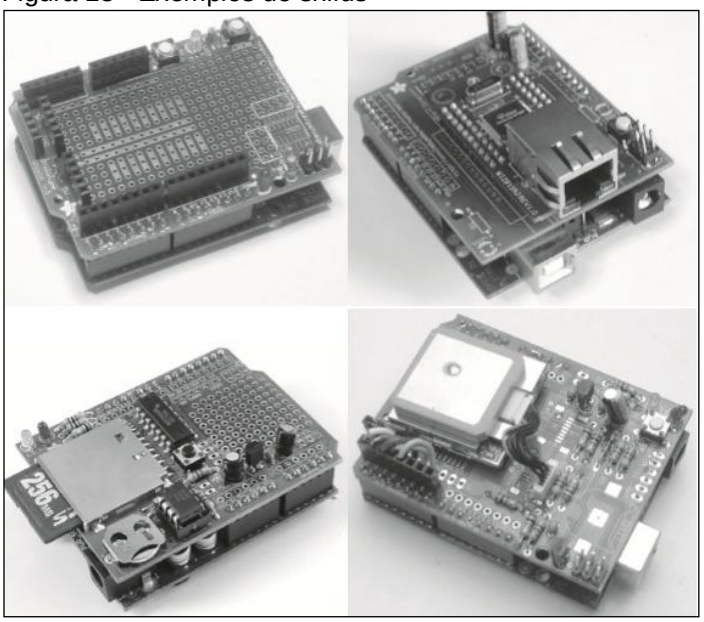
a) **Barato**: placas Arduino são relativamente baratos em comparação com outras plataformas de microcontroladores;
b) **Multi-plataforma**: O software Arduino (IDE) é executado em sistemas operacionais Windows, Macintosh OSX e Linux. A maioria dos outros sistemas de microcontroladores são exclusivos para Windows;
c) Open souce e software extensivo: O software Arduino é publicado como ferramenta de código aberto, disponível para extensão por programadores mais experientes;
d) Open souce e hardware extensível: Os projetos das placas Arduino são publicados sob uma licença Creative Commons, para que os designers de circuito experientes podem fazer a sua própria versão do módulo, entendê-lo e melhorá-lo.

4.4 SHIELDS

Um *Shield* constitui em uma placa de circuito, desenvolvida e preparada para se encaixar devidamente ao Arduino (MCROBERTS, 2011).

A figura 16 mostra os *shilds protoshield*, *shield* para *Ethernet*, *shield* registrador de dados, e um *shield* GPS.

Figura 13 - Exemplos de shilds



Fonte: Mcroberts (2011).

Os Shields possuem o benefício de oferecer novas funcionalidades ao Arduino, bastando apenas ser conectados a ele, utilizando a biblioteca do Shields para o funcionamento correto. Há todo tipo de Shields, capazes de estender as funções do Arduino para incluir elementos como acesso Ethernet, GPS, controle de relé, acesso a cartões SD e Micro SD, displays LCD, conectividade com TVs, comunicação wireless, telas sensíveis ao toque, displays de matriz de pontos e controle de iluminação DMX.

5 TRABALHOS CORRELATOS

Um sistema integrado para navegação autônoma de robôs móveis, trabalho feito por Oliveira (2010), este projeto consiste na adaptação e extensão de um sistema integrado para navegação autônoma de robô através do aperfeiçoamento da interface e também da incorporação de uma técnica de mapeamento topológico. Para isso, a técnica conhecida como Grade de Ocupação, utilizada em geral para mapeamento métrico é combinada com o método de esqueletização de imagens para a realização do mapeamento topológico. Além disso, transformações morfológicas de erosão e abertura adequadas a ambientes reais foram utilizadas, visando reduzir a influência de ruídos na abordagem proposta, uma vez que devido a ruídos inerentes as leituras sensoriais obtidas pelo robô, o mapa topológico gerado apresenta diversas linhas desnecessárias, dificultando consequentemente a tarefa de navegação autônoma. Os resultados obtidos demonstraram que a técnica de esqueletização de imagens combinada ao mapeamento métrico do ambiente é uma forma simples e viável de se obter as linhas topológicas do espaço livre do ambiente.

RoboEduc: uma metodologia de aprendizado com robótica educacional, trabalho feito por Silva (2009). Este trabalho tem por objetivo propor uma metodologia para ensino de robótica no Ensino Fundamental, baseada na teoria sócio histórica de Lev Vygotsky. Essa metodologia em conjunto com o kit Lego Mindstormsr e um software educacional compõem o sistema de robótica pedagoga denominado de RoboEduc. As atividades visaram produzir conhecimento sobre a construção de protótipos robóticos, sua programação e controle. Os resultados obtidos com o desenvolvimento das oficinas foi a possibilidade de analisar a utilização do robô como elemento mediador do processo de ensino-aprendizagem e as contribuições que o uso da robótica pode trazer para o ensino desde o nível fundamental.

Navegação autônoma de robôs móveis e detecção de intrusos em ambientes internos utilizando sensores 2D e 3D, trabalho desenvolvido por Correa (2013), o objetivo do trabalho é o desenvolvimento de um sistema de percepção do ambiente visando a navegação de robôs móveis autônomos em ambientes internos,

para a execução de tarefas de monitoramento e vigilância de ambientes fechados. O foco principal do trabalho é na navegação de robôs bem como em uma tarefa adicional que é a detecção de intrusos no ambiente em que está se locomovendo. Entre os softwares utilizados estão o OpenCV, uma biblioteca de visão computacional e processamento de imagens, o ROS, um meta-sistema operacional utilizado para o desenvolvimento de aplicações robóticas, o Player/Stage, uma ferramenta de arquitetura cliente/servidor utilizado para o controle de robôs móveis, e o JavaNNS, uma biblioteca construída na linguagem Java para a modelagem e o uso de Redes Neutrais Artificiais. O hardware utilizado foi um robô Pioneer P3AT, um sensor de distância Kinect, um sensor de calor FLIR TAU 640 e um notebook, para a aquisição e processamento de dados e o controle do robô. Como resultado para validar o funcionamento do robô foi necessário realizar a segmentação e o processamento das imagens de ambos os sensores para validar a detecção de seres humanos. A utilização de dois sensores possibilitou um aumento considerável no campo de visão do robô, podendo ter este sistema grande aplicabilidade para vigilância de ambientes internos.

Controle de formação e rastreamento de trajetória para robôs móveis utilizando funções potenciais, trabalho desenvolvido por Lima (2014), este trabalho apresenta uma estratégia de controle de formação e rastreamento de trajetória para sistemas multiagentes formados por veículos não homonômicos, utilizando as informações de posição e velocidade dos agentes vizinhos. A dissertação apresenta os problemas de controle de formação e de rastreamento de trajetória para sistemas multiagentes com parâmetros incertos. Também apresenta propostas de estratégias de controle baseadas em funções potenciais, lideres virtuais e controle de adaptativo. Utilizando uma análise de estabilidade de Lyapunov, foi visto que as estratégias propostas na dissertação garantem a convergência da formação permanecendo um erro residual que pode ser diminuído ao se aumentar os ganhos das estratégias de controle. O padrão geométrico desejado para a formação foi definido através do grafo de comunicação entre os vários robôs que compõem o sistema e das funções potenciais que definem a distância desejada entre os agentes e impedem a colisão entre eles.

6 DESENVOLVIMENTO DO SOFTWARE E IMPLEMENTAÇÕES DO ROBÔ AUTÔNOMO

Nesse capítulo será apresentada as implementações práticas definidas para o desenvolvimento do trabalho de conclusão de curso.

A metodologia utilizada no projeto foi realizada por meio de pesquisas bibliográficas com objetivo de melhorar a forma de estudo em eletrônica, softwares educacionais e grafos, destacando os principais conceitos de grafos e desenvolvimento de robótica. Deu-se preferência para artigos publicados em periódicos, teses e dissertações de mestrado e doutorado. Além disso, foram pesquisados vários projetos que utilizavam de robótica e automação para auxiliar nos estudos e desenvolvimentos de projetos acadêmicos visando tornar as aulas mais práticas e intuitivas para os alunos. A partir de observações dadas aos trabalhos citados, foi possível atingir a finalidade que se propõe este trabalho.

O trabalho é composto por um software responsável por definir as rotas que o robô seguidor de linha deve percorrer de um ponto inicial até um ponto de destino. O software utiliza o algoritmo de Dijkstra para definir as rotas com menor custo e envia as mesmas para o robô. O robô autônomo tem por objetivo percorrer as rotas de menor caminho enviadas do software. O robô utiliza sensores ópticos para identificar a linha e também é composto por dois motores com redutor responsáveis por realizar a movimentação do mesmo. Para o desenvolvimento do seguidor de linha foi utilizando a plataforma de prototipagem eletrônica Arduino. O robô parte de um vértice inicial e segue a linha até encontrar um outro vértice pelo caminho. Logo ele deve tomar uma decisão para qual direção deve-se direcionar. Essas direções são enviadas via Bluetooth do software para o Arduino. Então o Arduino lê essa informação e direciona o robô para o local correto e depois continua a seguir a linha. O robô repete esse procedimento até encontrar o vértice de destino.

6.1 ROBÔ SEGUIDOR DE LINHA

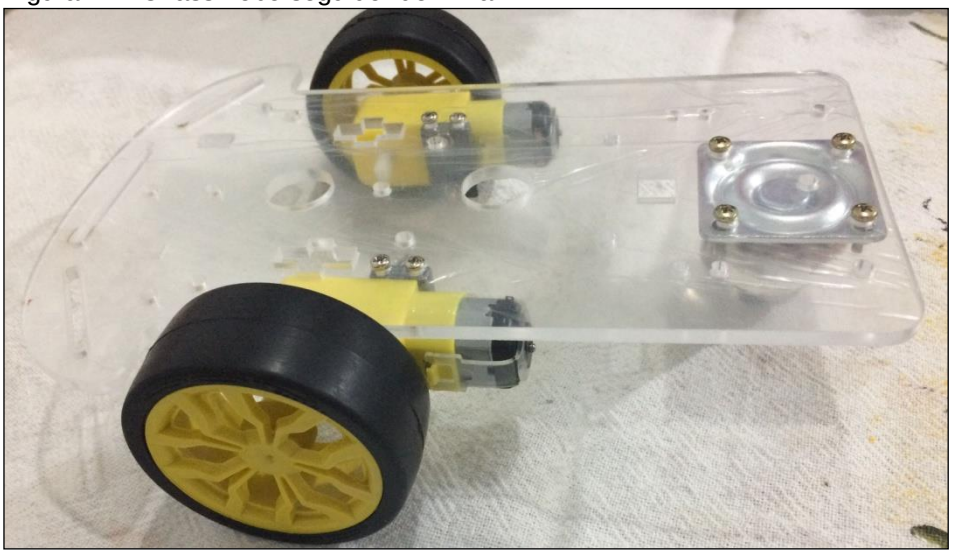
Um bom exemplo de utilização de hardware utilizando Arduino é em aplicações de robótica, como por exemplo em montagem de um robô seguidor de

linha. O robô possui sensores ópticos ligados lado a lado. Conforme a detecção de linha, cada sensor enviará ao Arduino as informações sobre a intensidade do sinal infravermelho refletido e o programa usará essa informação para calcular a velocidade de cada motor.

6.1.1 Chassi

Para facilitar a implementação do robô seguidor de linha, foi utilizado Kits de chassi feitos em acrílico. No Chassi é possível fixar todos os componentes necessários para o desenvolvimento do Robô. Existem vários modelos de Chassi disponibilizados no mercado para essa funcionalidade. Alguns utilizam 4 rodas, outros apenas duas. O Chassi utilizado nesse trabalho foi o de 2 rodas. A Figura 17 mostra o chassi já montado.

Figura 14 - Chassi robô seguidor de linha



Fonte: Do autor

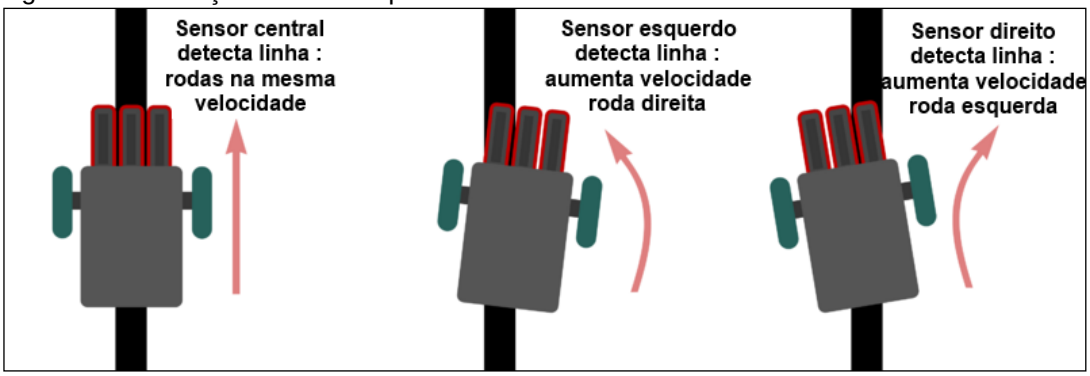
O chassi é composto por dois andares e possui furos específicos para a fixação dos componentes necessários para a implementação do seguidor de linha.

6.1.2 Implementação sensores seguidor de linha

O robô possui sensores ópticos ligados lado a lado. Conforme a detecção de linha, cada sensor enviará ao Arduino as informações sobre a intensidade do

sinal infravermelho refletido e o programa usará essa informação para calcular a velocidade de cada motor. A ilustração abaixo mostra, de forma resumida. Como os sensores se comportam.

Figura 15 - Ilustração sensores ópticos



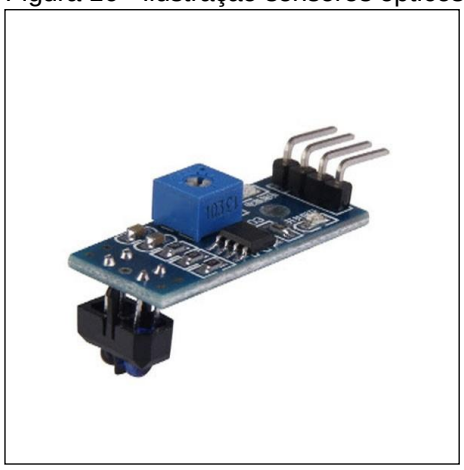
Fonte: FilipeFlop (2017)

O sensor utilizado para identificação da linha foi o módulo TCRT5000 como podemos ver na figura 19. Esse sensor é composto por um LED emissor de infravermelho e um fototransistor responsável por detectar a luz infravermelha.

Nesse módulo, o sensor infravermelho TCRT5000 emite sucessivamente luz infravermelha, através de seu diodo emissor, quando a luz não é refletida ou é refletida de volta, mas a intensidade não é suficientemente forte, o fototransistor não conduz; dessa forma, a saída do módulo é baixa. Se existir objetos na área de detecção, e a intensidade dos raios infravermelhos refletidos for forte o suficiente para saturar o fototransistor, a saída do módulo é alta. O módulo também possui um potenciômetro para ajuste de sensibilidade do sensor. De acordo com o fabricante o módulo possui algumas especificações técnicas.

a) usa o sensor reflexivo infravermelho TCRT5000;
b) distância de detecção 1mm a 8mm;
c) potenciômetro de ajuste de sensibilidade;
d) tensão de trabalho 3.3V a 5V;
e) utiliza comparador de tensão LM393.

Figura 16 - Ilustração sensores ópticos.

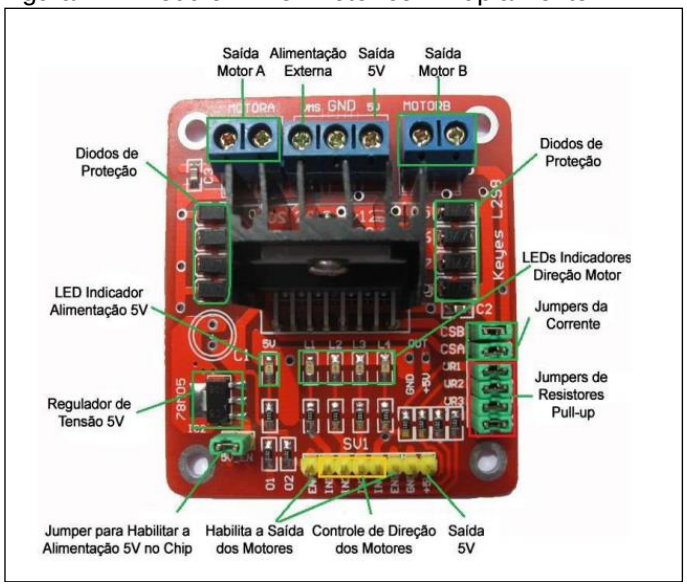


Fonte: Do autor

6.1.3 *Shield* L298N para controlar os motores

Para o controle de velocidade dos motores foi utilizado o módulo driver motor com dupla ponte-H baseado no chip L298N. Esse módulo tem capacidade de controlar a velocidade e o sentido de giro de até dois motores de 3 a 30V DC. A figura 20 ilustra a *shild*.

Figura 17 - Módulo Driver Motor com Dupla Ponte - H - L298N



Fonte: Masugux (2017)

A Quadro 21 representa todas as informações das entradas e saída do módulo e suas funcionalidades.

Figura 18 - Dados entrada e saída do módulo driver motor com Dupla Ponte - H - L298N

Nome da porta	Estado	Descrição
VMS e GND	-	Conexão para fonte de alimentação externa (6V a 35V)
ENA	Entrada	Controle de saída do motor: o estado "baixo (0V)" desativa o motor A
IN1	Entrada	Controle de direção do motor A
IN2	Entrada	Controle de direção do motor A
ENB	Entrada	Controle de saída do motor: o estado "baixo (0V)" desativa o motor B
IN3	Entrada	Controle de direção do motor B
IN4	Entrada	Controle de direção do motor B
Motor A	Saída	Saída para o motor A
Motor B	Saída	Saída para o motor B
CSA / CSB	-	Pinos para testar a corrente elétrica da ponte A / Ponte B
UR1 UR2 UR3 UR4	-	Pinos com resistores *pull-up*, usado em Microcontroladores com baixa resistência de entrada
5V e +5V	-	Saída de 5V
5V_EN	-	Se o jumper estiver conectado no pino 5V_EN o LM7805 irá fornecer 5V para alimentar o *chip* L298. Caso desconecte o jumper do pino será necessário fornecer 5V para o *chip* L298.

Fonte: Masugux (2014)

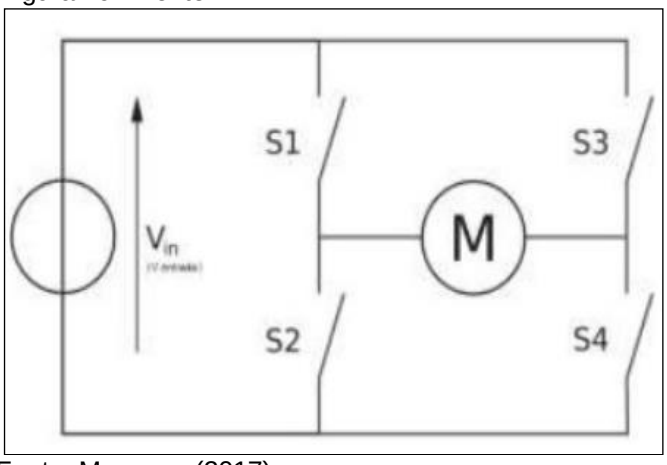
Quando o ENA estiver habilitado a ponte - H permitirá acionar o motor A. Se IN1 estiver ligado em 5V e IN2 em GND o motor A irá girar no sentido horário. Caso o IN1 estiver ligado ao GND e IN2 ligado ao 5V o motor irá girar no sentido anti-horário. O mesmo se dá para o motor B, porém com suas respectivas entradas. Para o controle de velocidade do motor A deve-se conectar o ENA a um pino PWM do Arduino. Para o controle de velocidade a porta de entrada é a ENB.

6.1.4 Ligação dos motores DC

O motor de Corrente Contínua (DC) usa os princípios do magnetismo para girar. O motor de corrente contínua por padrão tem dois imãs em torno de uma grade bobina de fios enrolada em forma de espiral em um rotor ou induzindo. Quando a corrente elétrica flui através do fio em espiral, essa bobina cria um campo magnético que interage com o campo do imã produzindo uma força que faz o rotor girar.

Para obter a rotação dos motores em ambos os sentidos, é necessário alterar o sentido da circulação da corrente aplicada ao motor, ou seja, basta alimentar as bobinas com Vcc e GND. Para fazê-lo girar no sentido contrário, basta inverter o Vcc com o GND. Para facilitar a inversão de rotação dos motores é indicado utilizar uma Ponte H para não ser necessário a utilização de recursos mecânicos como chaves e relés. A figura 22 pode-se verificar um motor DC ligado a quatro chaves. É essa configuração que é conhecida como uma ponte H, pois lembra uma letra H, com a ponte de carga no centro.

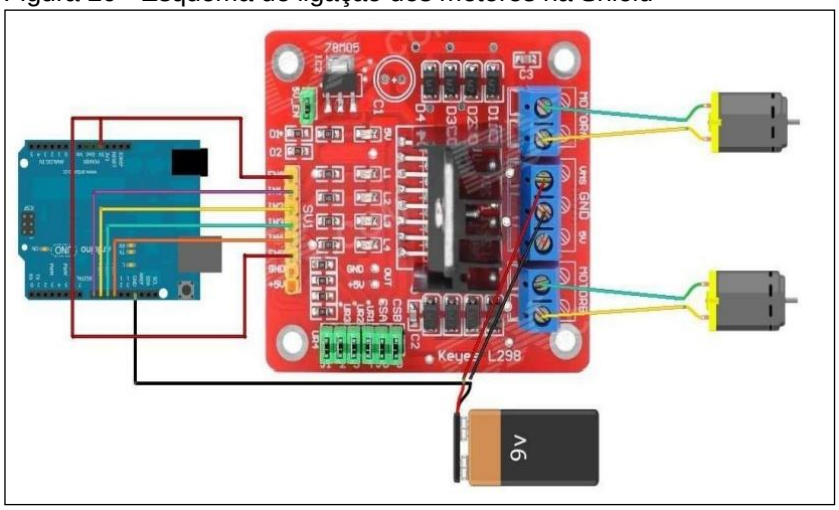
Figura 19 - Ponte H



Fonte: Masugux (2017)

A ligação do módulo driver com Dupla Ponte- H - L298N pode ser conectado ao Arduino ligando-se os pinos de entrada as portas digitais do Arduino. Os bornes de motor A e motor B devem ser ligados aos respectivos motores. O borne central do módulo deve ser ligado a uma fonte externa, de acordo com a capacidade dos motores. O GND do módulo deve ser ligada no GND da fonte externa e o GND da fonte externa ligado ao GND do Arduino. Caso seja necessário controlar a velocidade do Arduino é necessário ligar os pinos ENA e ENB nas portas PWM do mesmo e fazer as devidas configurações de programação. A figura 23 representa o esquema de ligação dos motores no módulo.

Figura 20 - Esquema de ligação dos motores na *Shield*



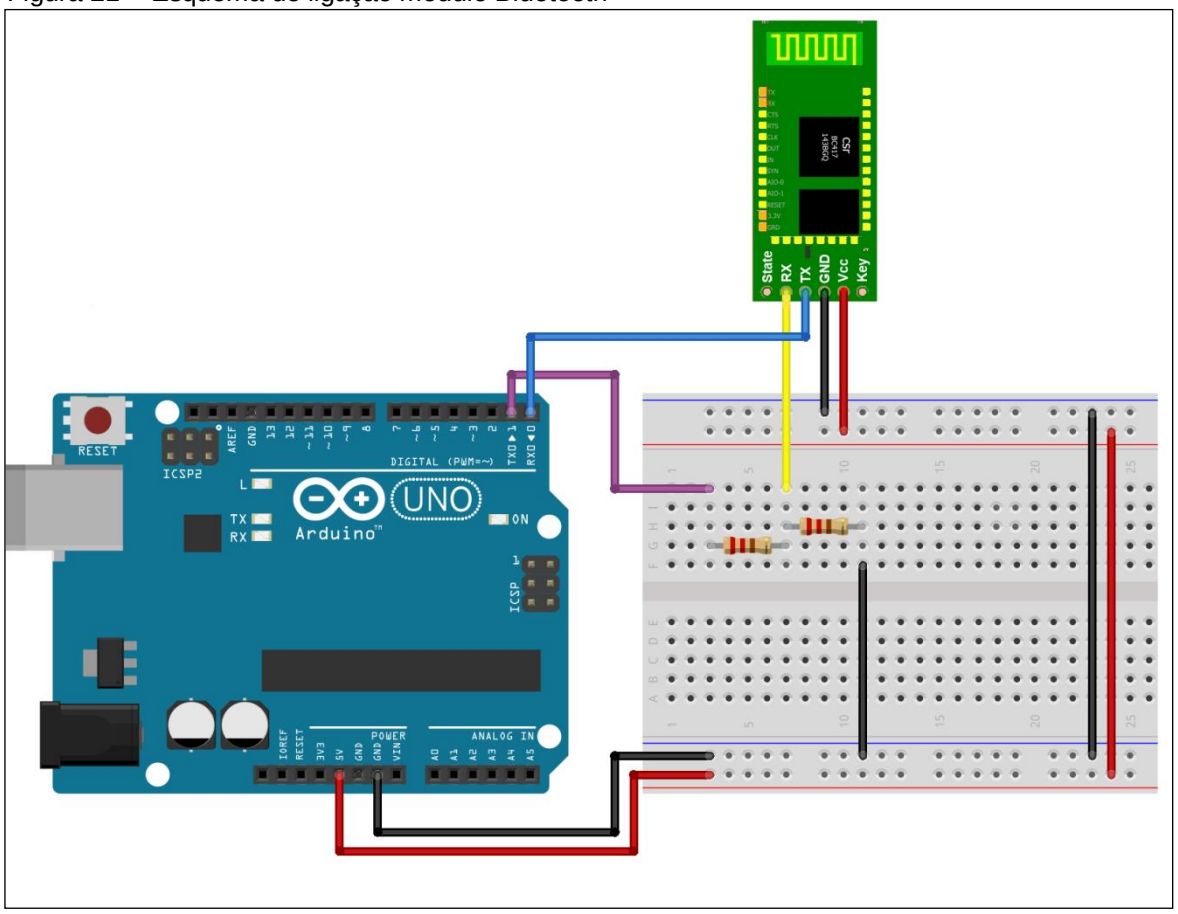
Fonte: Masugux (2017)

6.1.5 Implementação do Bluetooth

Bluetooth é um padrão de comunicação utilizado para trocar informações entre dispositivos eletrônicos compatíveis com essa tecnologia. A transmissão é feita através de uma frequência de rádio de curto alcance, licenciada globalmente.

O módulo utilizado no desenvolvimento do protótipo foi o modelo HC-05 que trabalha com uma frequência de 2.4Ghz, que se apresenta como uma opção simples e barata de realizar a comunicação via Bluetooth para o Arduino. Esse módulo possui a capacidade de trabalhar tanto em modo escravo quanto em modo mestre. Modo escravo é quando aceita o pareamento de outros dispositivos e modo mestre é capaz de ser pareado com outro dispositivo. Utilizando o módulo Bluetooth é possível realizar a comunicação serial entre o software desenvolvido com o robô autônomo, enviando informações das rotas para que o robô se locomova. Para realizar essa comunicação é necessário que o computador utilizado possua possibilidade de conexão Bluetooth. A figura 24 apresenta o esquema de ligação do Bluetooth.

Figura 21 - Esquema de ligação módulo Bluetooth



Fonte: FilipeFlop (2017)

Como pode ser observado na figura 24, o módulo Bluetooth pode ser alimentado por 5V, mas os pinos de RX/TX trabalham com a tensão média de 3.3V. Por esse motivo é necessário utilizar dois resistores como filtro de tensão para que a comunicação ocorra corretamente.

6.1.6 Controle do Robô seguidor de linha com Arduino

Para acionamento dos circuitos foi criado um programa em Arduino, esse programa tem a lógica que defini qual velocidade cada motor irá girar de acordo com as informações recebidas pelos sensores ópticos. Quando todos os sensores perderam a linha, foi implementada uma lógica para que o Robô consiga voltar a linha sem perder a sua direção.

6.2 SOFTWARE IMPLEMENTADO

A implementação do software foi realizada utilizando o ambiente de desenvolvimento integrado Code Blocks com a linguagem C++. O software se caracteriza pela comunicação via Bluetooth com o Arduino, enviando ao mesmo as orientações de quais rotas deve-se percorrer para ir de um ponto inicial até um ponto de destino. Foi utilizado o algoritmo de Dijkstra para determinar o caminho de menor custo. Nesse capítulo serão apresentados os módulos e as funcionalidades do software implementado. A figura 25 representa o menu principal do software.

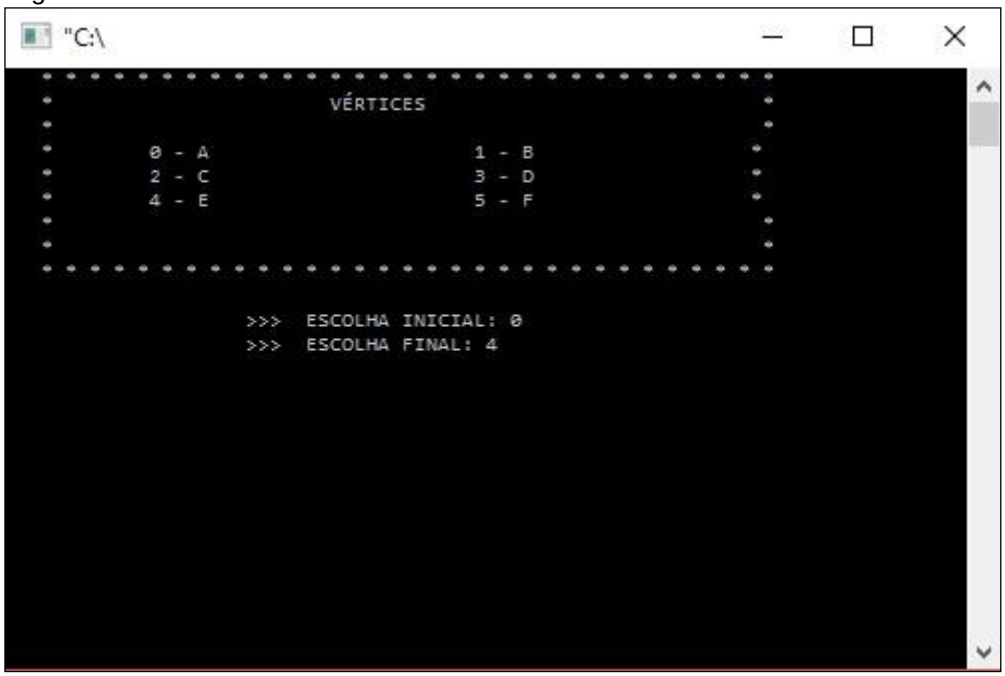
Figura 22 – Tela principal do software

Fonte: Do autor

6.2.1 Gerar rotas

Módulo responsável por gerar as rotas para o robô autônomo. O aluno poderá selecionar um ponto inicial e um ponto de destino, conforme é visto na figura 26. Com essas informações inseridas o software calcula qual a rota de menor caminho e envia ao Arduino.

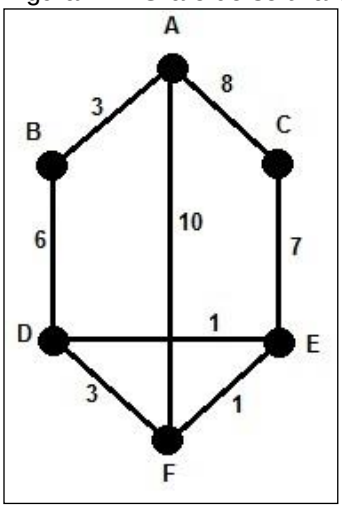
Figura 23 - Menu selecionar rota



Fonte: Do autor

O usuário pode selecionar entre os seis vértices um inicial e um de destino. Cada vértice possui uma ligação específica com outra e todas as ligação possuem um peso informado. A figura 27 representa o grafo inserido no Arduino.

Figura 24 - Grafo do software



Fonte: Do autor

Após selecionar o vértice inicial e o de destino o software apresenta os vértices que o carrinho irá percorrer para chegar ao destino e também apresenta o comprimento total do percurso realizado. A figura 28 ilustra esse exemplo.

Figura 25 - Resultado do processamento de caminho

Fonte: Do autor

6.2.2 Matriz de adjacência

Nesse módulo é representado a matriz de adjacência do grafo inserido no software. Os números representam os pesos das arestas de ligação entre os vértices conforme pode ser visto na figura 29. Para gerar a rota com caminho mínimo são considerados esses pesos.

Figura 26 - Matriz de adjacência

Fonte: Do autor

6.2.3 Controle robô

Módulo responsável para movimentar o robô na linha de acordo com a necessidade. As opções possíveis são ir para a linha em frente, seguir a linha pela direita ou seguir a linha pela esquerda. Utilizando essa opção o robô irá se mover até a próxima vértice encontrada e irá parar.

Figura 27 - Matriz de adjacência

Fonte: Do autor

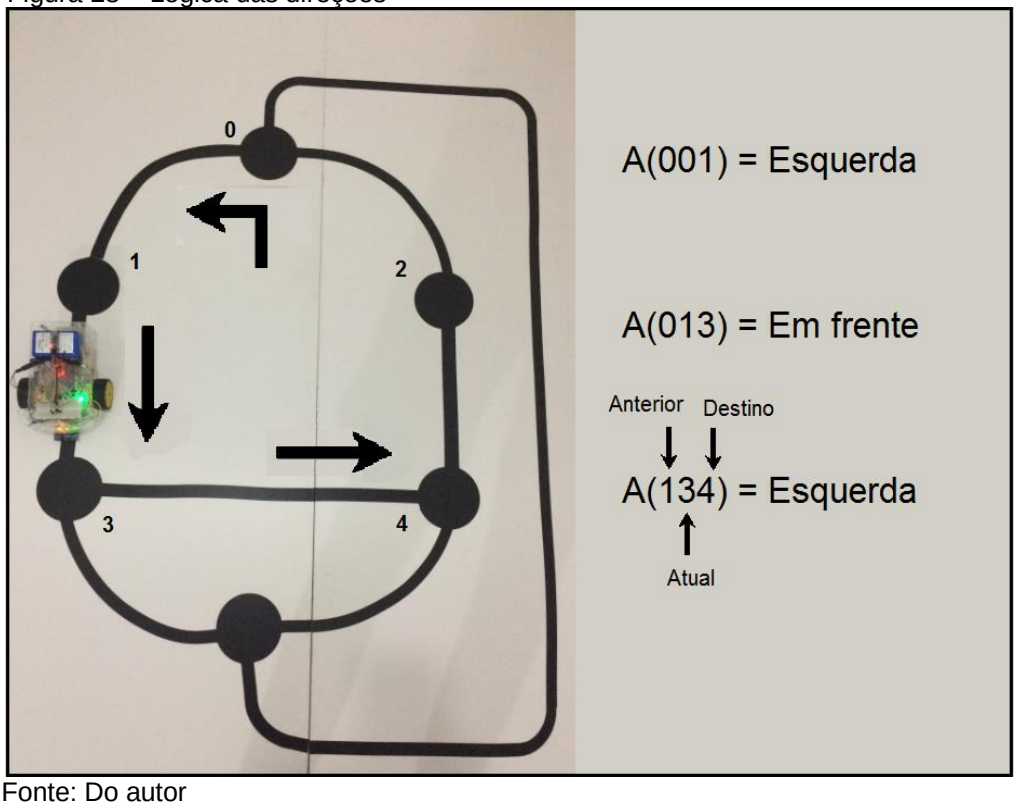
6.2.4 Direcionamento do robô

Para determinar as direções das rotas que o robô irá se locomover foi necessário criar uma estrutura de dados para armazenar essas informações. A estrutura utilizada para armazenamento foi a árvore B devido a facilidade de implementação e velocidade de acesso aos dados.

Árvore B é uma estrutura de dados projetada para funcionar especialmente em memória secundária. Dentre suas propriedades ela permite a inserção, remoção e busca de dados a partir de uma chave. A facilidade de acesso a informação utilizando árvore B foi que determinou a utilização dessa estrutura.

O funcionamento da lógica utilizada pode ser observado na figura 31.

Figura 28 – Lógica das direções



Fonte: Do autor

Conforme visto na figura 31 pode-se observar que o acesso a informação é feito através dos números de cada vértice, em que o primeiro número representa o vértice anterior, o segundo representa o vértice atual seguido do próximo vértice que o robô deve-se locomover. Um exemplo seria o robô se locomover do vértice ‘0’ até

o vértice ‘4’. Como o robô está partindo do vértice ’0’, ele não possui um vértice anterior, foi definido no software que essa situação deve ser informada com o mesmo código do vértice atual. Logo a busca na Árvore B seria feita através da chave 001, onde 1 representa o vértice de destino. Seguindo a tabela 1 que determina todas as direções que o robô pode se locomover, pode-se notar que o robô irá se locomover primeiramente para a esquerda. Essa informação é armazenada em um vetor para posteriormente ser enviada ao Arduino. Em seguida o código gerado será 013 que segundo a tabela 1 o robô seguirá em frente. Logo após o código gerado será o 134 e o robô irá virar à esquerda. Então o software verifica se o vértice ‘4’ é o vértice final que o aluno informou, e envia as informações do vetor para o Arduino através da comunicação Bluetooth.

Tabela 1 – Direções armazenadas na árvore

Chave	Direção
001	Esquerda
002	Direita
005	Frente
013	Frente
024	Frente
053	Esquerda
054	Direita
102	Frente
110	Frente
113	Frente
134	Esquerda
135	Frente
201	Frente
220	Frente
224	Frente
243	Direita
245	Frente
310	Frente
331	Esquerda
334	Frente
335	Direita
340	Frente
342	Esquerda
345	Direita
350	Direita
354	Frente
420	Frente
431	Direita
435	Esquerda
442	Direita
443	Frente
445	Esquerda
453	Frente
450	Esquerda
501	Direita
502	Esquerda
531	Frente
534	Direita
542	Frente
543	Esquerda
550	Frente
553	Direita
554	Esquerda

Fonte: Do autor

6.3 MONTAGEM ROBÔ SEGUIDOR DE LINHA

Foi efetuado o levantamento de componentes e ferramentas que eram necessárias para a implementação do trabalho proposto. Na tabela 2 são mostrados todos os itens adquiridos, listados com suas devidas quantidades e preços.

Tabela 2 - Preços dos componentes utilizados na montagem no robô

Descrição dos componentes	Valor unitário	Qtd.	Valor total
Arduino UNO	R$37,35	1	R$37,35
Placas acrílico	R$55,00	1	R$55,00
Roda com motor e redutor	R$19,90	2	R$39,80
Modulo L298N de motores	R$16,75	1	R$16,75
Sensor seguidor de trilha Tcrt5000	R$8,95	3	R$17,90
Suporte espaçador de metal	R$4,95	3	R$14,85
Protoboard 400 furos	R$12,35	1	R$12,35
Jumpers macho-femea 20cm	R$0,45	20	R$9,00
Jumpers macho-macho 10cm	R$0,20	20	R$4
Bateria 12V 4500mAh	R$45,00	1	R$45,00
Módulo Bluetooth HC-05	R$36,90	1	R$36,90
Resistor 10kΩ	R$0,04	1	R$0,04
Resistor 20kΩ	R$0,04	1	R$0,04
Total			R$289,98

Fonte: Do autor

Para o desenvolvimento do projeto e facilitar a montagem do robô autônomo foram necessárias a utilização de algumas ferramentas específicas. Para o desenvolvimento do robô foi utilizado um kit de carrinho seguidor de linha que é composto por dois acrílicos com furações especificas para fixar os componentes, também possui duas rodas de borracha fixadas a redutores, no qual o motor foi inserido. Para a ligação dos fios no motor foi necessário utilizar ferro de soldagem 60W. Os sensores foram fixados na parte da frente do motor, utilizando espaçadores de plástico com rosca dos dois lados, fixados com parafusos no acrílico. Foi utilizado multímetro para realizar a medições de tensão e corrente necessária para o bom funcionamento dos componentes eletrônicos. Para as ligações dos componentes eletrônicos foi utilizado jumpers macho-fêmea e macho-macho multicoloridos com secção do fio do condutor de 24AWG e comprimento de 20cm. Para a alimentação

do Arduino foi necessária uma bateria de 12VDC com 4500mAh. A bateria alimenta tanto o Arduino quanto a placa responsável por controlar a velocidade dos motores. Foi necessário dois plugs p2 para as ligações do Arduino e da bateria, também foram necessárias ferramentas para auxiliar na construção do robô como alicate de bico, conjuntos de chaves de fenda e chaves Philips. A imagem 32 mostra as ferramentas utilizadas na implementação do robô.

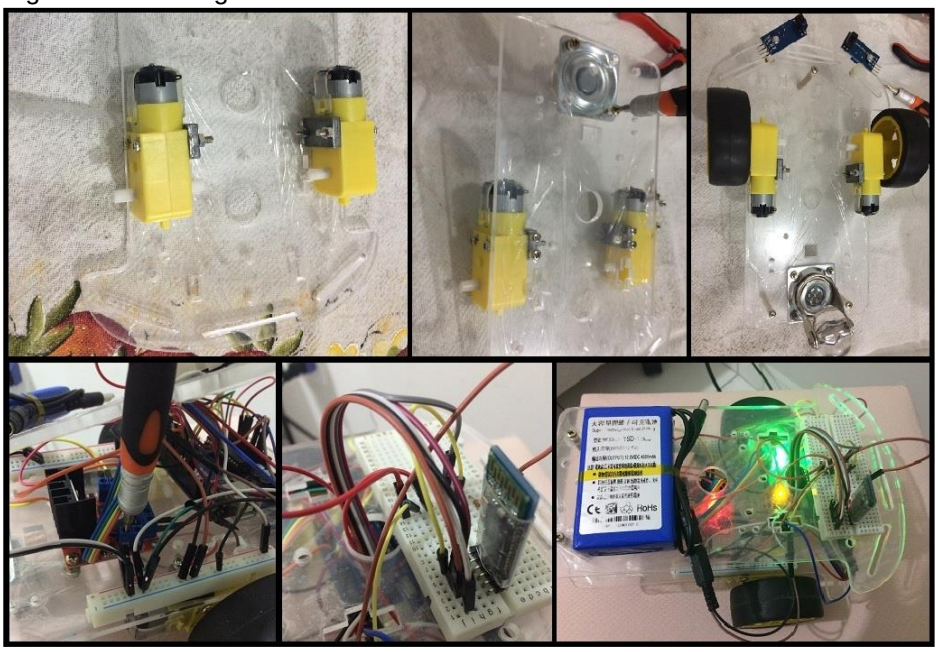
Figura 29 - Ferramentas utilizadas na implementação robô

Fonte: Do autor

Um dos problemas apresentados ao realizar testes com o robô seguidor de linha foi a bateria. Inicialmente foi utilizada uma bateria recarregável de 9v com 450 mAh, apenas para alimentador a placa controladora dos motores. Porém a mesma descarregava rapidamente, não sendo possível realizar muitos testes sem a necessidade de recarregá-la constantemente. Então foi substituída por uma bateria de 12VDC com 4500mAh.

Na figura 33 é mostrado algumas das etapas de montagem do protótipo.

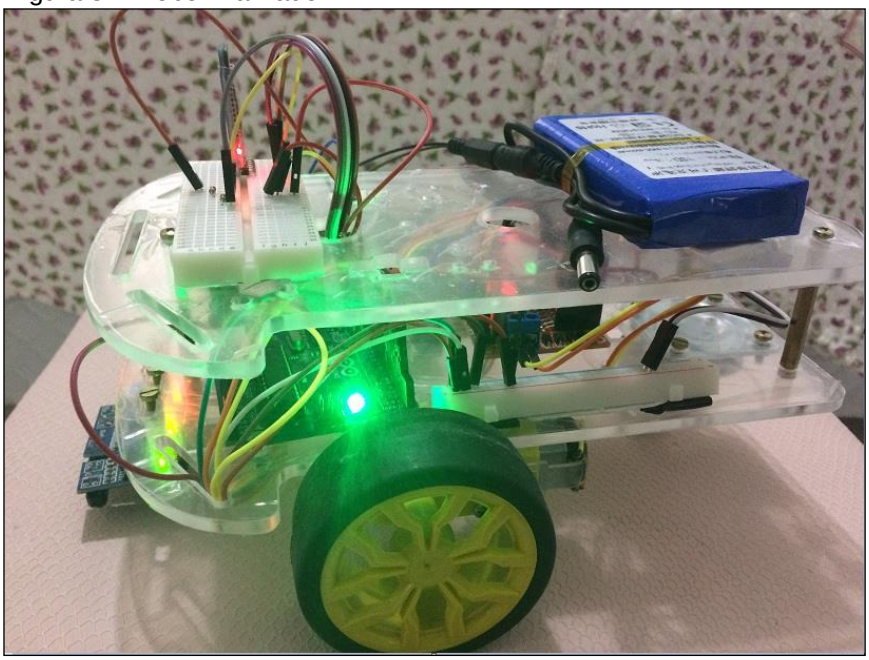
Figura 30 - Montagem robô autônomo



Fonte: Do autor

A montagem final do protótipo contendo todos os elementos descritos na metodologia é apresentado na figura 34.

Figura 31- Robô finalizado



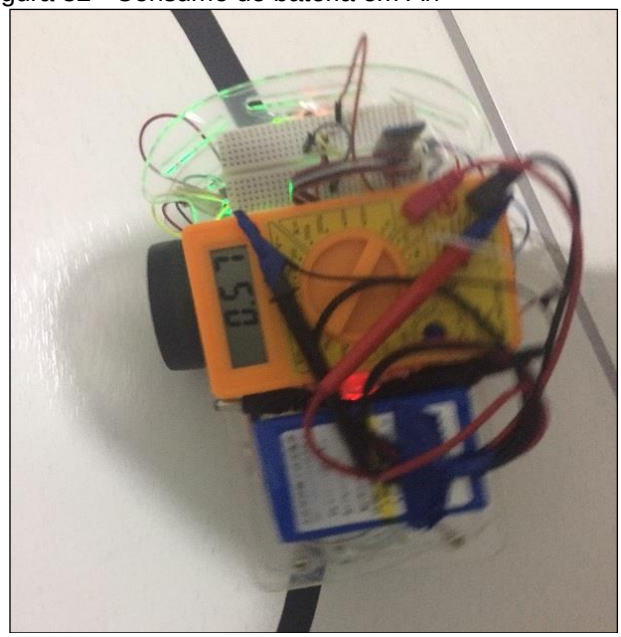
Fonte: Do autor

O software foi desenvolvimento em linguagem C++ utilizando a IDE Code Blocks. O computador utilizado foi um Sony Vaio com 8GB de memória RAM, HD de 512GB com processador Intel Core I5 2.50Ghz e sistema operacional Windows 10.

7 RESULTADOS E DISCUSSÕES

Os resultados obtidos foram resultados de testes realizados em um conjunto de diferentes larguras de linhas e cores do ambiente. Foi verificado que quando o piso possui alguns pontos com elevações, o sensor de linha acaba enviando ao Arduino valores incorretos de leitura, ocasionando problemas. Uma das soluções encontradas foi alterar a sensibilidade dos sensores de linha e outra foi o posicionamento do sensor mais distante do chão. Outro problema encontrado foi com relação a bateria, inicialmente foi utilizado uma bateria de 9V com 450mAh, porém ao realizar alguns testes foi visto que a bateria acabava descarregando muito rápido, sendo necessário realizar a recarga constantemente. Foi realizada a medição de consumo de carga do robô quando está em movimento, em que o consumo medido em média foi de 570mA conforme pode ser visto na Figura 35.

Figura 32 - Consumo de bateria em Ah



Fonte: Do autor

Para definir qual bateria utilizar no protótipo precisa-se saber a capacidade de energia que a bateria possui medida em miliampéres por hora cuja

sigla é mAh, mAh é o resultado de uma fórmula com 2 unidades, Corrente (mA) x hora tempo (h). Então com essas definições pode-se calcular qual duração de tempo de carga que as duas baterias utilizadas no protótipo possuem, utilizando a seguinte fórmula:

mAh (bateria)/ mA(Consumo carrinho) = Tempo de duração em horas

Utilizando a fórmula chega-se no resultado apresentado na tabela 3 e definiu-se que a bateria correta a ser utilizada no protótipo é a de 4500 mAh.

Tabela 3 - Tempo de duração da bateria de acordo com o consumo do robô

Descrição	Tempo de duração em horas
Bateria 9V 450mA	0.79 ou (47 minutos)
Bateria 12V 4500mA	7.89 ou (7 horas e 53 minutos)

Fonte: Do autor

Também realizou-se testes conforme a espessura da linha preta para o robô seguir. Com ela medindo aproximadamente 20mm o robô acabava perdendo muito a referência e andava de maneira incorreta, ocasionando perda de performance na velocidade do robô. Foi alterada a espessura da linha para 30mm e com isso o robô teve uma performance melhorada.

Os dados coletados foram analisados com auxílio do software IBM Statistical Package for the Social Sciencies (SPSS) versão 22.0. As variáveis quantitativas foram expressas por meio de média e desvio padrão.

Os testes estatísticos foram realizados com um nível de significância $\alpha = 0,05$ e, portanto, confiança de 95%. A distribuição dos dados quanto à normalidade foi avaliada por meio da aplicação do teste de Shapiro-Wilk. A investigação da variabilidade das variáveis quantitativas entre as categorias das variáveis qualitativas foi investigada por meio da aplicação do teste de Levene.

A comparação da média das variáveis quantitativas entre as categorias das variáveis qualitativas dicotômicas foi realizada por meio da aplicação do teste t de Student para amostras independentes.

O tempo médio que o dispositivo móvel levou para percorrer a trajetória em uma pista com arestas de 20 mm de largura foi de 13,07 ± 0,50 segundos e em uma pista com arestas medindo 30 mm de largura foi de 10,91 ± 0,28 segundos,

revelando que em uma pista cuja largura das arestas seja de 30 mm, o dispositivo apresenta um desempenho melhor quando comparado a essa mesma pista mas com arestas de 20 mm ($p < 0,001$).

Tabela 4 - Comparação do tempo de percurso do dispositivo móvel em um grafo de arestas medindo 20 mm e 30 mm de largura.

Largura da aresta	Tempo (segundos)	Valor – p*
20 mm	13,07 ± 0,50	< 0,001
30 mm	10,91 ± 0,28	

*Valor obtido após aplicação do teste t de Student.

A comunicação via Bluetooth do software desenvolvido na linguagem C++ com o Arduino foi realizada por meio da biblioteca SerialClass.h, essa biblioteca é utilizada para realizar a comunicação entre dois dispositivos a partir da comunicação serial. Quando é pareado o Bluetooth no computador ele cria uma porta serial para o dispositivo, depois basta informar essa porta serial no código fonte do software e a comunicação é feita. A figura 36 representa o código da comunicação Bluetooth.

Figura 33 - Codificação da comunicação Bluetooth

```
1   include "SerialClass.h"
2   Serial* Arduino;
3   int main(){
4
5   Arduino = new Serial("COM4");//Comunicação com o bluetooth do Arduino
6
7   EnviaArduino(saidaDirecao);
8       }
9
10
11  void EnviaArduino(string Caminhos)
12      {
13
14  char *Caminho = new char[Caminhos.length()+1];
15      memcpy(Caminho, Caminhos.c_str(), Caminhos.length() + 1);
16
17  if (Arduino->IsConnected() )
18  {
19      //cout<<"Arduino conectado"<<endl;
20      //cout<<"Enviando : "<<Caminho<<endl;
21
22  Arduino->WriteData(Caminho,sizeof(Caminho)-1);
23  }
24  }
25
```

Fonte: Do autor

Pode-se notar que na linha 5 é declarado o novo serial passando a porta COM do módulo Bluetooth que foi instalado no computador, essa entrada serial pode-se variar de acordo com o computador. Na linha 22 é chamado a função

responsável por enviar os dados gerados nos algoritmos de Dijkstra e enviado ao robô.

O software desenvolvido proporciona ao aluno uma interação de teoria dos grafos e algoritmo de Dijkstra, em que define o menor caminho e em seguida o envia ao Arduino, com o objetivo de enviar ao Robô qual direção deve percorrer para sair de um ponto inicial até um ponto de destino percorrendo o menor caminho possível. Esse tipo de interação faz com que estimule o aluno ao aprendizado de grafos e estimule o mesmo a busca conhecimentos nessa área colocando seu conhecimento em prática. O trabalho desenvolvido por Silva (2009) com título de RoboEduc por exemplo conclui que a robótica necessita de pesquisas não só em questões técnicas, como também metodológicas. Neste trabalho foi utilizado a robótica como forma de estímulo ao aluno para a compreensão de como pode ser empregado na prática a utilização de grafos.

Correa (2013) desenvolveu um robô para navegação autônoma e detecção de intrusos em ambientes internos. Ele descreve que utilizou sensores 2D e 3D tanto para vigilância quanto para a detecção do intruso. O objetivo desse trabalho é de um sistema de percepção do ambiente visando a navegação de robôs móveis autônomos em ambientes internos, para a execução de tarefas de monitoramento e vigilância de ambientes fechados. Comparando ao trabalho desenvolvido pode-se notar que o robô implementado por Correa tem a capacidade de locomover-se em áreas abertas sem a necessidade de uma linha para seguir como no presente trabalho.

8 CONCLUSÃO

Neste trabalho foram apresentados todos os componentes necessários para o desenvolvimento de um robô seguidor de linha, cada parte estrutural que contém no projeto foi devidamente ajustada, contribuindo para o bom funcionamento do protótipo. Além disso através de conhecimentos adquiridos em comunicação de dados foi desenvolvido a comunicação do software com o Arduino para transferência de dados através da tecnologia Bluetooth contribuindo para atingir a meta da implementação.

A estrutura desenvolvida para o robô autônomo apresentou bons resultados e cumpriu seu propósito, atuando como um estímulo de aprendizado aos alunos de algoritmos de grafos. A partir de estudos realizados em teoria de grafos, o software desenvolvido atua como um emissor de comandos e de acordo com as rotas definida pelo aluno, o software o envia ao robô fazendo-lhe mover conforme o menor caminho gerado pelo algoritmo de Dijkstra.

Foi adquirido o conhecimento de que, para ter-se o controle correto do robô sobre a linha, precisa-se ser capaz de efetuar a leitura dos sensores e a calibração do mesmo. Em alguns casos o sensor realizava leituras incorretas e foi necessário calibrar os mesmos melhorando a sensibilidade evitando problema de leitura.

Um dos desafios encontrados no desenvolvimento do robô consiste na implementação da lógica para que carrinho volte a linha quando todos os sensores perdem a leitura da mesma. A solução encontrada foi gravar em uma variável auxiliar qual o último sensor que identificou a linha. Com essa informação é possível saber em qual direção o carrinho estava no momento que perdeu a leitura da linha e com isso é possível direcionar o robô para o sentido contrário até que os sensores capturem a linha novamente.

Os resultados finais mostram que todas as metas da etapa do desenvolvimento do projeto foram cumpridas. Várias alterações de projeto foram feitas na tentativa de criar um robô seguidor de linha adequado para os objetivos do projeto. Seguindo as etapas metodológicas foi possível realizar a movimentação do robô autônomo sobre um plano corretamente.

Com base em estudos de teoria dos grafos, foi possível o desenvolvimento do software que utiliza o algoritmo de Dijkstra para determinar o menor caminho entre duas vértices. Para o desenvolvimento do software também foi necessário estudar técnicas de estruturas de dados, onde foi aplicado o conceito de Árvore B para armazenamento das informações que determina as direções que o carrinho irá se locomover.

Deseja-se de que este trabalho possa posteriormente servir como uma base de pesquisa aos acadêmicos interessados no tema. No geral, foi construído um software que utiliza algoritmo de Dijkstra para determinar a trajetória do robô desenvolvido em Arduino sobre uma rota planejada.

Devido ao tempo limitado não foi implementada a parte visual do software. Implementar uma interface para proporcionar uma melhor experiência ao aluno, pode ser dado como um dos trabalhos futuros. Outra ideia é implementar um robô seguidor de linha que percorra uma determinada área e calcule todas as distâncias de cada caminho, enviando as informações coletadas para o software que posteriormente pode calcular a menor rota de um ponto inicial até um ponto final. Um dos problemas do trabalho desenvolvido foi encontrar um meio de saber qual direção o software deveria direcionar para ir de um vértice inicial até um vértice de destino. No trabalho foi implementada uma lógica que armazenava todas as direções possíveis de um vértice a outro. Um exemplo de trabalho futuro seria utilizar sensores que detectam a cor para auxiliar na decisão de qual direção o robô deve direcionar para chegar a um vértice de destino.

.

REFERÊNCIAS

ARDUINO.CC. **Arduino** Disponível em: <http://arduino.cc/>. Acesso em: 20 de novembro 2016.

AHUJA, R., Magnanti, T., Orlin, J**., Network Flows Theory: algorithms and applications**,1993.

BARRAQUAND, J., Kavraki, L., Latombe, J. C. AND LI, T. **A random sampling scheme for path planning**, The International Journal of Robotics Research 16: 759–774. 1997.

BEKEY, G. A. **Autonomous Robots: From Biological Inspiration to Implementation and Control**. The MIT Press: Cambrigde, London. 563p. 2005.

BERG, J. P. AND Overmars, M. H. (2004**). Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners**, IEEE Internation Conference on Robotics and Automation pp. 453–460.

BERNARDI, S. T. **Utilização de Softwares Educacionais nos Processos de Alfabetização, de Ensino e Aprendizagem com uma Visão Psicopedagógica.** Revista de Educação do Ideau (REI), v.5,nº10. 2010.

BOAVENTURA NETTO, Paulo Oswaldo; JURKIEWICZ, Samuel. **Grafos: introdução e prática.** São Paulo: Blucher, 2009.

BOTTI, N. C. L. **Desenvolvimento e validação de software educativo de saúde mental.** *Revista Mineira de Enfermagem* v18.1, p. 218-223. 2014.

BRIOT, M.; Talou J.C.; Bauzil G et al. **Le Systeme de Perception du Robot HILARE.** 2eme Congress AFCET/IRIA, Toulouse, France. 1979.

BUZIN, P. F. W. K. (2001). **A epistemologia da Ciência da Computação: Desafio do Ensino dessa Ciência**, Revista de Educação, Ciência e Cultura, v. 6, nº 2. Centro Universitário La Salle. Canoas, RS, Brasil.

CAMARGO, LUIS GUILHERME MACHADO; LOPES, MARCELO TEIDER; ARAUJO, MATHEUS SILVA. **Robô Explorador de Ambientes.** Monografia. Curitiba, 2011.

CORREA, Diogo Santos Ortiz. **Navegação autônoma de robôs móveis e detecção de intrusos em ambientes internos utilizando sensores 2D e 3D.** Tese de Doutorado. Universidade de São Paulo. 2013.

COUTINHO, C. e Lisbôa, E. **Sociedade da informação, do conhecimento e da aprendizagem: desafios para educação no século XXI**. Revista de Educação, Vol XVIII, nº 1, p. 5-22.2011.

CRAIG, John J. **Robótica.** 3ª ed. São Paulo: Pearson Education do Brasil, 2012.

CUNHA, C. B. **Uma contribuição para o problema de roteirização de veículos com restrições operacionais.** São Paulo: EPUSP, Departamento de Engenharia de Transportes. 222p. (Tese de Doutoramento), 1997.

DUDEK, G.; Jenkin, M. **Computational Principles of Mobile Robotics**. Cambridge, London, UK: The MIT Press, 280 p.2000.

FIGUEIREDO, Tenorio R., Figueiredo C. B. **WarGrafos–Jogo para Auxilio na Aprendizagem da Disciplina de Teoria dos Grafos.** X Simpósio Brasileiro de Games e Entretenimento Digital (SBGames), 2011.

FILIPEFLOP, < http://blog.filipeflop.com> Acesso em: 16 de fevereiro de 2017.

GOLBARG, M. C. e H. P. R. Luna **Otimização Combinatória e Programação Linear.** Rio de Janeiro: Editora Campus, 1999.

GOODRICH, Michael T.; TAMASSIA, Roberto. **Estrutura de dados e algoritmos em Java**. 4. ed. Porto Alegre: Bookman, 2007.

HADEN, P.; Mann, S. (2003). **The Trouble with Teaching Programming**, In: Proc. of the 16th Annual NACCQ, Palmerston North, New Zealand, p. 63-70

HELSGAUN, K. **An effective implementation of the Lin-Kernigham Traveling Salesman Heuristic, European Journal of Operational Research**, v.126, p.106-130, 2000.

JUCÁ, C. S. **A relevância dos softwares educativos na educação profissional.** *Ciências & Cognição* 8. p. 22-28. 2006.

LENCASTRE, José A. **Educação On-line: um estudo sobre o blended learning na formação pós graduada a partir da experiência de desenho, desenvolvimento e implementação de um protótipo Web sobre a Imagem.** Tese de Doutoramento. Braga: Universidade do Minho. 2009.

LENSTRA, J. K. & RINNOOY KAN, A. H. G. **Some Simple Applications of the Traveling Salesman Problem. Operational Research Quarterly**, v. 26, n. 4, p. 717-733, 1975.

LIMA, A. De. Controle de formação e rastreamento de trajetória para robôs móveis utilizando funções potenciais, Tese de Doutorado. Universidade Federal do Rio de Janeiro, 2014

MATOS, I. M. D. **Teoria dos grafos no ensino básico e secundário**. Dissertação de Mestrado, Universidade de Aveiro, 2013.

MAUGUX, < http://www.masugux.com.br> Acesso em: 15 de fevereiro de 2017.

MCROBERTS, M. **Arduino Básico.** São Paulo: Novatec. 2011.

MERCADO, L. P. L. **Novas Tecnologias na Educação: Reflexões Sobre a Prática.** Maceió: EDUFAL, 2002.

MÉNDEZ, Y.; GUARDIA, L. **Problema do caminho mais curto-Algoritmo de Dijkstra**. São Domingos, 2008.

MURPHY, R. R**. Introductionto ai robotics.** Cambridge: The Mit Press, 2000.

NILSSON, N. J. **A Mobile Automaton An Application of Artificial Intelligence Techniques. Autonomous Mobile Robots Control**, Planning and Architecture, p. 233–239, 1969.

OLIVEIRA, J. R. de. **Um sistema integrado para navegaçao autônoma de robôs móveis**. Tese de Doutorado, Universidade de São Paulo, 2010.

PIO, J. L.. **A Robótica Móvel como Instrumento de Apoio à Aprendizagem de Computação.SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, XVII**, Brasília. Anais do Simpósio Brasileiro de Informática na Educação. SBIE: SBC.p. 8 - 10. 2006.

REDEL, R.; HOUNSELL, M. da S. Implementação de Simuladores de Robôs com o Uso da Tecnologia de Realidade Virtual. In: **IV Congresso Brasileiro de Computação, Itajaí–SC. IV CBCOMP**. p. 398-401. 2004.

RESNICK, Mitchel et al. Digital manipulatives: new toys to think with. In: **Proceedings of the SIGCHI conference on Human factors in computing systems**. ACM Press/Addison-Wesley Publishing Co., p. 281-287. 1998.

ROSÁRIO, J. M. **Princípios de mecatrônica.** São Paulo: Pearson, 2007.

SANCHO, J.M. ***Para uma Tecnologia educacional*.** Porto Alegre: ArtMed. 1998.

SANTOS, J. Plínio O., Mello, M. P. Murari, I.T.C. **Introdução à Análise Combinatória**, Editora da UNICAMP, 1995.

SANTOS, Pereira R., Costa H. A. X. **TBCGRAFOS/WEB–Treinamento Baseado em Computador para Algoritmos em Grafos Via Web,** 2006.

SANTOS, Rodrigues P., Heitor C. A. X. **Tbcaed: Um software gráfico para apresentação de algoritmos e estruturas de dados aos iniciantes em computação e informática,** 2005.

SILVA, Alzira Ferreira da. **RoboEduc: Uma metodologia de aprendizado com Robótica Educacional.** 2009.

SILVA, Fernandes D.; SANCHES, Leme A., **Aplicação conjunta do método de Dijkstra e otimização combinatória para solução do problema do caixeiro viajante**. SegeTSimpósio de Excelência em Gestão e Tecnologia, 2009.

SILVEIRA. **Avaliação dos software educativos na educação profissional.** Centro Federal de educação Tecnológica do Ceará - CEFET-CE. 2006.

SOARES, T. C. A. P., Cordeiro E. S., Stefani Í. G. A., Tirelo, F. **Uma Proposta Metodológica para o Aprendizado de Algoritmos em Grafos Via Animação Não Intrusiva de Algoritmos**, III Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG). Belo Horizonte, MG, Brasil, 2004.

VALENTE, José Armando. **Análise dos diferentes tipos de softwares usados na educação.** *O computador na sociedade do conhecimento* . p. 71. 1998.

VIEIRA, Fábia Magali Santos. **Avaliação de software educativo: reflexões para uma análise criteriosa.** Disponível em: <http://edutec.net/Textos/Alia/MISC/edmagali2.htm>. Acesso em 25 de novembro de 2016, v.14, 1999.

WALTER W. G.. **An Imitation of Life.** Scientific American, 182(5). p.42-45. 1950.

WOLF, D. F. **Robótica móvel inteligente: Da simulação às aplicações no mundo real.** Mini-Curso: Jornada de Atualização em Informática (JAI), Congresso da SBC. p. 13. 2009.

ZAMBONI, L. C.; PAMBOUKIAN, S. V. D.; BARROS, E. A. R., **C++ para Universitários**. São Paulo: Páginas & Letras, 2006.

APÊNDICE A – ARTIGO CIENTÍFICO

UTILIZAÇÃO DE ALGORITIMO DE DIJKSTRA PARA PLANEJAMENTO DE TRAJETÓRIA DE UM PROTÓTIPO DE ROBÔ AUTÔNOMO DESENVOLVIDO EM ARDUINO

Eduardo Rebelo[1], Sergio Coral[2], Kristian Mandeira[2]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologia - Universidade do Extremo Sul Catarinense (UNESC) – Criciúma – SC – Brasil

[2]Professor do Curso de Ciência da Computação– Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) –Criciúma – SC– Brasil

dudu_rbl@hotmail.com[1], sergiocoral@unesc.net[2], kma@unesc.net[2]

Resumo. *Neste trabalho é apresentada a utilização de um software, que através da utilização do algoritmo de Dijkstra determina a trajetória de custo mínimo, partindo de um vértice inicial até um vértice de destino. Essas rotas são enviadas a um robô autônomo através de comunicação Bluetooth, com a qual é possível realizar, a movimentação do mesmo através de uma ilustração física do grafo utilizado. O robô tem o objetivo de realizar o deslocamento através de linhas pretas na qual representam as arestas do grafo e são detectadas através de sensores de linha. O processamento do algoritmo define a proposta do trabalho, a qual tem por objetivo determinar o trajeto de menor caminho entre um ponto de origem e de destino realizando o deslocamento do robô através do caminho gerado. Também são utilizas técnicas de estruturas de dados, nas quais foram aplicados os conceitos de Árvore B para armazenamento de informações que determinam as direções do robô seguidor de linha. A solução encontrara para realizar a comunicação Bluetooth foi utilizar a biblioteca SerialClass.h, sendo ela utilizada devido a sua compatibilidade de realizar a comunicação através das portas seriais. Esse trabalho também tem como proposta a atuação como um estímulo de aprendizado de algoritmos de grafos através da utilização da robótica.*

Abstract. *In this paper is presented the use of a software, which, through the use of Dijkstra algorithm, determines the minimum cost trajectory, starting from an initial vertex to a destination vertex. These routes are sent to the autonomous robot through Bluetooth communication, with which it is possible to carry out the movement of the same through an illustration of the graph used. The robot has the objective to carry out the displacement through the black lines in which they represent the edges of the graph and are detected through line sensors. The algorithm processing define the work proposal, which aims to determine the trajectory of the smallest path between a point of origin and destination by performing the displacement of the robot through the created path. The work also uses techniques of data structures, in which the concept of Tree B was applied to store information that determine the directions of the row follower car. The solution found to perform the Bluetooth communication was to use the SerialClass.h library, being used because of its compatibility of performing the communication through the serial doors. The proposed work has the objective of acting as a stimulus for algorithms learning of graphs through the use of robotics.*

1. Introdução

Uma linha de pesquisa, visando melhorar o processo de estudo é direcionada a utilização de ferramentas e software computacionais como ambiente de estudo. A partir de observações em aulas de graduação, percebe-se um melhor resultado de aprendizado por meio de atividades práticas de desenvolvimento utilizando ferramentas e simuladores didáticos de representação de conceitos abstratos (SANTOS; RODRIGUES; HEITOR, 2005).

Existem problemas computacionais interessantes em termos de grafos. Entretanto, parte dos alunos enfrentam dificuldades na implementação de algoritmos em grafos, devido a necessidade de suporte para implementação. Implementações essas que são compostas por estruturas de dados, ferramentas de visualização e iteração com grafos, diferença de níveis de abstração entre as definições teóricas dos algoritmos e representações computacionais necessárias para implementá-los (SOARES, 2004)

Um deles é o problema de menor caminho, ele é utilizado em muitas aplicações, como por exemplo, em telecomunicações, transporte, correios, entre outros, na qual seja necessário encontrar um caminho de menor custo, ou mais rápido, entre dois pontos. Um dos algoritmos que solucionam esse problema, é o algoritmo de Dijkstra. Este algoritmo funciona apenas em grafos que os pesos das arestas não forem negativos. Ele é aplicado em grafos simples que tem o propósito de determinar o caminho de menor custo entre dois vértices. Quando o grafo não é simples, então o algoritmo procura transformá-lo em um grafo simples eliminando seus laços e eventuais arestas múltiplas deixando apenas com o peso menor (SILVA; FERNANDES; SANCHES ,2009).

A partir disso, o presente trabalho tem por objetivo mostrar ao aluno a utilização do algoritmo de Dijkstra como forma de determinar a rota de caminho mínimo, em que através das rotas geradas, realize a movimentação do robô autônomo a partir de um ponto de origem até um ponto de destino.

2. Problema do menor caminho

O problema de menor caminho é muito conhecido da teoria de grafos, ele é utilizado em muitas aplicações, por exemplo, em telecomunicações, transporte, correios, entre outros, onde seja necessário encontrar um caminho de menor custo, ou mais rápido, entre dois pontos (SILVA; FERNANDES; SANCHES ,2009).

Esse problema consiste em encontrar o menor caminho entre uma origem e um destino, dentro de uma rede, minimizando o custo de travessia de um grafo, entre dois pontos, esses custos são dado pela soma dos pesos de cada aresta percorrida (SILVA; FERNANDES; SANCHES ,2009).

Os algoritmos responsáveis em solucionar o problema do menor caminho são normalmente chamados de "algoritmos de busca de caminhos". O algoritmo que mais se destaca pela sua simplicidade é o algoritmo de Dijkstra (SILVA; FERNANDES; SANCHES ,2009).

3. Algoritmo de Dijkstra

O algoritmo de Dijkstra, foi desenvolvido em 1959 por Edsger Wybe Dijkstra, por isso foi dado esse nome ao algoritmo. Este algoritmo funciona apenas em grafos que os pesos das arestas não forem negativos. Ele é aplicado em grafos simples que tem o propósito de determinar o caminho de menor custo entre dois vértices. Quando o grafo não é simples, então o algoritmo procura transformá-lo em um grafo simples eliminando seus laços e eventuais

arestas múltiplas deixando apenas com o peso menor. A complexidade do Algoritmo de Dijkstra é O (n^2), onde n é a quantidade de vértices do grafo (ZAMBONI, 2006).

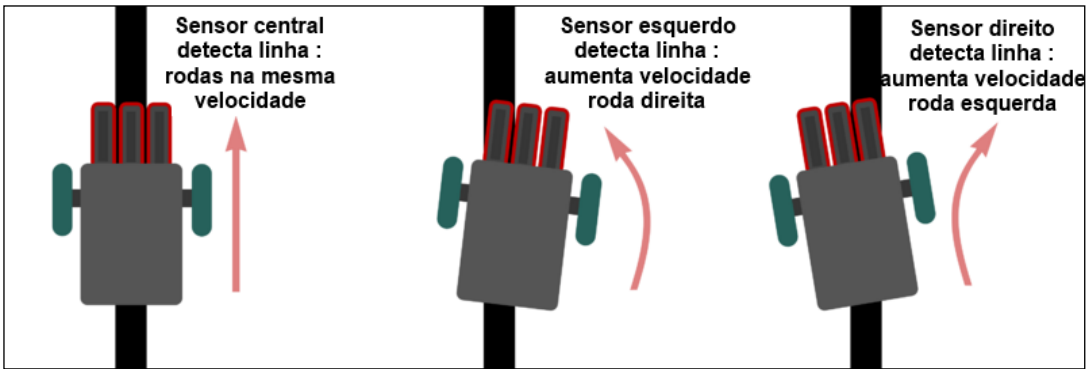
Segundo Méndez e Gardia (2008) o algoritmo de Dijkstra encontra o menor caminho de um nó fonte até um nó de origem de uma rede orientada para o caso em que todos os pesos dos arcos sejam não negativos. O algoritmo grava a distância rotulada para cada nó inicial e o limite superior do caminho mais curto até o nó de origem. Os nós são divididos em dois grupos: os rotulados permanentemente e rotulados temporariamente. A distância rotulada dos nós permanentes representa a menor distância do nó fonte até um outro nó. No caso dos nós temporários, esta distância representa o limite superior da distância do caminho mais curto até o nó.

4. Robô seguidor de linha

Um bom exemplo de utilização de hardware utilizando Arduino é em aplicações de robótica, como por exemplo em montagem de um robô seguidor de linha. O robô possui sensores ópticos ligados lado a lado. Conforme a detecção de linha, cada sensor enviará ao Arduino as informações sobre a intensidade do sinal infravermelho refletido e o programa usará essa informação para calcular a velocidade de cada motor.

4.1. Implementação sensores seguidor de linha

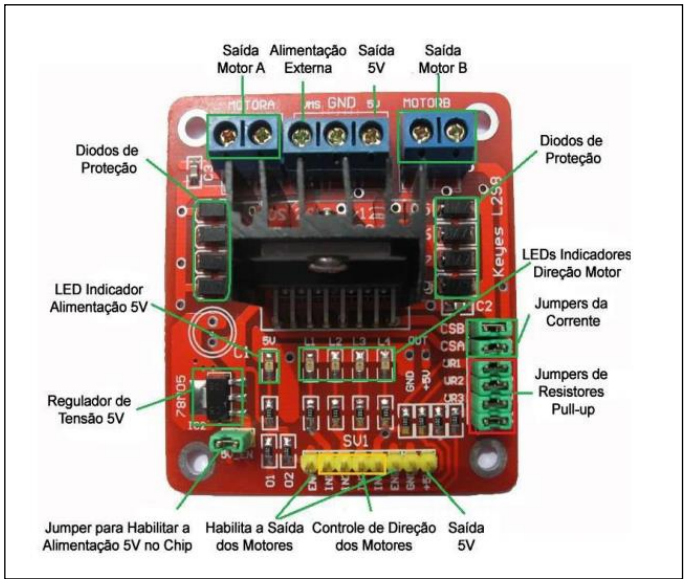
O robô possui sensores ópticos ligados lado a lado. Conforme a detecção de linha, cada sensor enviará ao Arduino as informações sobre a intensidade do sinal infravermelho refletido e o programa usará essa informação para calcular a velocidade de cada motor. A figura 1 mostra, de forma resumida, como os sensores se comportam.



2 Figure 1. Ilustração sensores ópticos

4.2. Shield L298N para controlar os motores

Para o controle de velocidade dos motores foi utilizado o módulo driver motor com dupla ponte-H baseado no chip L298N. Esse módulo tem capacidade de controlar a velocidade e o sentido de giro de até dois motores de 3 a 30V DC. A figura 2 ilustra a *shild*.

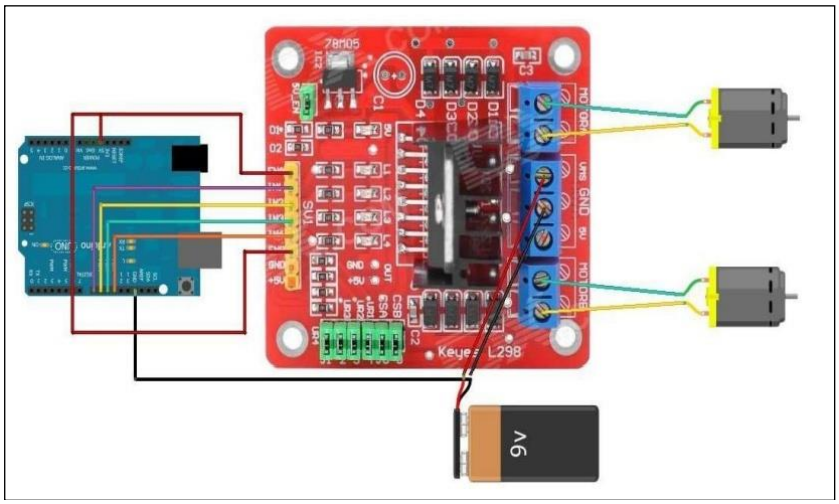


3 Figure 2. Módulo driver motor com dupla ponte – H – L298N

4.3. Ligação dos motores DC

O motor de Corrente Contínua (DC) usa os princípios do magnetismo para girar. O motor de corrente contínua por padrão tem dois imãs em torno de uma grade bobina de fios enrolada em forma de espiral em um rotor ou induzindo. Quando a corrente elétrica flui através do fio em espiral, essa bobina cria um campo magnético que interage com o campo do imã produzindo uma força que faz o rotor girar.

Para obter a rotação dos motores em ambos os sentidos, é necessário alterar o sentido da circulação da corrente aplicada ao motor, ou seja, basta alimentar as bobinas com Vcc e GND. Para fazê-lo girar no sentido contrário, basta inverter o Vcc com o GND. Para facilitar a inversão de rotação dos motores é indicado utilizar uma Ponte H para não ser necessário a utilização de recursos mecânicos como chaves e relés. A figura 3 representa o esquema de ligação dos motores no módulo.



4 Figure 3. Esquema de ligação dos motores na **Shield**

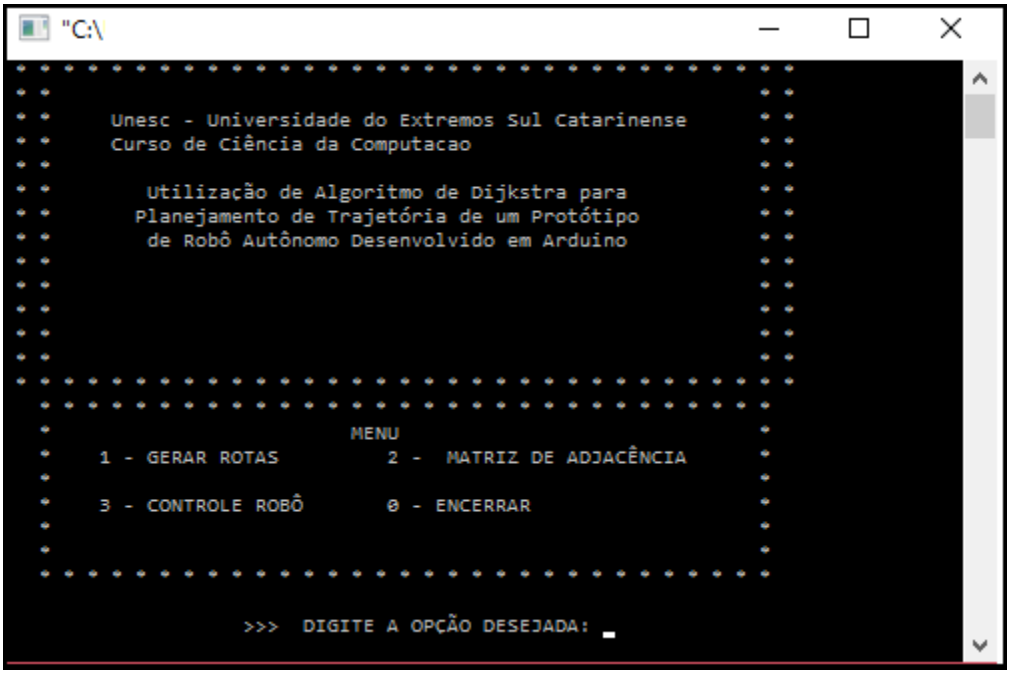
4.4. Implementação do Bluetooth

Bluetooth é um padrão de comunicação utilizado para trocar informações entre dispositivos eletrônicos compatíveis com essa tecnologia. A transmissão é feita através de uma frequência de rádio de curto alcance, licenciada globalmente.

O módulo utilizado no desenvolvimento do protótipo foi o modelo HC-05 que trabalha com uma frequência de 2.4Ghz, que se apresenta como uma opção simples e barata de realizar a comunicação via Bluetooth para o Arduino. Esse módulo possui a capacidade de trabalhar tanto em modo escravo quanto em modo mestre. Modo escravo é quando aceita o pareamento de outros dispositivos e modo mestre é capaz de ser pareado com outro dispositivo. Utilizando o módulo Bluetooth é possível realizar a comunicação serial entre o software desenvolvido com o robô autônomo, enviando informações das rotas para que o robô se locomova. Para realizar essa comunicação é necessário que o computador utilizado possua possibilidade de conexão Bluetooth.

5. Software implementado

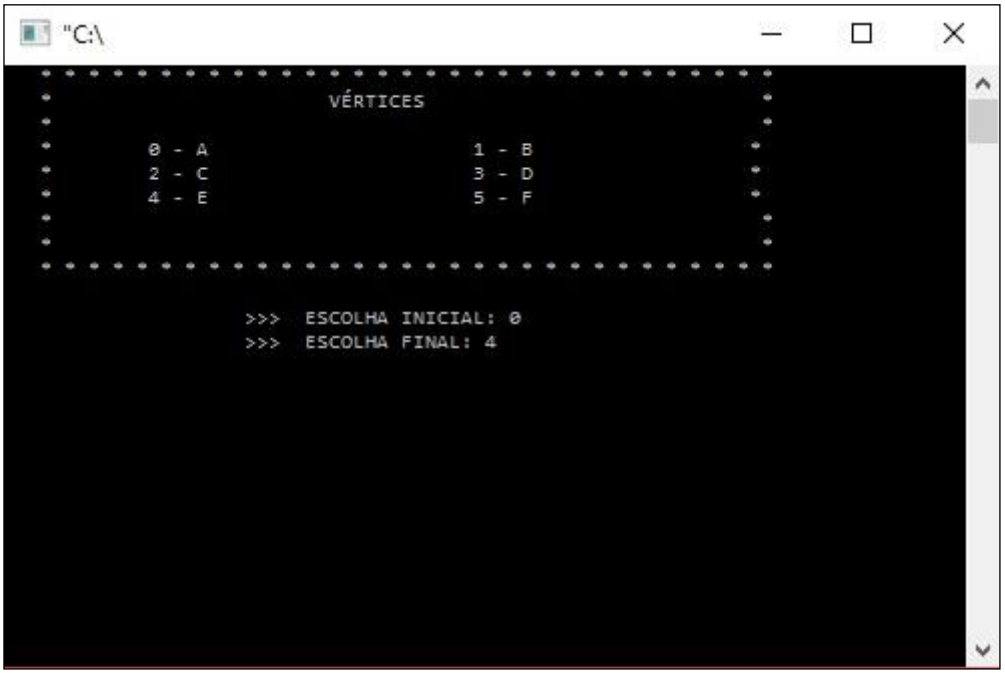
A implementação do software foi realizada utilizando o ambiente de desenvolvimento integrado Code Blocks com a linguagem C++. O software se caracteriza pela comunicação via Bluetooth com o Arduino, enviando ao mesmo as orientações de quais rotas deve-se percorrer para ir de um ponto inicial até um ponto de destino. Foi utilizado o algoritmo de Dijkstra para determinar o caminho de menor custo. A figura 4 representa o menu principal do software.



5 Figure 4. Tela principal do software

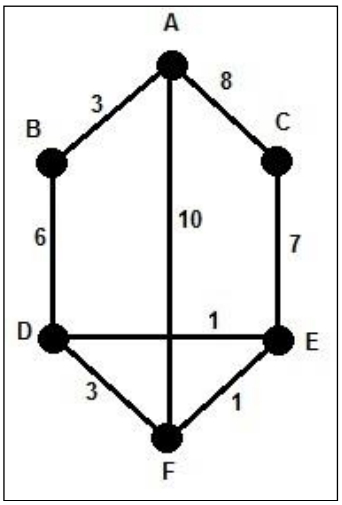
5.1. Gerar rotas

Módulo responsável por gerar as rotas para o robô autônomo. O aluno poderá selecionar um ponto inicial e um ponto de destino, conforme é visto na figura 5. Com essas informações inseridas, o software calcula qual a rota de menor caminho e envia ao Arduino.



6 Figure 5. Menu gerar rotas

O usuário pode selecionar entre os seis vértices um inicial e um de destino. Cada vértice possui uma ligação específica com outra e todas as ligação possuem um peso informado. A figura 6 representa o grafo inserido no Arduino.



7 Figure 6. Grafo do software

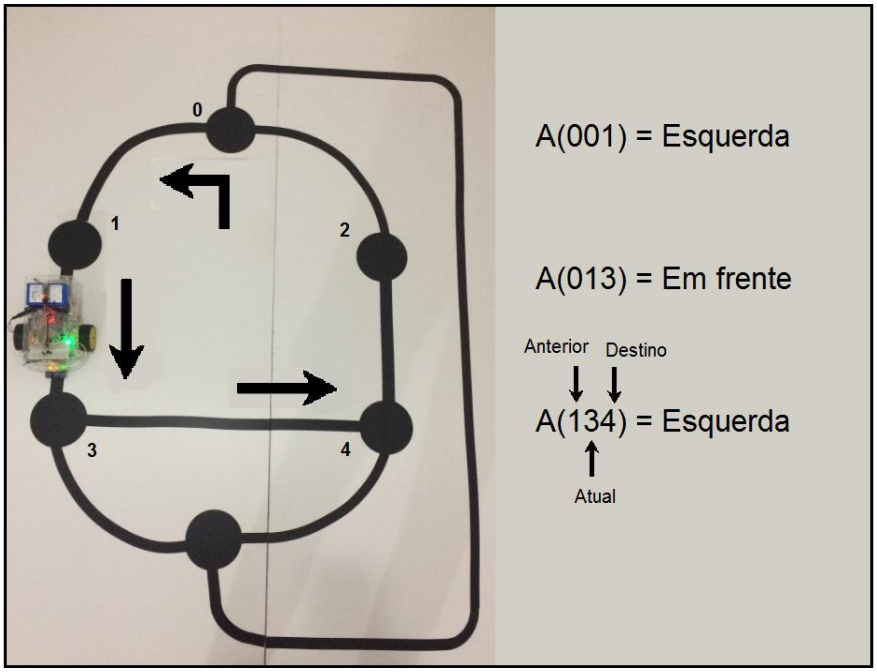
Após selecionar o vértice inicial e o de destino o software apresenta os vértices que o carrinho irá percorrer para chegar ao destino e também apresenta o comprimento total do percurso realizado.

6. Direcionamento do robô

Para determinar as direções das rotas que o robô irá se locomover foi necessário criar uma estrutura de dados para armazenar essas informações. A estrutura utilizada para armazenamento foi a árvore B devido a facilidade de implementação e velocidade de acesso aos dados.

Árvore B é uma estrutura de dados projetada para funcionar especialmente em memória secundária. Dentre suas propriedades ela permite a inserção, remoção e busca de dados a partir de uma chave. A facilidade de acesso a informação utilizando árvore B foi que determinou a utilização dessa estrutura.

O funcionamento da lógica utilizada pode ser observado na figura 7.



8 Figure 7. Lógica das direções

Conforme visto na figura 7 pode-se observar que o acesso a informação é feito através dos números de cada vértice, em que o primeiro número representa o vértice anterior, o segundo representa o vértice atual seguido do próximo vértice que o robô deve-se locomover. Um exemplo seria o robô se locomover do vértice ‘0’ até o vértice ‘4’. Como o robô está partindo do vértice ’0’, ele não possui um vértice anterior, foi definido no software que essa situação deve ser informada com o mesmo código do vértice atual. Logo a busca na Árvore B seria feita através da chave 001, onde 1 representa o vértice de destino. De acordo com o mapeamento realizado com todas as direções possíveis, o robô irá se locomover primeiramente para a esquerda. Essa informação é armazenada em um vetor para posteriormente ser enviada ao Arduino. Em seguida o código gerado será 013, na qual o robô seguirá em frente. Logo após o código gerado será o 134 e o robô irá virar à esquerda. Então o software verifica se o vértice ‘4’ é o vértice final que o aluno informou, e envia as informações do vetor para o Arduino através da comunicação Bluetooth.

6. Resultados e discussões

Os resultados obtidos foram resultados de testes realizados em um conjunto de diferentes larguras de linhas e cores do ambiente. Foi verificado que quando o piso possui alguns pontos com elevações, o sensor de linha acaba enviando ao Arduino valores incorretos de leitura, ocasionando problemas. Uma das soluções encontradas foi alterar a sensibilidade dos sensores de linha e outra foi o posicionamento do sensor mais distante do chão. Outro problema encontrado foi com relação a bateria, inicialmente foi utilizado uma bateria de 9V com 450mAh, porém ao realizar alguns testes foi visto que a bateria acabava descarregando muito rápido, sendo necessário realizar a recarga constantemente. Foi realizada a medição de

consumo de carga do robô quando está em movimento, em que o consumo medido em média foi de 570mA.

Para definir qual bateria utilizar no protótipo precisa-se saber a capacidade de energia que a bateria possui medida em miliampéres por hora cuja sigla é mAh, mAh é o resultado de uma fórmula com 2 unidades, Corrente (mA) x hora tempo (h). Então com essas definições pode-se calcular qual duração de tempo de carga que as duas baterias utilizadas no protótipo possuem, utilizando a seguinte fórmula:

mAh (bateria)/ mA(Consumo carrinho) = Tempo de duração em horas

Utilizando a fórmula chega-se no resultado apresentado na tabela 1 e definiu-se que a bateria correta a ser utilizada no protótipo é a de 4500 mAh.

Descrição	**Tempo de duração em horas**
Bateria 9V 450mA	0.79 ou (47 minutos)
Bateria 12V 4500mA	7.89 ou (7 horas e 53 minutos)

9 Tabela 1. Tempo de duração da bateria de acordo com o consumo do robô

Também realizou-se testes conforme a espessura da linha preta para o robô seguir. Com ela medindo aproximadamente 20mm o robô acabava perdendo muito a referência e andava de maneira incorreta, ocasionando perda de performance na velocidade do robô. Foi alterada a espessura da linha para 30mm e com isso o robô teve uma performance melhorada.

Os dados coletados foram analisados com auxílio do software IBM Statistical Package for the Social Sciencies (SPSS) versão 22.0. As variáveis quantitativas foram expressas por meio de média e desvio padrão.

Os testes estatísticos foram realizados com um nível de significância $\alpha = 0,05$ e, portanto, confiança de 95%. A distribuição dos dados quanto à normalidade foi avaliada por meio da aplicação do teste de Shapiro-Wilk. A investigação da variabilidade das variáveis quantitativas entre as categorias das variáveis qualitativas foi investigada por meio da aplicação do teste de Levene.

A comparação da média das variáveis quantitativas entre as categorias das variáveis qualitativas dicotômicas foi realizada por meio da aplicação do teste t de Student para amostras independentes.

O tempo médio que o dispositivo móvel levou para percorrer a trajetória em uma pista com arestas de 20 mm de largura foi de $13,07 \pm 0,50$ segundos e em uma pista com arestas medindo 30 mm de largura foi de $10,91 \pm 0,28$ segundos, revelando que em uma pista cuja largura das arestas seja de 30 mm, o dispositivo apresenta um desempenho melhor quando comparado a essa mesma pista mas com arestas de 20 mm ($p < 0,001$).

Largura da aresta	**Tempo (segundos)**	**Valor – p***
20 mm	$13,07 \pm 0,50$	$< 0,001$
30 mm	$10,91 \pm 0,28$	

10 Tabela 2. Comparação de tempo de percurso do dispositivo móvel em um grafo de arestas medindo 20 mm e 30 mm de largura

*Valor obtido após aplicação do teste t de Student.

7. Conclusão

Neste trabalho foram apresentados todos os componentes necessários para o desenvolvimento de um robô seguidor de linha, cada parte estrutural que contém no projeto foi devidamente ajustada, contribuindo para o bom funcionamento do protótipo. Além disso através de conhecimentos adquiridos em comunicação de dados foi desenvolvido a comunicação do software com o Arduino para transferência de dados através da tecnologia Bluetooth contribuindo para atingir a meta da implementação.

A estrutura desenvolvida para o robô autônomo apresentou bons resultados e cumpriu seu propósito, atuando como um estímulo de aprendizado aos alunos de algoritmos de grafos. A partir de estudos realizados em teoria de grafos, o software desenvolvido atua como um emissor de comandos e de acordo com as rotas definida pelo aluno, o software o envia ao robô fazendo-lhe mover conforme o menor caminho gerado pelo algoritmo de Dijkstra.

Os resultados finais mostram que todas as metas da etapa do desenvolvimento do projeto foram cumpridas. Várias alterações de projeto foram feitas na tentativa de criar um robô seguidor de linha adequado para os objetivos do projeto. Seguindo as etapas metodológicas foi possível realizar a movimentação do robô autônomo sobre um plano corretamente.

Com base em estudos de teoria dos grafos, foi possível o desenvolvimento do software que utiliza o algoritmo de Dijkstra para determinar o menor caminho entre duas vértices. Para o desenvolvimento do software também foi necessário estudar técnicas de estruturas de dados, onde foi aplicado o conceito de Árvore B para armazenamento das informações que determina as direções que o carrinho irá se locomover.

Devido ao tempo limitado não foi implementada a parte visual do software. Implementar uma interface para proporcionar uma melhor experiência ao aluno, pode ser dado como um dos trabalhos futuros. Outra ideia é implementar um robô seguidor de linha que percorra uma determinada área e calcule todas as distâncias de cada caminho, enviando as informações coletadas para o software que posteriormente pode calcular a menor rota de um ponto inicial até um ponto final. Um dos problemas do trabalho desenvolvido foi encontrar um meio de saber qual direção o software deveria direcionar para ir de um vértice inicial até um vértice de destino. No trabalho foi implementada uma lógica que armazenava todas as direções possíveis de um vértice a outro. Um exemplo de trabalho futuro seria utilizar sensores que detectam a cor para auxiliar na decisão de qual direção o robô deve direcionar para chegar a um vértice de destino.

8. Referências

MÉNDEZ, Y.; GUARDIA, L. Problema do caminho mais curto-Algoritmo de Dijkstra. São Domingos, 2008.

SANTOS, Rodrigues P., Heitor C. A. X. Tbcaed: Um software gráfico para apresentação de algoritmos e estruturas de dados aos iniciantes em computação e informática, 2005.

SILVA, Fernandes D.; SANCHES, Leme A., Aplicação conjunta do método de Dijkstra e otimização combinatória para solução do problema do caixeiro viajante. SegeTSimpósio de Excelência em Gestão e Tecnologia, 2009.

SOARES, T. C. A. P., Cordeiro E. S., Stefani Í. G. A., Tirelo, F. Uma Proposta Metodológica para o Aprendizado de Algoritmos em Grafos Via Animação Não Intrusiva de Algoritmos, III Workshop de Educação em Computação e Informática do Estado de Minas Gerais (WEIMIG). Belo Horizonte, MG, Brasil, 2004.

ZAMBONI, L. C.; PAMBOUKIAN, S. V. D.; BARROS, E. A. R., C++ para Universitários. São Paulo: Páginas & Letras, 2006.