UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO MATHEUS DE CAMPOS CARDOSO

DESENVOLVIMENTO DE UM ROBÔ MÓVEL COM TRAJETÓRIA PROGRAMÁVEL UTILIZANDO A PLATAFORMA ARDUÍNO

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2016

BRUNO MATHEUS DE CAMPOS CARDOSO

DESENVOLVIMENTO DE UM ROBÔ MÓVEL COM TRAJETÓRIA PROGRAMÁVEL UTILIZANDO A PLATAFORMA ARDUÍNO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Erikson Freitas de Morais

PONTA GROSSA

2016

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa

Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UM ROBÔ MÓVEL COM TRAJETÓRIA PROGRAMÁVEL UTILIZANDO A PLATAFORMA ARDUINO

Por

BRUNO MATHEUS DE CAMPOS CARDOSO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 10 de Novembro de 2016 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. Dr. Erikson Freitas de Morais
Orientador

_____
Prof. Dr. André Pinz Borges
Membro titular

_____
Prof. Dr. Gleifer Vaz Alves
Membro titular

_____
Prof. Dr. Augusto Foronda
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. Erikson Freitas de Morais
Coordenador do curso

- A Folha de Aprovação assinada encontra-se arquivada na Secretaria Acadêmica -

AGRADECIMENTOS

Com certeza não conseguiria expressar, neste espaço, minha gratidão a todos aqueles que participaram de algum modo da minha caminhada.

Agradeço primeiramente a Deus, que me concedeu o dom da vida e me fortaleceu nos momentos onde pensei estar sozinho. Agradeço a minha família, principalmente meus pais, Paulo e Janete, e minha irmã Bruna por sempre me incentivarem e nunca me deixarem desistir em meio aos momentos de tribulação, pelo carinho e amor pelo qual sempre me demonstraram. Meus padrinhos Adauto e Elen e seus filhos por tanto amor demonstrado, Edson, Simone e Samuel por cuidarem de mim como um filho durante meu tempo nesta cidade.

Agradeço a minha namorada, Lydiane, que por tantas vezes ficou até de madrugada ajudando, mesmo distante, incentivando e tendo paciência nos momentos de dificuldade, por sempre se demonstrar firme e mostrar que tudo iria ficar bem.

Agradeço ao meu orientador Prof. Dr. Erikson, que me proporcionou participar deste especial projeto e pela paciência e dedicação com que me ajudou e me guiou nesta trajetória e aos professores que de alguma forma contribuíram para incremento do meu conhecimento e crescimento.

Aos meus amigos de tantos anos de faculdade e que foram de suma importância para me manter firmes nesta luta, Allan, Breno, Diego, Diogo, Filipe, Jônatas, Letícia, Lucas Migliorini, Priscilla, os quais compartilhei e adquiri conhecimento que foram fundamentais para minha formação. Os que conheci no tempo em que estive nesta cidade, Antonia, Debora, Evandro, Fernanda, João, Larissa, Lucas “FerCam”, Pr. Leandro, Rubinho, Stella, Taciane, Will e Jéssica.

Aos meus amigos que ficaram em minha cidade, em especial ao Pr. Walner e sua família, Adelmo, Danilo, “Dé”, Dauta, Eduardo, Jaqueline, Laercio, Lucas Castilho, Lucas Wilian, Tiago, Walninho e os que fiz durante os acampamentos e outros momentos da vida, Christian, Dani, Edinho, Julia, Lívia, Mayara, Thairine, que me proporcionaram muitas risadas e momentos de descontração e momentos de alegria.

Enfim, agradeço a todos que participaram de alguma forma contribuindo com este trabalho, ou me incentivando na realização do mesmo.

RESUMO

CARDOSO, Bruno Matheus de Campos. **Desenvolvimento de um Robô Móvel com Trajetória Programável Utilizando a Plataforma Arduíno**. 2016. 55 f. Trabalho de Conclusão de Curso Superior de Bacharelado em Ciência da Computação - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016.

Os problemas envolvendo trajetórias móveis no âmbito da robótica vêm sendo estudados desde a década de 60 e utilizados em larga escala nos diversos setores da nossa sociedade, como reconhecimento de ambientes, indústrias, e ambientes que podem ser nocivos ao ser humano. Este trabalho tem como principal objetivo o estudo de técnicas e ferramentas utilizadas no desenvolvimento de trajetórias para robôs móveis, utilizando para isso a plataforma Arduino, um sistema de prototipação eletrônica que conta com *hardware* livre. Procura-se não somente desenvolver a parte física de um robô, como também o algoritmo para o problema proposto. Esta arquitetura é também conhecida por sua relativa facilidade de trabalho aliado a um baixo custo. Deseja-se que o robô consiga percorrer uma trajetória programada previamente.

Palavras-chave: Arduino. Robô. Trajetória. Móvel. Robótica.

ABSTRACT

CARDOSO, Bruno Matheus de Campos Cardoso. **Development of a Mobile Robot with Programmable Path using the Arduino Platform**. 2016. 55 f. Work of Conclusion Course (Graduation in Computer Science) – Federal Technology University - Paraná. Ponta Grossa, 2016.

The problems involving paths mobiles within robotics have been studied since 60's and used in large scale in the many sectors of our society, like recognition environments, industries and environments that can be harmful to the humans. This work has as main goal the study of techniques and tools used in development of path to mobile robot, using the Arduino platform, an electronic prototyping system which includes free hardware. It seeks to not only develop the physical part of the robot, as also the algorithm for the proposed problem. This architecture is also known by relative facility to work together with a low cost. It's desired that the robot can go a programmed path.

Keywords: Arduino. Robot. Path. Mobile. Robotic.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ACRÔNIMOS

LISTA DE SIGLAS

CI	Circuitos Integrados
E/S	Entrada/Saída
GPS	Global Positioning System
IDE	Integrated Development Environment
PWM	Pulse Width Modulation
SMS	Short Message Service
USB	Universal Serial Bus

LISTA DE ACRÔNIMOS

IDII	Interaction Design Institute Ivrea
NASA	National Aeronautics and Space Administration
RIA	Robot Institute of America
RISC	Reduced Instruction Set Computer

SUMÁRIO

1 INTRODUÇÃO

Os problemas envolvendo trajetórias móveis no âmbito da robótica vêm sendo estudados desde a década de 60 e, a partir de então, foram inseridos gradativamente na indústria (ADÔRNO et. al, 2006). Segundo Secchi, esse aumento, motivado por uma grande variedade de possibilidades, atraiu o interesse dos pesquisadores em desenvolver robôs mais rápidos, precisos e com certa facilidade de programação. Este avanço também fez com que o ramo da automação industrial desse um novo passo e, com isso, surgiram os primeiros modelos de célula de fabricação robotizada (SECCHI, 2008).

O conceito de robô tem origem moderna. Porém é possível encontrar pela antiguidade diversos projetos que, nos dias de hoje, poderiam facilmente ser considerados como instrumentos autônomos (ou robôs). Na Grécia, apareceram alguns sistemas de pesos e bombas automáticas (ANDRADE, 2013). No Japão, surgiram os Karakuris, mecanismos de cordas e engrenagens que possuíam bonecos e marionetes utilizados para servir chá, apresentações ou outras situações que pudessem impressionar os nobres. Aqueles que utilizavam Karakuris eram também conhecidos como “mágicos” (SCHLATTER, 2013).

Na literatura moderna, um robô é descrito como um agente mecânico ou virtual, geralmente uma máquina eletromecânica, controlado por um programa computacional ou circuito eletrônico.

O termo derivou de uma peça teatral de cunho satírico, escrita por Karel Capek, e originalmente foi utilizado para referir-se à “força laboral” ou “trabalho forçado”, como pode ser observado na Figura 1 (CAPEK, 1921).

A robótica é uma área que cresce cada vez mais e nos mais diversos campos, sendo utilizada principalmente em aplicações comerciais, residenciais e industriais.

Desde 2010, a demanda por robôs industriais tem acelerado consideravelmente devido à uma grande tendência de automação e a contínua melhoria de técnicas inovadoras de robôs industriais.

Figura 1: Cena da peça de teatro RUR

Fonte: University of Michigan - *Robots In Literature*

De acordo com o IFR (*International Federation of Robotics*), em 2014 a venda de robôs teve um crescimento de 29% em relação ao ano de 2013, atingindo um volume de mais de 229 mil unidades. Fornecedores de peças automotivas e indústrias elétrico/eletrônicas foram os principais responsáveis por este crescimento, liderados pela China que compôs aproximadamente 25% do total de vendas realizadas no período (IFR, 2016).

O termo robótica tem sido popularmente atribuído a Isaac Asimov em seu renomado livro "*I, Robot"* (Eu, Robô) referindo-se ao estudo e utilização dos robôs (ASIMOV, 1950). Além disso, dele também surgiram às proposições das três principais leis da robótica:

1ª: Um robô não pode ferir um ser humano ou permitir que um ser humano sofra algum mal;

2ª: Um robô deve obedecer às ordens que lhe sejam dadas por seres humanos, exceto nos casos em que tais ordens entrem em conflito com a Primeira Lei; e por fim

3ª: Um robô deve proteger sua própria existência, desde que tal proteção não entre em conflito com a Primeira ou Segunda Lei.

Posteriormente, Asimov adicionou outra lei que ficou conhecida como "Lei Zero". Esta última, que estaria acima das outras leis, diz que: um robô não pode causar mal a humanidade ou, por omissão, permitir que a humanidade sofra algum mal (RIBEIRO, 2006).

Segundo o RIA (*Robot Institute of America*) o conceito de um robô é definido como um manipulador reprogramável, multifuncional, projetado para mover

materiais, peças, ferramentas ou dispositivos especializados, através de várias funções programadas para o desempenho de uma variedade de tarefas (RIVIN, 1988).

A preferência pela utilização de robôs em sistemas industriais traz algumas vantagens, como por exemplo, o fato de não se cansarem, não necessitarem de salários, conseguem manter um padrão uniforme no seu trabalho, melhoram a qualidade e também aumentam a produção.

Porém, eles também apresentam algumas desvantagens como: custo dos reparos, mão de obra capacitada para eventuais problemas, possuem movimentos e aprendizagem limitados e, muitas vezes, necessitam de um sistema sofisticado e maior autonomia na tomada de decisões.

Existem diversos tipos e classificações de robôs, porém não existe ainda uma classificação definitiva. De modo geral, podem ser agrupados seguindo a abordagem de três aspectos: segundo anatomia (aéreo, terrestre, aquático), o tipo de controle empregado (teleoperados, semiautônomos e autônomos) e a funcionalidade (industriais, campo, serviço e pessoais) (PIERI, 2012).

Um robô é considerado móvel se ele pode se deslocar e interagir com um ambiente, seja no solo (esteiras, rodas, patas), ar ou água (WOLF, 2009). Além disso, os robôs móveis têm tomado grandes proporções e têm sido cada vez mais objeto de estudo de muitos pesquisadores e interessados no assunto. A robótica móvel tem se desenvolvido constantemente, atraindo os olhares e atenção de uma grande comunidade que busca pelo desenvolvimento de novas tecnologias e melhoria das técnicas já existentes.

O principal objetivo deste trabalho é apresentar os passos para o desenvolvimento de um robô móvel com trajetória programável utilizando Arduino. O Arduino é uma plataforma de prototipação eletrônica livre, de baixo custo e com um grande potencial para desenvolvimento de diversos tipos e gêneros de projetos, desde os mais simples até projetos mais complexos.

Neste trabalho, procura-se apresentar os diversos aspectos, desde a modelagem do projeto, montagem, funcionamento básico e também os algoritmos utilizados no desenvolvimento do robô.

1.1 OBJETIVOS

Abaixo são listados os objetivos gerais e específicos para a execução deste trabalho.

1.1.1 OBJETIVO GERAL

O objetivo geral é utilizar o microcontrolador Arduino para desenvolver um robô móvel terrestre, capaz de se deslocar em uma trajetória pré-determinada, desenvolver as ações para o robô seguir a rota estipulada até alcançar o seu destino.

1.1.2 OBJETIVOS ESPECÍFICOS

Para que seja alcançado o objetivo principal do trabalho, alguns passos são necessários, listados na forma de objetivos específicos:

- Estudar e compreender os componentes utilizados no projeto e a sua teoria, como: microcontrolador Arduino e a linguagem de desenvolvimento utilizada, Zumo Chassis, Dual Motor *Shield* e micromotor;
- Estudar e entender como as partes mecânicas e eletrônicas se relacionam com o desenvolvimento das trajetórias;
- Construir o robô: desenvolvimento de um projeto e montagem do *hardware* (estrutura física) do robô;
- Estudar os *shields:* expansões disponíveis para utilização em conjunto com o Arduino, em especial o escolhido pra este trabalho, Dual Motor *Shield,* e sua biblioteca própria que são responsáveis por controlar os motores do robô;
- Implementar o algoritmo para realizar a movimentação do robô;
- Identificar as restrições apresentadas pelo robô, tais como: ambiente em que é possível utilizá-lo, eventuais restrições mecânicas, mudanças de direção do robô, peso e velocidade dos motores.

1.2 JUSTIFICATIVA

A robótica é uma área que está em constante crescimento e evolução, causando um grande impacto nos diversos setores da sociedade, e também muito utilizada na realização de uma variedade de tarefas, dentre elas: reconhecimento de ambientes, agricultura, diversos setores da indústria (automotiva, alimentícia, farmacêutica, entre outras), instalações militares e ambientes que podem ser nocivos ao ser humano (WOLF et. al*,* 2009).

As pesquisas nesta área envolvem a utilização do conhecimento de diversas áreas, como Engenharia Mecânica, Engenharia Eletrônica e as diferentes áreas da Computação. Esta última, em especial, tem uma grande importância na utilização de técnicas que permitem aos robôs possuírem “sistemas mais robustos, seguros, autonomos e inteligentes” (WOLF et. al, 2009). Tem sido amplamente impulsionada na busca por automatizar processos, deixando-os mais simples, reduzindo custos e aumentando a produção industrial.

A robótica móvel é uma subárea da robótica que vem apresentando muitos avanços no seu campo de atuação. São robôs de menor tamanho e que podem conter sensores ou atuadores. Segundo Vieira “Um robô móvel é composto por uma arquitetura de hardware e software que é responsável por garantir a execução correta de sua navegação para uma determinada tarefa” (VIEIRA, 2005). Assim, justifica-se sua utilização na sociedade no uso de aplicações domésticas, industriais, urbanas e militares.

No campo da robótica, existem diversas abordagens que fazem o uso de robôs móveis, como: robôs seguidores de linha, robôs exploradores, robôs controlados via software, seja por módulo Bluetooth, wifi*,* ou operados de outras maneiras, como utilizando-se de um microcontrolador, que é o modelo utilizado neste projeto.

Além disso, a robótica móvel tem sido objeto de estudo intensificado por grandes instituições de pesquisa, como a NASA, que tem utilizado seus robôs exploradores para verificação de substâncias presentes na atmosfera e na superfície de outros planetas (NASA, 2015).

Assim, com este projeto, busca-se então apresentar conceitos básicos da robótica móvel, a ampliação da literatura neste campo de estudo e o desenvolvimento de soluções que possam contribuir para o desenvolvimento da área da robótica móvel.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante do trabalho está dividido em 5 capítulos. No capítulo 2 apresenta-se o referencial teórico, onde encontra-se uma breve explicação sobre microcontroladores, uma abordagem histórica do Arduino e também onde são explicados alguns dos principais componentes utilizados no projeto. No capítulo 3 tem-se o desenvolvimento, explicando como foi realizado o projeto e como os componentes foram inseridos e conectados ao projeto. O capítulo 4 apresenta uma breve explicação dos resultados alcançados durante a realização do trabalho. No capítulo 5 está a conclusão do presente trabalho e as considerações finais. E por fim no capítulo 6 tem-se a descrição de sugestões para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Para o desenvolvimento de um robô móvel, alguns conhecimentos são necessários para um melhor entendimento do projeto. Assim, temos a fundamentação teórica, onde são especificados algumas etapas do trabalho e os principais componentes necessários para execução do projeto.

2.1 MICROCONTROLADOR

Com a diminuição dos custos dos Circuitos Integrados (CIs) e o surgimento dos microprocessadores, que possuem processamentos maiores e genéricos, houve a necessidade de desenvolver soluções para tarefas de sistemas dedicados e embarcados. Assim, surgiram os primeiros microcontroladores, como os PICs (*Programmable Interface Controller)*.

O microcontrolador é uma unidade de processamento que contém todos os “ingredientes” necessários ao seu funcionamento, desde memórias voláteis/não voláteis a conversores analógicos/digitais, tudo em um único chip. Esta última característica é que faz com que eles sejam amplamente utilizados para sistemas específicos (BENTES, 2013).

Os microcontroladores operam em uma frequência bem inferior a um microprocessador e possuem uma quantidade de dados também reduzida, porém o que a torna adequada para a maioria das aplicações que são designados a trabalharem. Consomem pouca energia e podem entrar em modo *stand by,* ou seja, em modo de espera até receber um evento externo, como apertar um botão, pressionamento de uma tecla, acionamento de um sensor. São utilizados principalmente em projetos de controle e automação.

2.2 ARDUINO

O Arduino é uma plataforma para prototipação eletrônica que contém hardware livre (qualquer pessoa pode modificar ou personalizar sua estrutura) e placa única (um sistema onde todos os componentes que serão utilizados estão

armazenados em uma placa de circuito impresso) que utiliza um microcontrolador programável para desenvolver diversas aplicações (MENEGUELE et. al, 2011).

O Arduino é considerado uma plataforma de computação embarcada, onde existe interação com o ambiente através de *hardware* e *software* (MCROBERTS, 2011).

2.2.1 HISTÓRICO

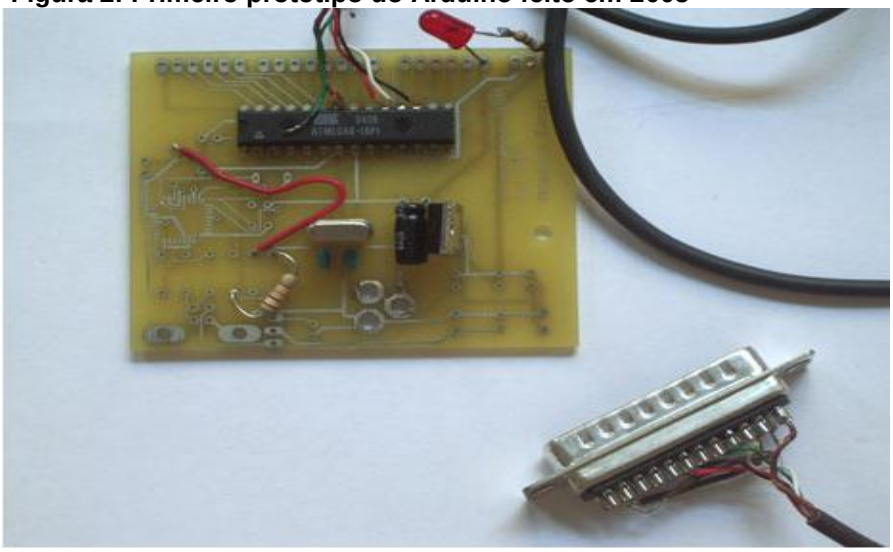
O desenvolvimento do Arduino iniciou em 2005 na Itália no *Interaction Design Institute Ivrea* (IDII), na cidade de Ivrea, elaborado por um grupo de cinco pesquisadores: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. O projeto foi liderado pelo professor Massimo Banzi, que tinha como objetivo uma forma mais barata dos estudantes de *design* se envolverem com tecnologia.

Assim, ele levou o problema até outro professor, David Cuartielles de uma universidade suíça, que também procurava pela mesma solução. Os produtos até então existentes, possuíam um preço elevado e certa dificuldade para se trabalhar.

Foi então que juntos decidiram desenvolver o seu próprio microcontrolador, algo que tivesse um baixo custo, fosse funcional e tivesse fácil programação, para chamar atenção dos outros estudantes e também de quaisquer outras pessoas que demonstrassem interesse na área para que pudessem utilizá-lo nos seus projetos.

Assim, Cuartielles fez o desenho de como deveria ser a placa e um aluno de Banzi, David Mellis, fez a programação do *software.* Eles também contrataram um engenheiro local chamado Gianluca Martino, que aceitou participar do projeto e fez a produção das primeiras duzentas placas (EVANS et. al, 2013). Os primeiros modelos da placa eram semelhantes à Figura 2.

Figura 2: Primeiro protótipo do Arduino feito em 2005



Fonte: Kushner, 2011

Em um primeiro momento a placa ainda não havia sido nomeada, foi então que Massimo Banzi sugeriu o nome Arduino, em referência a um bar local muito frequentado por estudantes da IIDI chamado "*Re di Arduino*" em homenagem a um antigo rei da região chamado Arduin. Para facilitar a utilização e desenvolvimento dos projetos por parte dos alunos, as placas passaram a ser vendidas em kits. Esses kits obtiveram enorme sucesso rapidamente, e logo mais kits passaram a ser produzidos para suprir a demanda.

Obteve sucesso também entre outros públicos quando começaram a perceber a facilidade de uso, o baixo custo e a variedade de projetos que poderiam ser desenvolvidos utilizando-se do Arduino.

Devido a uma disputa judicial entre dois dos fundadores da empresa, desde maio de 2015 as placas produzidas pela empresa com a marca Arduino têm sido comercializada apenas nos Estados Unidos. Nos demais países vêm sendo comercializada como Genuino (SENESE, 2015).

Existem diversas versões e modificações das placas Arduino. Posteriormente serão dadas pequenas introduções sobre alguns dos principais modelos de placas Arduino, suas especificações, as utilizações e as principais diferenças entre uma e outra.

2.2.2 ESPECIFICAÇÕES

A maior parte dos modelos desenvolvidos são baseados em um microcontrolador Atmel de 8 bits AVR *Reduced Instruction Set Computer* (RISC). A primeira placa desenvolvida possuia um *clock* de 16 MHz e memória *flash* de 8KB e mais tarde, outras placas desenvolvidas utilizavam memória de 16 KB.

Atualmente, as versões do Uno e Duemilanove, também conhecida como Due, usam o Atmega328 com memória *flash* de 32 KB. O Arduino Mega1280 possui memória de 128 KB e o Arduino Mega2560 de 256 KB, podendo ser aplicados em projetos que necessitam de mais Entrada/Saída (EVANS et. al, 2013).

Essas placas possuem, em geral, 14 pinos de entradas digitais, que podem ser definidas como entrada e/ou saída. Quando trabalha-se com LED, configuramos como saída, e ao utilizar a leitura de uma tecla, por exemplo, configura-se como entrada.

A configuração padrão do Arduino está em modo de entradas digitais. Este tipo de entrada permite apenas dois estados: *HIGH* e *LOW,* simplificados como 1 e 0 (FOROUZAN, 2006).

Estas entradas também podem ser programadas para funcionarem como saída de modulação por largura de pulso (*Pulse Width Modulation* – PWM), que consiste uma técnica muito utilizada para chaveamento de fontes e controle de velocidade, onde utilizando uma largura de pulso é possível que a frequência de uma onda se mantenha, assim variando quanto tempo o sinal fica em *HIGH* (SOUZA-a, 2013).

A maioria das placas Arduino também possuem outros seis pinos de entradas analógicas. As saídas analógicas possuem infinitos valores de tensão num determinado período de tempo (FOROUZAN, 2006).

Como o Arduino trabalha internamente apenas com dados digitais, é utilizada uma técnica para a leitura do sinal analógico e conversão em digital, utilizando o conversor Analógico Digital, ou A/D (SOUZA-a, 2013) (EVANS et. al, 2013).

Esta conversão é dada pela Equação 1:

$$\textbf{resolução} = \textbf{Vref} / 2^{n} \quad (1)$$

Onde:

Vref = nível de tensão de referência utilizada no conversor A/C;

n = número de bits que o conversor possui (SOUZA-a, 2013);

As placas podem ser alimentadas facilmente, tanto conectando-se a um computador através de um cabo USB, ou utilizando-se de uma fonte externa de alimentação (EVANS et. al, 2013).

Para o desenvolvimento de projetos, possui uma linguagem de programação própria baseada no C/C++, facilitando o trabalho para usuários que não possuem um conhecimento tão grande ou até mesmo aqueles que nunca utilizaram linguagens de baixo nível presentes em outros microcontroladores, que geralmente utilizam a linguagem *Assembly*.

2.2.3 MODELOS EXISTENTES

O Arduino é composto, basicamente, de duas principais partes: a placa Arduino, que é o hardware para desenvolver os projetos e o Arduino IDE, onde são desenvolvidos os softwares (*sketches*) para fazer o *upload* até a placa. Nesta seção serão abordados alguns dos principais modelos. Existem diversos modelos de placas Arduino disponíveis no mercado e para cada aplicação pode ser usada uma versão diferente dependendo das necessidades do projeto.

Entre as placas mais conhecidas estão: Arduino Uno, 101, Mega, Pro e Mini, que podem ser vistos na Figura 3. No próprio site da empresa, essas placas são designadas como “Nível de Entrada”, além de outros níveis, como Recursos Avançados, *Internet of Things,* Vestíveis e Impressão 3D (ARDUINO, 2015). Por serem de fácil utilização e prontas para iniciar os primeiros projetos, elas são indicadas para quem está aprendendo as partes eletrônicas e cujo foco está na codificação e não no hardware.

Figura 3: Alguns modelos de placas Arduino

Fonte: Arduino Team

Os modelos listados na Figura 4, são geralmente utilizados para projetos complexos, com funcionalidades avançadas ou que necessitam de maiores desempenhos para trabalhar. Alguns desses modelos mais utilizados para executar essas tarefas são o Arduino Mega, o Due e o Zero.

Figura 4: Alguns modelos placas Arduino (continuação)

Fonte: Arduino Team

A Tabela 1 compara os principais modelos de placas existentes no mercado, utilizando características chaves de cada placa, como *clock,* números de entrada/saída digitais e analógicas, voltagem da alimentação e tamanho da memória *flash* disponível. Posteriormente o modelo UNO R3 será descrito com maiores detalhes.

Tabela 1: Comparativo de modelos de Placas Arduino

Modelo	Clock	E/S Digital	E/S Analógico	Alimentação	Flash
Due (ARM)	84 MHz	54	12	12 V	512 KB
Leonardo	16 MHz	20	12	12 V	32 KB
Uno	16 MHz	14	6	12 V	32 KB
Duemilanove	16 MHz	14	6	12 V	32 KB
Pro	8 MHz	14	6	12 V	32 KB
Mega	16 MHz	54	16	12 V	256 KB
Mini 05	16 MHz	14	6	9 V	32 KB
Fio	8 MHz	14	8	12 V	32 KB
LilyPad	8 MHz	14	6	5,5 V	32 KB

Fonte: Bentes, 2013 (Adaptado)

Como é possível analisar pela Tabela 1, as placas mais leves como Uno, Due, Pro e outras, possuem, em sua maioria, 14 pinos de E/S digitais, 6 de entradas analógicas, memória *flash* de 32 KB e *clock* de 16 MHz. A placa com especificações mais robustas é a Duemilanove (Due), com *clock* de 84 MHz, 54 pinos de E/S digitais e 12 analógicas, e memória *flash* de 512KB.

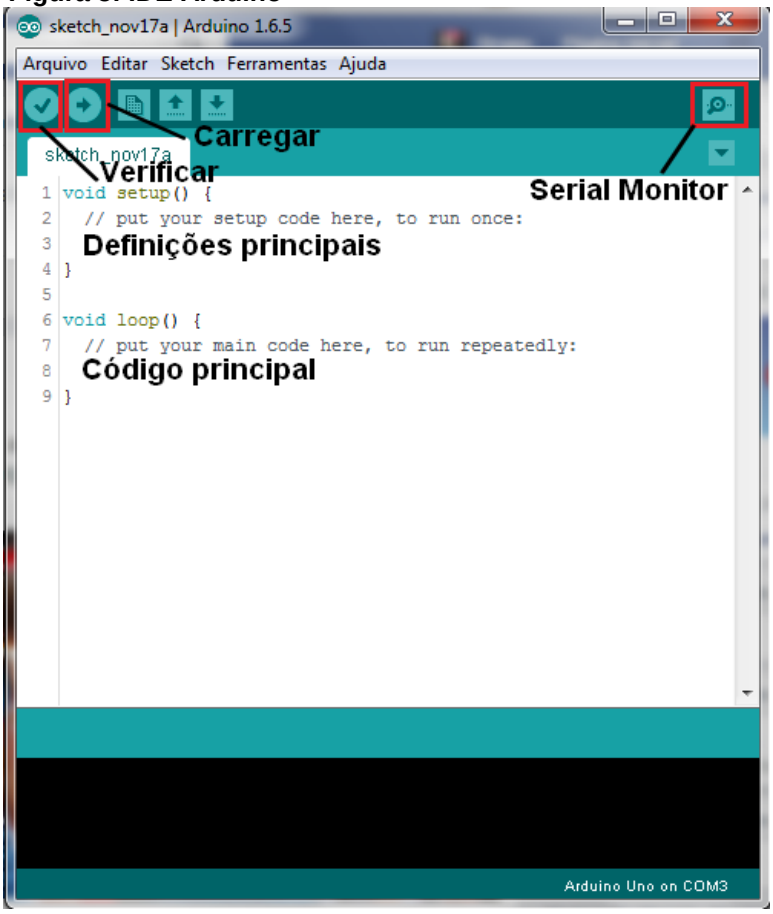
2.3 AMBIENTE DE DESENVOLVIMENTO

A IDE (*Integrated Development Environment*) é um programa executado no computador que permite escrever *sketches* para a placa Arduino em uma simples linguagem modelada depois de ser realizado o processamento da linguagem. Após apertar o botão “carregar” (ver Figura 6), o código escrito é traduzido para a linguagem C e então é passado para o compilador avr-gcc, que é uma parte de *software open source* que faz a tradução final para a linguagem entendida pelo microcontrolador. Esse último passo é importante porque o Arduino deixa tudo mais simples, escondendo as complexidades de programação de microcontroladores dos projetistas (BANZI, 2011).

Outras características importantes desta ferramenta são a portabilidade e a facilidade de uso nos diversos sistemas operacionais mais utilizados hoje, como Windows, Linux e Mac OS. Na Figura 5 é possível ver algumas das principais

características da IDE utilizada pelo Arduino, uma ferramenta gratuita e pode ser baixada diretamente pelo site do Arduino Team.

Figura 5: IDE Arduino



Fonte: Autoria própria

O ciclo básico de programação no Arduino é descrito abaixo (BANZI, 2011):

1 Conecte a placa do Arduino em uma porta USB no computador;
2 Escreva um *sketch* que fará o Arduino funcionar;
3 Carregue este *sketch* para a placa através da conexão USB e espere a placa reiniciar;
4 A placa executará o *sketch* que foi escrito.

Para começar o desenvolvimento do *sketch,* a IDE oferece duas funções básicas: "*void setup*()", onde é colocado todo código que deve ser executado no início do programa e "*void loop*()", que contém todo o código principal do programa e que será executado diversas vezes. Isso acontece porque o Arduino não é como um computador normal, isto é, não consegue executar diversos programas ao mesmo

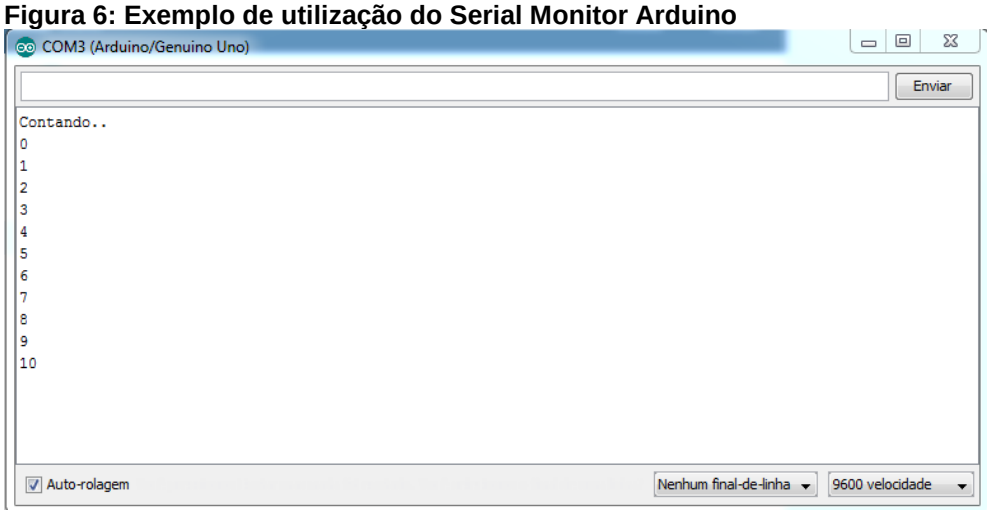
tempo. Quando a placa é conectada o código irá começar a ser executado e se quiser pará-lo é necessário o desligamento da placa.

Após o desenvolvimento do *sketch* pode-se clicar no botão VERIFICAR, que fará a verificação do código desenvolvido e apontará eventuais erros ou dirá se o código escrito está correto. Então, o próximo passo é clicar no botão CARREGAR que enviará o código compilado (já traduzido) para a placa.

Toda vez que há um novo carregamento de código na placa, a mesma faz uma reinicialização do circuito, sobrescrevendo os dados anteriores contidos na memória RAM da placa. Essa reinicialização também pode ser executada pressionando-se o botão *RESET (*BENTES, 2013).

O botão SERIAL MONITOR é uma ferramenta interessante quando é necessário visualizar informações em tempo real que estão sendo trocados pelo Arduino e o computador. Essa ferramenta pode ser observada através da Figura 6.

Figura 6: Exemplo de utilização do Serial Monitor Arduino



Fonte: Autoria Própria

2.4 EXPANSÕES

As placas Arduino, e algumas outras semelhantes, possuem expansões para o *hardware,* e estas expansões são conhecidas como *Shields*. Essas expansões são conectadas na parte de cima do microcontrolador, fazendo a comunicação com a placa principal através dos pinos de E/S, tanto analógicos/digitais, ou de pinos seriais.

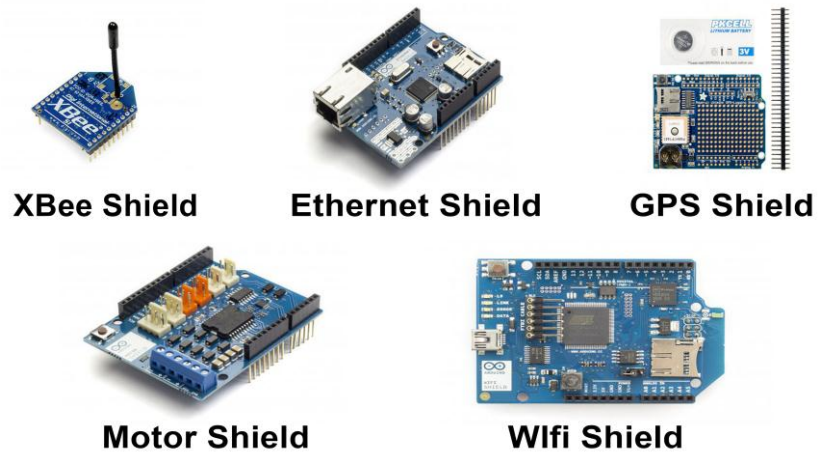
A principal ideia por trás destas expansões está no desenvolvimento de componentes que possuam certa especialização, assim como acontece com os *frameworks* no desenvolvimento de *software* (BENTES, 2013). Existem vários *shields* para serem utilizados em diversas aplicações, alguns dos mais conhecidos são:

- Arduino Ethernet: este permite a conexão do Arduino com a Internet utilizando-se um cabo de rede padrão;
- WiFi *Shield:* tem sido amplamente utilizado no desenvolvimento de aplicações voltadas para a chamada Internet das Coisas (IoT – *Internet of Things*);
- Bluetooth *Shield:* utilizado para aplicações de comunicação sem fio, e seu alcance máximo varia de acordo com cada fabricante, mas possui, em média, alcance de até 10 metros;
- Controle de motores, Motor *Shield*: responsável por controlar os motores conectados a uma placa, onde cada fabricante possui suas próprias bibliotecas;
- Xbee *Shield*: permite a conexão *wireless* de um Arduino com um módulo Xbee com alcance de até 30 metros em ambiente fechado e até 90 metros em ambiente aberto;
- GPS: este *shield* permite utilizar a tecnologia GSM, GPRS e GPS na mesma placa, podendo utilizar SMS, voz e dados via Internet;

Além destes citados, existem outros *shields* utilizados em outras aplicações. É importante observar a facilidade de conexão entre os *shields* e as placas, já que estes utilizam um padrão de pinagem, assim possibilitando o uso de vários *shields* diferentes sem preocupar-se com a aplicação em si.

Para o desenvolvimento deste trabalho em específico, foi optado por utilizar um *shield* para fazer o controle dos motores do robô, o Dual Motor *Shield*. Na próxima seção esse *shield* será especificado com maiores detalhes. Abaixo, na Figura 7, pode-se observar os modelos citados anteriormente:

Figura 7: Exemplos de Shields



Fonte: Arduino Team (Adaptado)

2.5 MOTOR

Um dos principais requisitos em sistemas embarcados, tais como os robôs móveis, é o deslocamento físico propriamente dito. Muitos dos atuadores utilizados para criar esses movimentos são elétricos. Solenoides (ou bobinas) podem ser usados para criar movimento linear, servos somente para movimentos angulares, e os motores DC (*Direct Current)* e de passo podem ser usados para movimentos angular e de rotação (WILMSHURST, 2007).

Motores DC e motores de passo são largamente utilizados em sistemas embarcados. Mesmo sendo baseados em princípios diferentes, eles acabam concorrendo por aplicações bem similares. Através de uma aplicação correta, ambos podem ser utilizados para movimentos rotativos ou movimentos angulares.

O motor transforma energia elétrica em energia mecânica. Eles são divididos em duas categorias: os motores de corrente contínua (CC) e motores de corrente alternada (CA). Os motores de corrente alternada são usualmente utilizados por máquinas de grande porte e também recebem sua energia de uma rede de distribuição de energia. Os robôs móveis, por sua vez, utilizam motores de corrente contínua, pois geralmente são carregados com bateria ou pilha (GIOPPO et. al, 2009).

Sua popularidade se dá devido à sua grande capacidade de controlar sua velocidade, faixa de velocidade útil e sua potencial boa eficiência (WILMSHURST, 2007).

Para que o robô consiga realizar o deslocamento correto, o motor deverá trabalhar em uma velocidade positiva e com um torque mais baixo (O torque é uma força que faz com que objetos sejam girados ou rodados (NICE, 2015)). Assim, para que essa ordem seja invertida, este motor deve estar conectado a uma caixa de redução que gera nova saída com giros mais lentos, assim produzindo um torque maior. Hoje a maior parte dos motores de corrente contínua disponíveis no mercado são vendidos com a caixa de redução neles, como o motor utilizado neste projeto (GIOPPO et. al, 2009).

Na caixa de redução existem várias combinações possíveis de voltas. Uma taxa de redução tem combinação X:Y, o que indica que o motor irá trabalhar com X voltas para o eixo rodar Y voltas.

Estes micromotores podem ser usados propriamente com 6V de tensão, mesmo que se tenha a possibilidade de usá-los com tensões acima ou abaixo desta tensão normal, devem operar na faixa de 3-6V. Com tensões menores ou muito altas, pode ocorrer mal funcionamento do micromotor, levando a diminuição drástica da vida útil do mesmo.

Estão disponíveis em uma grande variedade de relações, desde 5:1 até 1000:1, também oferecendo escolhas entre motores de diferentes potências: alta potência, médica potência e padrão. Todos os micromotores, menos o de 1000:1, possuem o mesmo tamanho físico, facilitando a substituição por outro, de acordo com as necessidades do projeto (BAÚ DA ELETRÔNICA, 2016).

2.6 CHASSI

O chassi consiste na estrutura física do robô. No mercado existem diversas opções de tamanho, estruturas e modelos de chassi, podendo ser de acrílico, plástico, ou algum outro material. Essa estrutura vem geralmente quase pronta, bastando apenas realizar a montagem do kit.

O chassi para robô móvel pode ser tradicional, com duas ou quatro rodas, ou do tipo esteira, que é bom para utilizar em terrenos acidentados. Eles conseguem passar pelos obstáculos maiores que um robô com rodas tradicionais não conseguiriam. Porém, esse tipo de robô exige um gasto maior com energia durante a realização do seu deslocamento.

Com este tipo de chassi, torna-se desnecessário a utilização de rodas direcionais para realização de deslocamentos laterais ou curvas. Assim, quando for necessário realizar um deslocamento lateral, basta que uma esteira gire em um sentido e o outro lado gire no sentido inverso. Já para a realização do movimento de curva, basta que uma das esteiras gire mais rapidamente do que a outra esteira. Isto facilita a mobilidade e o controle do robô.

O tipo de chassi a ser escolhido depende da especificação e das necessidades de cada projeto.

Como exemplo de chassis tem-se: Zumo Chassi, da Pololu (POLOLU-a, 2015), chassi esteira Tamiya (HOBBY MODELISMO, 2016), 2 rodas (2WD) e 4 rodas (4WD) (FILIPE FLOP, 2016).

2.7 TRABALHOS RELACIONADOS

Nesta seção, serão apresentados alguns trabalhos semelhantes e que nortearam o desenvolvimento deste trabalho durante o levantamento bibliográfico.

Meneguele (MENEGUELE et al., 2011) descreve a construção de um robô autônomo que, dado um ponto qualquer, consegue explorar e solucionar um labirinto 2D. Utilizando-se fitas pretas e um fundo branco para compor o labirinto, o robô deve percorrê-lo deslocando-se sempre sobre as linhas pretas. Utiliza sensores infravermelhos e usando o algoritmo Seguidor de Parede para realizar as tomadas de decisões. No desenvolvimento do projeto, utilizaram um Arduino modelo Duemilanove acoplado a um chassi desenvolvido pelo próprio grupo e motores CC com caixa de redução.

Perez (PEREZ et al., 2013) demonstram como a utilização da plataforma Arduino pode auxiliar no ensino e aprendizagem da robótica. Os alunos procuraram ministrar aulas de eletrônica, programação e robótica móvel para alunos da rede estadual de educação, capacitando os alunos no desenvolvimento de aplicações robóticas de pequeno porte. Utilizando-se um kit básico de programação em Arduino, contendo a placa Uno com um chassi 2W e vários outros elementos, como *jumpers*, resistores, LEDs, etc.

Andrade (ANDRADE, 2013) propôs um robô autônomo seguidor de linha que sem utilizar a eletrônica digital consiga percorrer uma determinada trajetória, no

caso, uma linha preta com fundo branco. Com um sistema baseando-se na emissão e detecção da luz que é refletida no solo para que seja possível controlar os motores DC a partir da intensidade do sinal que é gerado e captado pelo sistema. Utilizou neste projeto uma plataforma robótica da *HobbyKing* e uma placa de circuito impresso desenvolvida pelo próprio aluno.

Gioppo (GIOPPO et al., 2009) também propôs um robô autônomo seguidor de linha, porém utilizando-se de uma placa Arduino. Assim como Andrade, são utilizados sensores infravermelhos para detecção e captação do sinal e da intensidade do mesmo. O chassi do robô foi desenvolvido utilizando-se papel foam, pois possui certa rigidez, mas também apresenta uma leveza ao mesmo tempo, e podendo ser modelado da melhor maneira, adaptando-se ao projeto. O microcontrolador utilizado no projeto foi o Freeduino, uma versão derivada do Arduino e suas aplicações são todas compatíveis com o Arduino convencional. Realizaram o controle de velocidade utilizando o PWM (largura de pulso), que faz a variação de tensão na entrada da ponte.

3 DESENVOLVIMENTO

Para o desenvolvimento deste projeto, utilizou-se o Arduino UNO R3 conectado ao Dual Motor *Shield* e, posteriormente, acoplado ao Zumo Chassi. Para o deslocamento, desenvolveu-se uma aplicação responsável por gerar a trajetória e transmitir a mesma, para que o robô realize o deslocamento programado. Nesta seção, estes componentes serão explicados com maiores detalhes.

3.1 DUALMOTOR SHIELD

Como visto anteriormente, para que o robô possa se movimentar é necessário que exista um motor para definir o sentido e a potência que o robô realizará o deslocamento. Assim, para um melhor controle e funcionamento dos motores, optou-se por utilizar um *shield.* Este *shield* é responsável por cuidar do acionamento, da velocidade e o sentido dos motores.

O Dual Motor *Shield* é uma ferramenta que foi desenvolvida pelo Laboratório de Garagem, uma empresa de Barueri que possui uma iniciativa de integração de projetistas onde foi desenvolvida uma rede social para maior contrato entre esses projetistas e proporcionar apoio a esses desenvolvedores independentes. A empresa possui diversos tutoriais para Arduino e também para outros embarcados, como Raspbery PI, Garagino, etc (LABORATORIO DE GARAGEM, 2016).

O Dual Motor *Shield* possui uma biblioteca contendo funções próprias que foram desenvolvidas por Marco Mello, engenheiro do Laboratório de Garagem. O *Shield* pode ser melhor observado na Figura 10. Essa biblioteca contém poucas funções e elas são bem simples de serem utilizadas.

A primeira função é a de instanciar um objeto do tipo DualMotor. Depois disso, temos as funções para acionamento dos motores e recebem dois argumentos, são elas: M1move(v, s) e M2move(v, s), onde v é a velocidade que se deseja obter, variando de 0 a 255; e s é o sentido da rotação do motor, representado por 0 (sentido horário) e 1 (sentido anti-horário). Também possui as funções de parada do motor, são elas: M1parar() e M2parar(). Essas funções não recebem argumento.

Nas Figura 8 e Figura 9, podemos observar a descrição das funções na biblioteca DualMotor e um exemplo de aplicação do código, respectivamente:

Figura 8: Funções principais DualMotor

```
26  void DualMotor::M1move(int velocidade, int sentido)
27  {
28      analogWrite(pwmPinM1, velocidade);
29      digitalWrite(sentidoPinM1, sentido);
30  }
31
32  void DualMotor::M1parar()
33  {
34      analogWrite(pwmPinM1, 0);
35  }
36
37  void DualMotor::M2move(int velocidade, int sentido)
38  {
39      analogWrite(pwmPinM2, velocidade);
40      digitalWrite(sentidoPinM2, sentido);
41  }
42
43  void DualMotor::M2parar()
44  {
45      analogWrite(pwmPinM2, 0);
46  }
```

Fonte: Autoria própria

Figura 9: Exemplo DualMotor

```
1   #include <DualMotor.h>
2
3   int v, s;
4
5   DualMotor dualmotor; //Instancia o DualMotor
6
7   void setup(){
8       dualmotor.M1parar();
9       dualmotor.M2parar();
10  }
11
12  void loop(){
13      dualmotor.M1move(v, s);
14      dualmotor.M2move(v, s);
15  }
```

Fonte: Autoria própria

Assim, com um simples comando, é possível definir o sentido e a velocidade de até dois motores. É importante ressaltar, porém, que como cada motor utilizado possui uma tensão máxima de 6V e o Arduino suporta uma tensão de no máximo de 12V, assim torna-se possível controlar, no máximo, dois motores simultaneamente (LAB DE GARAGEM, 2015). A Figura 10 ilustra esse *shield.*

Figura 10: Dual Motor Shield

Fonte: Lab de Garagem

Este *shield* é muito bem empregado quando utiliza-se ferramentas como o Zumo Chassis (utilizado neste projeto), Magician e Esteira Tamiya, que são outras plataformas robóticas bastante utilizadas pelos projetistas.

A Tabela 2 abaixo apresenta a resposta para a movimentação dos motores, onde 0 significa que o motor estará trabalhando em sentido horário e o 1 indica que o motor estará trabalhando em sentido anti-horário. Para fazer com que o robô vire (direita ou esquerda), como este projeto utiliza um chassi do tipo esteira, é necessário que um motor esteja trabalhando em um sentido e o outro motor esteja trabalhando no sentido inverso:

Tabela 2: Representação dos movimentos dos motores

Motor 1	Motor 2	Movimentação
0	0	Trás
0	1	Esquerda
1	0	Direita
1	1	Frente

Fonte: Autoria própria

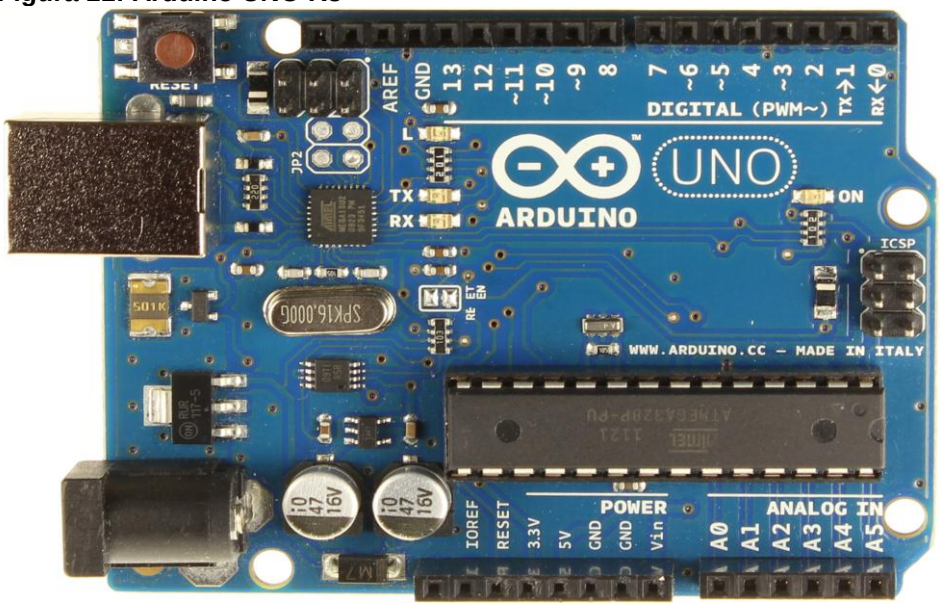
3.2 ARDUINO UNO R3

Para o desenvolvimento deste trabalho, optou-se por utilizar o Arduino Uno R3, por ser uma placa que atende a todas as necessidades do projeto, oferecendo um ótimo custo benefício, já que possui também um dos menores preços entre as placas Arduino disponíveis no mercado.

Esta placa é um dos modelos mais básicos, excelente para aqueles que estão iniciando seus projetos eletrônicos. Ainda assim, ela representa uma poderosa plataforma para prototipação e também oferece uma grande variedade de funções. Tem sido a placa mais utilizada de toda a família Arduino/Genuino (ARDUINO, 2015).

Como visto na Tabela 1, ela possui *clock* de 16 MHz, memória *flash* de 32 KB, 14 pinos de E/S digitais, sendo que 6 destes podem ser usados como saídas PWM, onde utiliza-se a largura de pulso de uma onda quadrada para controlar a potência ou a velocidade dos dados transmitidos. No Arduino, essas entradas estão destacadas com um sinal de “~” na frente da numeração da porta, que pode ser observado na Figura 11. Também possui outras 6 entradas analógicas, conexão via cabo USB ou alimentação de fonte externa e um botão de *reset.*

Figura 11: Arduino UNO R3



Fonte: Arduino Team

O nome “Uno”, que significa literalmente “um” em italiano, foi escolhido como nome dessa placa para marcar o lançamento do *software* Arduino (IDE) 1.0, onde se destacou e foi referência entre os modelos da plataforma Arduino por muito tempo.

Um dos principais motivos para a escolha do Arduino, e deste modelo em específico, é o seu baixo custo, diversidade de aplicações, oferecimento de uma programação simples e limpa.

Isto faz com que o projeto seja simplificado, pois a programação de sistemas embarcados é um pouco complexa, gerando atrasos no projeto, principalmente quando não se tem um contato maior com a área. Como o Arduino possui uma linguagem fácil de trabalhar e portas de fácil utilização, gera-se um ganho de tempo e custo benefício para projeto.

3.3 MICROMOTOR

Para o desenvolvimento deste projeto, foi escolhido um micromotor com caixa de redução de 75:1, que pode ser observado na Figura 12, pois o chassi escolhido já possui entrada para micromotores deste modelo em especial. Ele possui tensão de 6 V.

Esse modelo de motor possui, no mínimo, dois terminais que ao aplicar-se uma tensão em um dos terminais o motor gira em determinado sentido, e ao inverter-se a polaridade o motor girará no sentido inverso (POLOLU-b, 2015).

Figura 12: Micromotor 75:1

Fonte: Pololu

3.4 CHASSI

Para a estrutura física do trabalho, pela falta de um maior conhecimento e experiência em projetos como este, na área eletrônica e seus componentes, portanto dando um maior enfoque na área da programação, no desenvolvimento do algoritmo em si e no controle dos motores, optou-se por utilizar uma estrutura praticamente pronta que fosse de uma relativa facilidade para se trabalhar e desenvolver as aplicações.

Após algumas pesquisas por modelos de chassis prontos, que fosse simples de se utilizar, com um preço de custo não elevado, optou-se por utilizar o Zumo Chassi para o desenvolvimento do projeto.

Este chassi foi desenvolvido pela empresa Pololu, possui menos que 10 cm de tamanho, o que permite a utilização do mesmo em competições como a de Mini Sumo, por exemplo. O corpo principal é composto por um plástico ABS (Um termoplástico utilizado em aplicações que precisam de resistência ao impacto e resistência mecânica, uma boa aparência, dureza e fácil de moldar) preto e possui compartimento para quatro pilhas do tipo AA e suporte para dois micromotores (POLOLU-a, 2015). Na Figura 13 é possível observar maiores detalhes deste chassi.

Figura 13: Zumo Chassis

Fonte: Pololu

Cada lado do chassi possui uma roda dentada livre e uma roda dentada conectada a um micromotor. O compartimento de pilhas se projeta através do chassi

e pode ser acessado a partir do lado superior. Para este tipo de chassi, é recomendada a utilização de micromotores com redução de 50:1, 75:1 e 100:1.

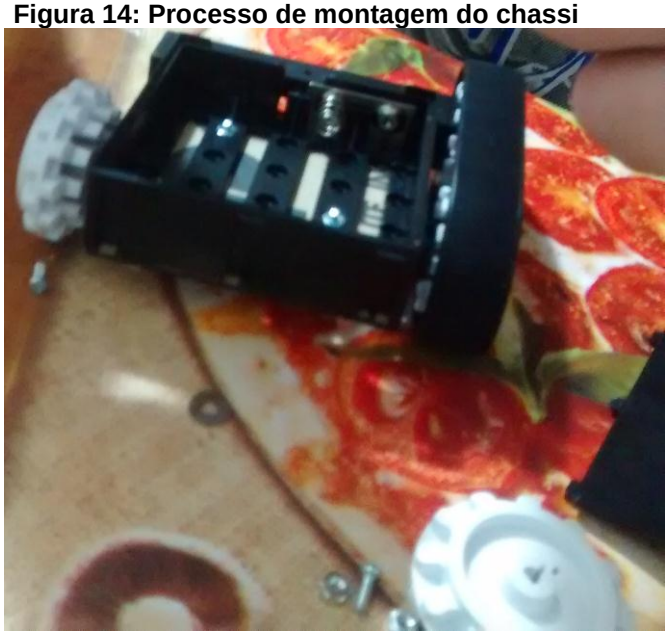
Como citado anteriormente, para esse projeto foi escolhido um micromotor de 75:1. Mas a potência pode variar de acordo com a necessidade de cada projeto, por isso é importante que os projetistas fiquem atentos à escolha do micromotor, de acordo com torque, velocidade e consumo de corrente necessários.

3.5 MONTAGEM FINAL DO ROBÔ

Com os equipamentos necessários em mãos, deu-se início a montagem do robô.

Primeiramente, foi realizada a montagem do chassi. O kit de montagem da Pololu possui uma relativa facilidade, bastando apenas encaixar as peças e parafusar. O chassi já possui um compartimento próprio para se colocar os micromotores. Foi necessário soldar os *jumpers* aos micromotores. Posteriormente, esses *jumpers* foram conectados às entradas específicas do Dual Motor *Shield.* Esse processo pode ser observado na Figura 14.

Figura 14: Processo de montagem do chassi

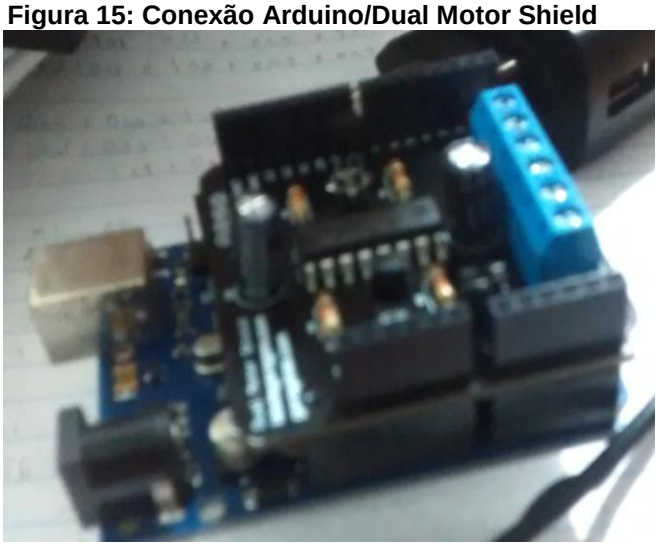


Fonte: Autoria própria

Depois de montado o chassi, o Arduino foi conectado ao Dual Motor *Shield.* As entradas do *Shield* foram conectadas utilizando-se todas as entradas/saídas do

Arduino. O *shield* possui as mesmas conexões e entradas/saídas do Arduino. Isso permite que se utilize mais *shields* ou até mesmo outros *jumpers* ou qualquer outro elemento sem limitação de espaço, dando mais liberdade para o desenvolvimento de aplicações. A Figura 15 ilustra esse processo:

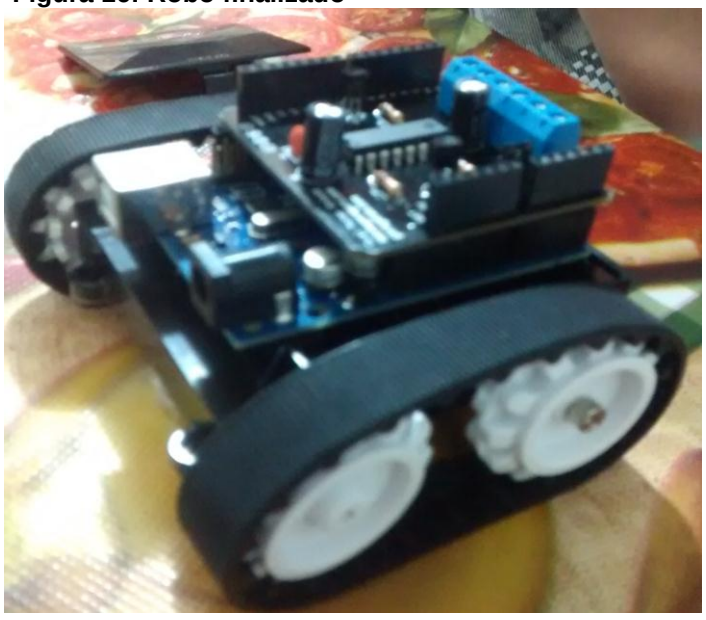
Figura 15: Conexão Arduino/Dual Motor Shield



Fonte: Autoria própria

Após esses dois processos, houve então um último processo para conectar o chassi e o Arduino. A Figura 16 mostra a montagem final do robô:

Figura 16: Robô finalizado



Fonte: Autoria própria

3.6 SOFTWARE DE TRADUÇÃO

Um dos principais problemas para o desenvolvimento deste projeto era a realização do tratamento dos dados e como estes seriam disponibilizados para o robô, de que maneira seria efetuada a leitura desses dados para que o robô realizasse seu deslocamento.

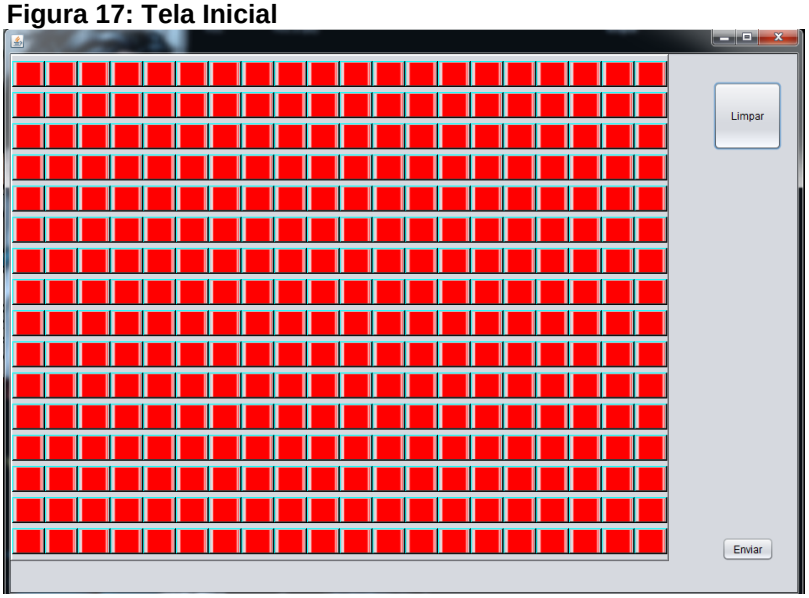
Então, pensando em um método que facilitasse o desenvolvimento das trajetórias, as quais constituirão os conjuntos de testes para o robô, desenvolveu-se uma aplicação utilizando a linguagem Java. Esta aplicação é responsável por gerar a trajetória que o robô deverá percorrer.

Optou-se por utilizar a linguagem Java pela facilidade de implementação, facilidade de uso, e, assim como o Arduino, possui uma grande portabilidade.

O Java é uma linguagem orientada a objetos, o que favorece a reusabilidade, robusta, segura e possui suporte para aplicações concorrentes. Segundo a TIOBE, Java é a linguagem de programação mais utilizada hoje no mundo (TIOBE, 2016).

A aplicação constitui-se em uma tela para o usuário, onde é possível “desenhar” a trajetória escolhida selecionando-se os quadradinhos vermelhos, e dois botões principais: Limpar e Enviar. Esta tela pode ser observada na Figura 17.

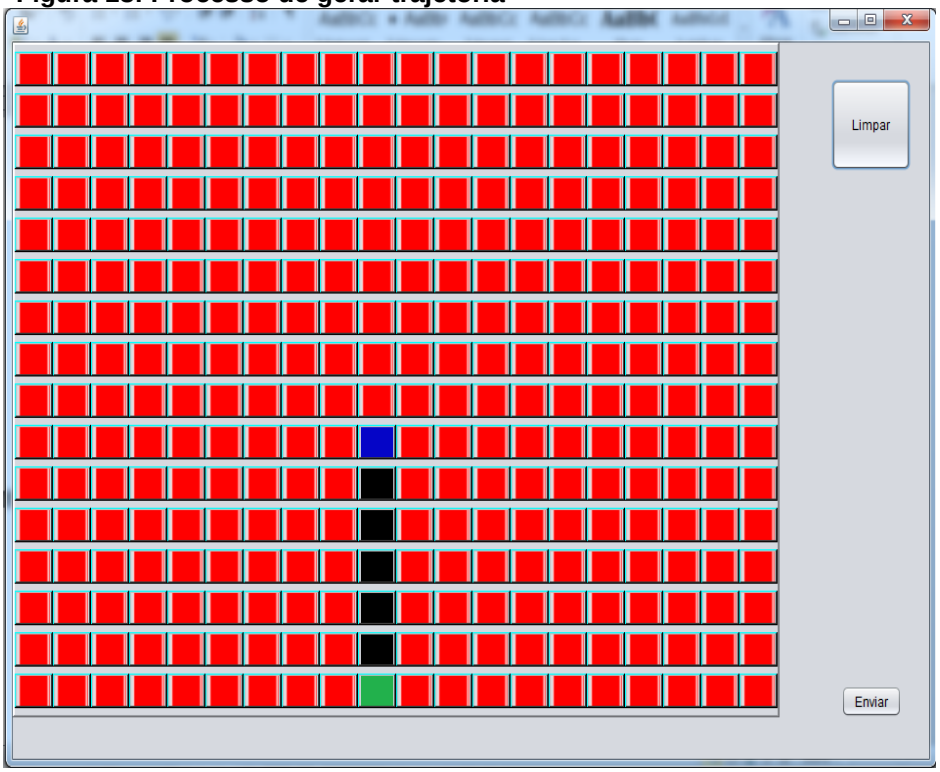
Figura 17: Tela Inicial



Fonte: Autoria Própria

É possível pressionar qualquer um dos quadradinhos vermelhos e ir criando uma trajetória. Ao pressionar o primeiro quadradinho, ele ficará na cor verde indicando que este é o começo da trajetória. Após isso, os demais quadradinhos ficarão na cor preta, indicando que fazem parte daquela trajetória, e o último quadradinho irá ficar na cor azul, marcando o final da trajetória escolhida. Este processo pode ser observado na Figura 18 abaixo:

Figura 18: Processo de gerar trajetória



Fonte: Autoria própria

Para facilitar o desenvolvimento, definiu-se alguns números para representar as trajetórias direcionais, são eles: 0 – cima, 1 – baixo, 2 – direita e 3 – esquerda.

Ao clicar no botão Enviar, abre-se então uma caixa de diálogo, onde é possível selecionar se deseja realmente enviar a trajetória ou cancelar o envio. Ao clicar na opção Sim, a aplicação ira fazer a chamada da função obterTrajetoria(), que pode ser observado na Figura 19, 20 e 21.

Cada vez que um elemento é adicionado à trajetória na tela principal, um índice é adicionado ao *ArrayList* do tipo inteiro denominado **marcados.** Esse *ArrayList* servirá de base para a construção do *ArrayList* do tipo inteiro denominado **trajetóriaFinal.**

Então, após a chamada da função obterTrajetoria(), um laço é iniciado para percorrer os elementos do *ArrayList* **marcados**, e é feita a verificação posição a posição, a partir do segundo elemento. O primeiro elemento é utilizado para definir o sentido inicial do robô, se ele deverá permanecer na posição atual, ou realizar uma conversão para a direita ou esquerda, ou até mesmo um giro de 90º.

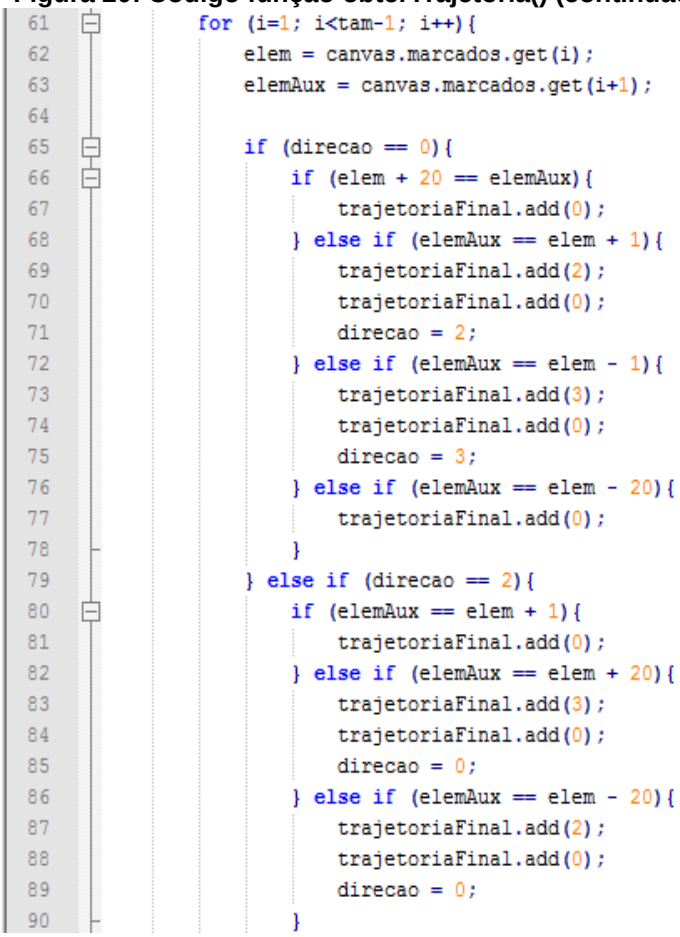
Figura 19: Função obterTrajetoria()

```
33      public void obterTrajetoria(){
34          /*
35           0 - Cima
36           1 - Baixo
37           2 - Direita
38           3 - Esquerda
39          */
40
41          int i=0, elem, elemAux = 0, tam = canvas.marcados.size(), direcao = 0;
42
43          if (tam > 1){
44              elem = canvas.marcados.get(0);
45              elemAux = canvas.marcados.get(1);
46
47              if (elem + 20 == elemAux){
48                  trajetoriaFinal.add(0);
49                  direcao = 0;
50              } else if (elem + 1 == elemAux){
51                  trajetoriaFinal.add(2);
52                  direcao = 2;
53              } else if (elem - 1 == elemAux){
54                  trajetoriaFinal.add(3);
55                  direcao = 3;
56              } else {
57
58              }
59          }
```

Fonte: Autoria própria

Após a verificação do sentido inicial, o laço continua e é verificado o próximo passo que o robô deverá executar. Avançar no mesmo sentido, conversão à direita, esquerda ou giro de 90º. E isso é executado até que o *ArrayList* chegue no seu último elemento.

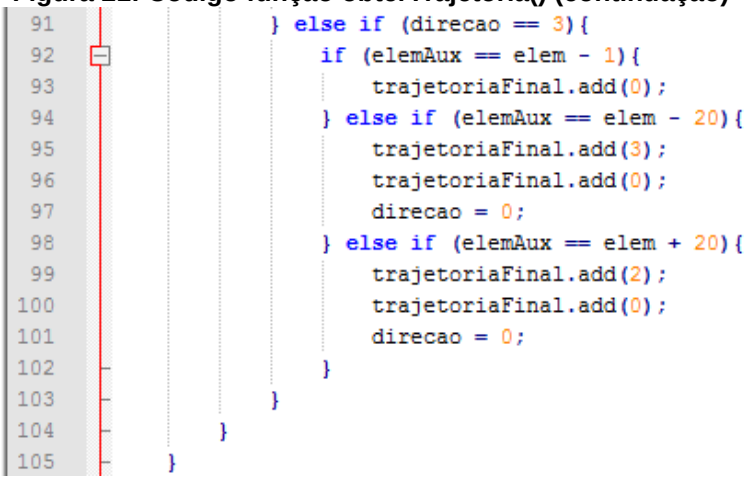
Figura 20: Código função obterTrajetoria() (continuação)



```
61          for (i=1; i<tam-1; i++){
62              elem = canvas.marcados.get(i);
63              elemAux = canvas.marcados.get(i+1);
64
65              if (direcao == 0){
66                  if (elem + 20 == elemAux){
67                      trajetoriaFinal.add(0);
68                  } else if (elemAux == elem + 1){
69                      trajetoriaFinal.add(2);
70                      trajetoriaFinal.add(0);
71                      direcao = 2;
72                  } else if (elemAux == elem - 1){
73                      trajetoriaFinal.add(3);
74                      trajetoriaFinal.add(0);
75                      direcao = 3;
76                  } else if (elemAux == elem - 20){
77                      trajetoriaFinal.add(0);
78                  }
79              } else if (direcao == 2){
80                  if (elemAux == elem + 1){
81                      trajetoriaFinal.add(0);
82                  } else if (elemAux == elem + 20){
83                      trajetoriaFinal.add(3);
84                      trajetoriaFinal.add(0);
85                      direcao = 0;
86                  } else if (elemAux == elem - 20){
87                      trajetoriaFinal.add(2);
88                      trajetoriaFinal.add(0);
89                      direcao = 0;
90                  }
```

Fonte: Autoria própria

Figura 21: Código função obterTrajetoria() (continuação)



```
91              } else if (direcao == 3){
92                  if (elemAux == elem - 1){
93                      trajetoriaFinal.add(0);
94                  } else if (elemAux == elem - 20){
95                      trajetoriaFinal.add(3);
96                      trajetoriaFinal.add(0);
97                      direcao = 0;
98                  } else if (elemAux == elem + 20){
99                      trajetoriaFinal.add(2);
100                     trajetoriaFinal.add(0);
101                     direcao = 0;
102                 }
103             }
104         }
105     }
```

Fonte: Autoria própria

A cada verificação, é adicionado um elemento direcional definido acima (0, 1, 2 ou 3) ao *ArrayList* **trajetoriaFinal**. Se houver um movimento de conversão,

serão adicionados dois elementos ao *ArrayList* **trajetoriaFinal**: um para indicar o elemento direcional e outro para fazê-lo avançar.

Com o término do laço, **trajetoriaFinal** conterá os elementos correspondentes a cada movimento de deslocamento que deverá ser realizado pelo robô. Então é chamada a classe **Trajetoria** que é responsável por criar o arquivo final no formato de extensão **.ino**, que é o formato padrão de um arquivo Arduino.

Primeiramente, a função *criaCabecalho()* é chamada para criar o cabeçalho do arquivo que contém os elementos básicos de um arquivo Arduino, e, logo em seguida, é chamada a função *criaTrajetoria()*, onde são feitas a verificação do tamanho do *ArrayList* trajetoriaFinal, para a criação do vetor de movimentos do robô, e em seguida, os elementos contidos no *ArrayList* serão copiados para o vetor de movimentos do Arduino.

Figura 22: Função criaCabecalho()

```
55      public void criaCabecalho(){
56          gravaArq.printf("#include<DualMotor.h>\n\n"
57                  + "DualMotor dualmotor;\n\n"
58                  + "#define cima 0\n"
59                  + "#define baixo 1\n"
60                  + "#define direita 2\n"
61                  + "#define esquerda 3\n\n"
62                  + "void setup(){\n"
63                  + "     dualmotor.M1parar();\n"
64                  + "     dualmotor.M2parar();\n"
65                  + "     delay(4000);\n}");
66          System.out.println("\nEscrevendo...");
67      }
```

Fonte: Autoria própria

O restante do arquivo é criado, respeitando-se os padrões do Arduino, e o loop principal do Arduino contém um laço para percorrer o vetor de movimentos, onde são feitas as verificações para definir qual movimento de deslocamento deve ser realizado pelo robô. O arquivo é então finalizado.

Figura 23: Função criaTrajetoria()

```
69      public void criaTrajetoria(){
70          int tam = trajetoria.size();
71          gravaArq.printf("\n\nint trajetoria[] = {");
72          for (int j=0; j<tam-1; j++){
73              gravaArq.printf(String.valueOf(trajetoria.get(j)) + ",");
74          }
75          gravaArq.printf(String.valueOf(trajetoria.get(tam-1)) + "};\n");
76          gravaArq.printf("\nvoid loop(){\n"
77                          + "      int tam = " + tam + ", i;\n"
78                          + "      for (i=0; i<tam; i++){\n"
79                          + "           if (trajetoria[i] == cima){\n"
80                          + "                dualmotor.M1move(255, 1);\n"
81                          + "                dualmotor.M2move(248, 1);\n"
82                          + "                delay(1030);\n"
83                          + "           } else if (trajetoria[i] == baixo){\n"
84                          + "                dualmotor.M1move(255, 0);\n"
85                          + "                dualmotor.M2move(248, 0);\n"
86                          + "                delay(1030);\n"
87                          + "           } else if (trajetoria[i] == direita){\n"
88                          + "                dualmotor.M1move(255, 0);\n"
89                          + "                dualmotor.M2move(255, 1);\n"
90                          + "                delay(650);\n"
91                          + "           } else if (trajetoria[i] == esquerda){\n"
92                          + "                dualmotor.M1move(255, 1);\n"
93                          + "                dualmotor.M2move(255, 0);\n"
94                          + "                delay(650);\n"
95                          + "           }\n"
96                          + "      }\n"
97                          + "      dualmotor.M1parar();\n"
98                          + "      dualmotor.M2parar();\n"
99                          + "      delay(10000000);");
100
101         gravaArq.printf("\n}");
102         gravaArq.close();
103     }
```

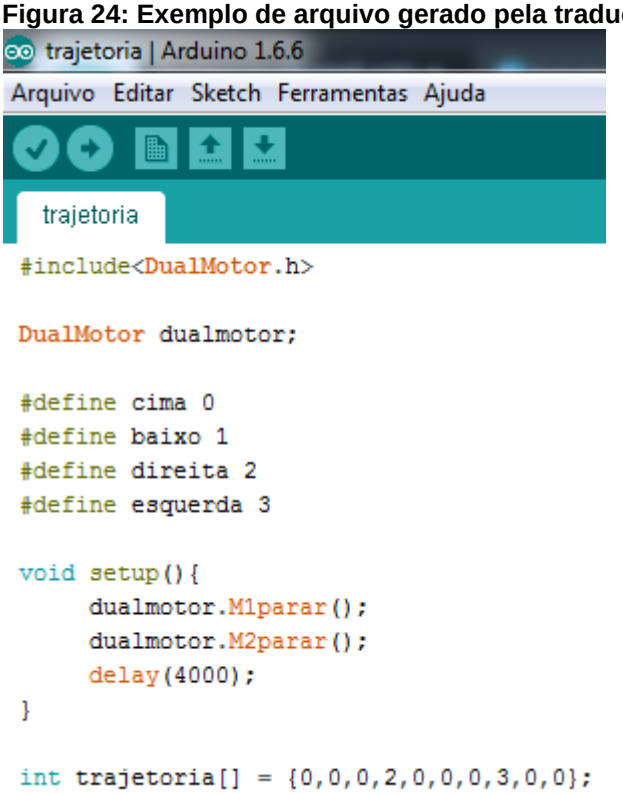
Fonte: Autoria própria

O arquivo gerado para o Arduino contém algumas informações relevantes para o seu funcionamento. Como dito anteriormente, o Dual Motor *Shield* possui algumas funções próprias de sua biblioteca. A utilizada para movimentação são as funções *dualmotor.M1move(velocidade, sentido)* e *dualmotor.M2move(velocidade, sentido)*, então utilizou-se para velocidade o valor máximo (255), e após dizer para o Arduino qual a velocidade e o sentido que será realizado o deslocamento, utiliza-se também um *delay* para quanto tempo essa função será executada.

Utilizou-se o *delay* com valor de 1030 milissegundos, que é equivalente ao deslocamento progressivo de aproximadamente 15 centímetros. Já para os deslocamentos laterais, tanto para à direita como para à esquerda, utilizou-se um *delay* de 635.

Para chegar-se até esses números, foram realizados diversos testes utilizando-se uma fita com 1,5 metros de comprimento. Durante esses testes e diversas medições, foi possível observar e chegar até esses valores.

Figura 24: Exemplo de arquivo gerado pela tradução




```
#include<DualMotor.h>

DualMotor dualmotor;

#define cima 0
#define baixo 1
#define direita 2
#define esquerda 3

void setup(){
      dualmotor.M1parar();
      dualmotor.M2parar();
      delay(4000);
}

int trajetoria[] = {0,0,0,2,0,0,0,3,0,0};
```

Fonte: Autoria própria

Figura 25: Exemplo de arquivo gerado pela tradução (continuação)

```
void loop(){
     int tam = 10, i;
     for (i=0; i<tam; i++){
          if (trajetoria[i] == cima){
               dualmotor.M1move(255, 1);
               dualmotor.M2move(248, 1);
               delay(1030);
          } else if (trajetoria[i] == baixo){
               dualmotor.M1move(255, 0);
               dualmotor.M2move(248, 0);
               delay(1030);
          } else if (trajetoria[i] == direita){
               dualmotor.M1move(255, 0);
               dualmotor.M2move(255, 1);
               delay(650);
          } else if (trajetoria[i] == esquerda){
               dualmotor.M1move(255, 1);
               dualmotor.M2move(255, 0);
               delay(650);
          }
     }
     dualmotor.M1parar();
     dualmotor.M2parar();
     delay(10000000);
}
```

Fonte: Autoria própria

Com o término da criação do arquivo, pode-se conectar o Arduino ao computador para que seja possível transmitir o conteúdo do arquivo para a memória do robô.

Definiu-se um *delay* de quatro segundos para o início do funcionamento do programa, ou seja, após a transferência do arquivo, o Arduino aguardará quatro segundos até que ele seja movido para o laço principal e comece a desenvolver sua movimentação. Assim, é possível ter um tempo maior de reação ao transferir o programa para a memória do Arduino, para que o robô não inicie sem que o mesmo esteja em solo. Esse valor para o *delay* pode ser ajustado, aumentando-se ou decrementando-se este valor, de acordo com necessidades específicas na sua utilização.

4 RESULTADOS

Com a finalização do projeto, foram realizados alguns testes para validação das funcionalidades apresentadas.

Após o desenvolvimento da aplicação em Java e do desenvolvimento do robô, foram geradas algumas trajetórias utilizando a aplicação de tradução e estas foram transferidas para a memória do robô para que ele realizasse o deslocamento para que fosse possível avaliar os resultados obtidos.

Durante o processo, alguns pontos, porém, precisaram ser observados com mais cautela. Mesmo utilizando-se dois micromotores iguais e com mesma potência, observou-se que eles trabalhavam com uma velocidade ligeiramente diferente, fazendo com que o deslocamento progressivo do robô acabasse sendo levemente desviado da sua rota original. No caso, quando colocado para andar em linha reta, o robô acabava cedendo mais para a direita.

Em um deslocamento pequeno, de até 30 centímetros, isso não resultava em uma alteração significativa, mas para um deslocamento maior, a diferença acabava sendo significativa.

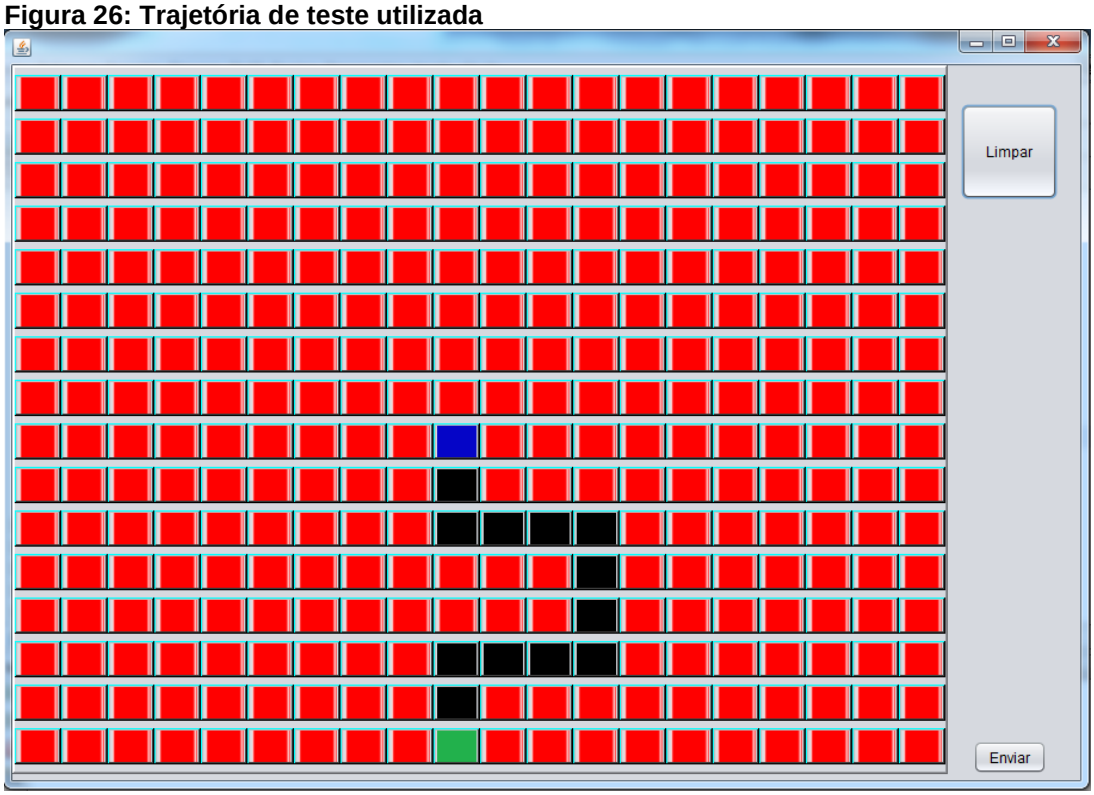
Então foi necessário encontrar um padrão para que o robô conseguisse realizar o deslocamento o mais próximo possível da rota estipulada. Alguns testes e ajustes foram realizados para tentar encontrar esse padrão, através de observação e mudança dos valores dos motores, tanto esquerdo como direito.

Um dos primeiros problemas encontrados foi com a pilha, fator que estava influenciando diretamente no comportamento do robô. Como são utilizadas quatro pilhas AA para enviar energia ao Arduino e ao Dual Motor *Shiled*, ficando duas pilhas enviando energia para cada motor, como duas pilhas estavam mais gastas que as outras duas pilhas, isso fazia com que houvesse o desequilíbrio entre os lados.

Depois de constatado esse problema, as pilhas foram trocadas. Notou-se que houve uma melhora do problema, mas o mesmo ainda persistia e o robô continuava com um leve desvio na sua trajetória. Então foram realizados ajustes manuais na velocidade dos motores, diminuindo-se de um em um a potência do motor esquerdo até encontrar um valor aceitável para que os motores trabalhassem de igual maneira e o robô conseguisse realizar o deslocamento correto.

Pouco a pouco, após diversos testes e ajustes, conseguiu-se chegar a um valor que fosse aceitável, mesmo com uma pequena alteração na trajetória. Como teste, foi utilizada a trajetória apresentada na Figura 26 abaixo. Com essa trajetória, foi possível ajustar a velocidade do motor, para que o robô realizasse esse deslocamento.

Figura 26: Trajetória de teste utilizada



Fonte: Autoria própria

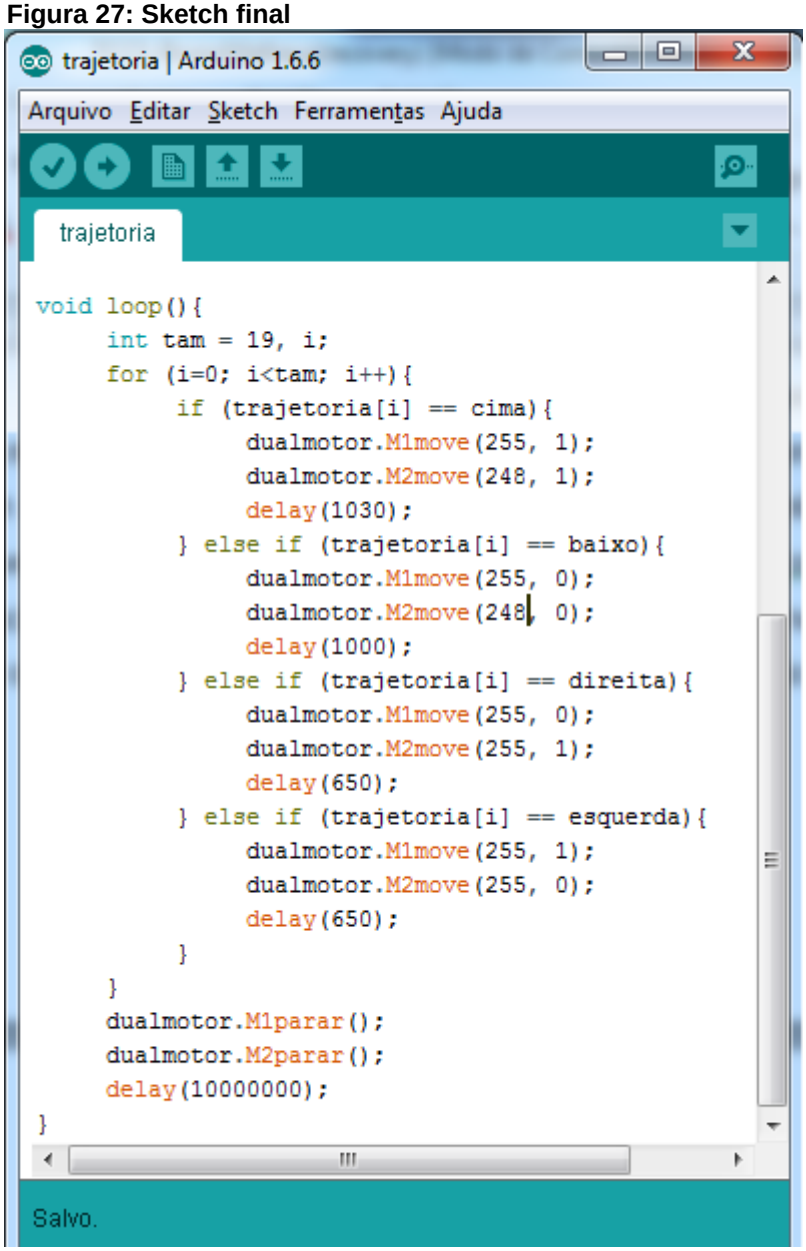
O ajuste foi realizado da seguinte maneira: deixando-se o motor direito (M1) com a velocidade máxima e reduzindo-se pouco a pouco a velocidade do motor esquerdo (M2), até que o robô andasse o mais próximo possível da trajetória teste. Diminuiu-se os valores um a um e o resultado obtido foi: M1 = 255 e M2 = 248. Com esses valores foi possível observar uma melhora nos resultados obtidos.

Para que o robô realizasse os deslocamentos laterais, além da velocidade dos motores, é necessário que o robô realize esse procedimento durante um período de tempo. Com os testes, foi possível chegar ao valor de: 650 milissegundos, tanto para conversão à direita como a esquerda.

Para realizar as duas conversões foram mantidas a velocidade máxima dos motores, pois a maior influência, nesse caso, era do tempo que o robô levaria para girar os motores até ficar na posição correta.

Com isso, chegou-se ao sketch final apresentado na Figura 20.

Figura 27: Sketch final



Fonte: Autoria própria

5 CONCLUSÃO

A robótica é uma área que está constantemente em evolução e a robótica móvel é uma das áreas da robótica que ainda hoje representa grandes desafios.

O objetivo deste trabalho foi desenvolver um robô que fosse capaz de percorrer uma trajetória determinada, utilizando Arduino.

Durante a elaboração do projeto, conseguiu-se desenvolver uma aplicação simples, que facilitou o desenvolvimento das trajetórias e também várias dificuldades foram enfrentadas, principalmente no ajuste da velocidade dos motores. Porém essas dificuldades trouxeram um aprendizado na área de programação de microcontroladores, montagem de *hardwares* simples e um maior contato com a área de métodos científicos de pesquisa.

Porém, mesmo com as dificuldades apresentadas, o projeto acabou atendendo as expectativas. O robô conseguiu realizar as trajetórias propostas, com leves diferenças que acabaram não sendo significativas. Além de o projeto ser extremamente interessante, foi uma experiência gratificante e de grande valia desenvolver este projeto.

Com algumas adaptações no projeto e utilizações de *shields,* entre outras ideias, este projeto pode ser utilizado para outras diversas aplicações, como detecção de mudança de trajetória e de obstáculos.

Espera-se, portanto, que este projeto seja de grande valia para outros projetistas entusiastas, sejam iniciantes ou não, na robótica móvel.

6 TRABALHOS FUTUROS

Para que a aplicação desenvolvida em Java fosse melhor empregada, dando maior agilidade ao processo de enviar as trajetórias, para o robô realizar seu deslocamento, pensou-se em utilizar duas expansões que agilizassem esse processo.

- Expansão de *WiFi*: permite criar uma comunicação fácil e ágil entre o microcontrolador, computador, ou qualquer outro módulo que possua uma porta serial. Permitiria que a trajetória fosse enviada simultaneamente ao Arduino. Para isso, seria necessário utilizar o Xbee *Shield* ou Protobee *Shield* + Xbee Módulo. A grande vantagem de utilizar essa expansão é o seu alcance, podendo chegar até 100 metros;
- Expansão *Bluetooth*: assim como a expansão de *WiFi*, permite a comunicação serial de forma ágil entre um celular, ou qualquer outro dispositivo que realize a comunicação *bluetooth*, e o Arduino, podendo transferir rapidamente a trajetória para o robô. Suas desvantagens em relação ao módulo de *WiFi* são: possui uma velocidade menor no envio de dados e o seu alcance máximo de 10 metros sem obstáculos;
- Sensores: permitiria que o robô pudesse se tornar mais autônomo, podendo desviar de obstáculos, quando necessário, e até mesmo configurar sua velocidade e corrigir sua trajetória. Os sensores mais comuns utilizados são: infravermelhos e ultrassônicos;
- Desenvolvimento de uma aplicação *mobile:* assim como a aplicação em Java, o desenvolvimento de uma aplicação *mobile* facilitaria o processo de desenvolvimento de trajetórias, e, utilizando-se com algum dos *shields* citados, deixando o processo mais ágil.

REFERÊNCIAS

ADÔRNO, B. V.; BORGES, G. A. **Um método de planejamento de trajetória para robôs móveis através de passeios aleatórios adaptativos e mapa de rotas.** In: XVI Congresso Brasileiro de Automática. 2006. p. 1-6.

ANDRADE, D.S. **Projeto: Robô Seguidor de Linha.** 2013. Laboratório de Eletrônica Aplicada EEL 7300. Departamento de Engenharia Elétrica. Universidade Federal de Santa Catarina – Florianópolis, 2013.

ARAUJO, L. F. M. **Desenvolvimento de um Sistema Embarcado para Controle de Um Robô Móvel.** 2013. 59 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação. Universidade Federal de Lavras, 2013.

ARDUINO TEAM. **Arduino Products.** Disponível em: <https://www.arduino.cc/en/Main/Products>. Acesso em: 16 de Novembro de 2015.

BANZI, M. **Getting Started with Arduino.** 2. ed. Highway North: O'Reilly Media, Inc, 2011.

BAÚ DA ELETRÔNICA. **Micromotor com Caixa de Redução de Metal 75:1 LP – Pololu.** 2016. Disponível em: <http://www.baudaeletronica.com.br/micro-motor-com-caixa-de-reducao-de-metal-75-1-hp-pololu.html>. Acesso em: 30 de Setembro de 2016.

BENTES, L. M. A. **Sistema de Segurança Veicular com uso de GPS baseado em Arduino.** 2013. 114 f. Trabalho de Conclusão de Curso (Graduação) – Curso Superior de Engenharia de Computação. Universidade do Estado do Amazonas. Manaus, 2013.

ČAPEK, K; PLAYFAIR, N; SELVER, P. **RUR (Rossum's universal robots): a Fantastic Melodrama in Three Acts and an Epliogue.** Doubleday, Page, 1923.

EVANS, M.; HOCHENBAUM J; NOBLE, J. **Arduino em Ação.** 1. ed. São Paulo: Novatec Editora, 2013.

FOROUZAN, B.A. **Comunicação de Dados e Redes de Computadores.** 3. ed. Porto Alegre: Bookman, 2006.

GIOPPO, L. L; HIGASKINO, M. M. K; COSTA, R. F.; MEIRA, W. H. T. **Robô Seguidor de Linha.** 2009. 35 f. Trabalho de Conclusão de Curso (Graduação) – Curso Superior de Engenharia da Computação. Universidade Tecnológica Federal do Paraná. Curitiba, 2009.

HOBBY MODELISMO. **Chassis Com Tração Esteira C/ Caixa de Redução e Motor Mabuchi.** 2016. Disponível em: <http://www.hobbymodelismo.com.br/detalhe.asp?cod=TAM70108>. Acesso em: 30 de Setembro de 2016.

INTERNATIONAL FEDERATION OF ROBOTICS. **Industrial Robot Statistics.** 2015. Disponível em: <http://www.ifr.org/industrial-robots/statistics/>. Acesso em: 30 de Março de 2016.

KURSHNER, D. **The Making of Arduino.** 2011. Disponível em: <http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino>. Acesso em: 12 de Novembro de 2015.

MAKE US OF. **8 Arduino Robots You Can Build For Less than $125.** 2015. Disponível em: <http://www.makeuseof.com/tag/8-arduino-robots-can-build-less-125/>. Acesso em: 30 de Setembro de 2016.

MCROBERTS, M. **Arduino básico**. 1. ed. São Paulo: Novatec Editora, 2011.

MEDINA, B.V.O. **Sistema de Visão Computacional Aplicado a Um Robô Cilíndrico Acionado Pneumaticamente.** 2015. 132 f. Dissertação (Mestrado) – Programa de Pós Graduação em Engenharia Mecânica, Universidade Federal do Rio Grande do Sul. Porto Alegre, 2015.

MENEGUELE, B. E. O; FERREIRA, F. P; ARCANJO, V. S. **Robô Explorador de Labirintos 2D.** 2011. 52 f. Monografia do Projeto de Oficinas de Integração 2 – Curso Superior de Engenharia da Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION – NASA. **NASA's Curiosity Rover Team Confirms Ancient Lake on Mars.** Disponível em: <http://www.nasa.gov/feature/jpl/nasas-curiosity-rover-team-confirms-ancient-lakes-on-mars>. Acesso em: 16 de Outubro de 2016.

NICE, K. **O Que é torque?** Disponível em: <http://ciencia.hsw.uol.com.br/forca-potencia-torque-energia4.htm>. Acesso em: 24 de Novembro de 2015.

NOSSOS ROBÔS. **Monte Seu Robô Com Arduino.** 2012. Disponível em: <http://nossosrobos.blogspot.com.br/2012/09/monte-seu-robo-com-arduino.html>. Acesso em: 30 de Setembro de 2016.

PEREZ, A. L. F. et al. **Uso da Plataforma Arduino para o Ensino e o Aprendizado de Robótica.** In: International Conference on Interactive Computer aided Blended Learning (ICBL). 2013.

PIERI, E.R. **Curso de Robótica Móvel.** 2012. 141 f. Programa de Pós Graduação em Engenharia Elétrica. Universidade Federal de Santa Catarina, Florianópolis, 2012.

POLOLU-a. **Zumo Chassi Kit.** Disponível em: <https://www.pololu.com/product/1418>. Acesso em: 17 de Novembro de 2015.

POLOLU-b. **75:1 Micro Motor Gearmotor HP.** Disponível em: <https://www.pololu.com/product/2361>. Acesso em: 17 de Novembro de 2015.

RIBEIRO, C.R. **RobôCarochinha: Um Estudo Qualitativo sobre a Robótica Educativa no 1º Ciclo do Ensino Básico.** 2006. 207 f. Dissertação (Mestrado). – Instituto de Educação e Tecnologia, Universidade do Minho. Braga, 2006.

SCHLATTER, B. RPGista. **Os kakuris tamuranianos.** Porto Alegre: 2013. Disponível em: <http://rpgista.com.br/2013/06/02/os-karakuri-tamuranianos/>. Acesso em: 25 de Outubro de 2015.

SENESE, M. Make. **Arduino annouces new brand, Genuino, manufacturing partnership with Adafruit.** Disponível em: <http://makezine.com/2015/05/16/arduino-adafruit-manufacturing-genuino/> Acesso em: 12 de Novembro de 2015.

TERRA NOTÍCIAS. **Robô reúne mais provas da existência de antigo lago em Marte.** Disponível em: < http://noticias.terra.com.br/ciencia/espaco/robo-reune-mais-provas-da-existencia-de-antigo-lago-em-marte,e374244019b2a410VgnCLD200000b2bf46d0RCRD.html> Acesso em: 17 de Novembro de 2015.

SOUZA-a, F. Embarcados. **Arduino – Saídas PWM.** Disponível em: <http://www.embarcados.com.br/arduino-saidas-pwm/> Acesso em: 30 de Setembro de 2016.

SOUZA-b, F. Embarcados. **Arduino – Entradas Analógicas.** Disponível em: <http://www.embarcados.com.br/arduino-entradas-analogicas/> Acesso em: 30 de Setembro de 2016.

TIOBE. **TIOBE Index for September 2016.** Disponível em: <http://www.tiobe.com/tiobe-index/>. Acesso em: 04 de Outubro de 2016.

UNIVERSITY OF MICHIGAN. ***Robots In Literature – R.U.R.*** 2016. Disponível em: <http://www.umich.edu/~engb415/literature/pontee/RUR/RURsmry.html> Acesso em: 10 de Abril de 2016.

VIEIRA, F. C. **Controle dinâmico de robôs móveis com acionamento diferencial.** 2005. 106 f. Dissertação (Mestrado) - Programa de Pós Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte. Natal, 2005.

WILMSHURST, T. **Designing Embedded Systems with PIC Microcontrollers - Principles and applications.** 1. ed. Londres: Elsevier, 2007.

WOLF, D. F.; do Valle Simões, E.; Osório, F. S.; Junior, O. T. **Robótica móvel inteligente: Da simulação às aplicações no mundo real**. In *Mini-Curso: Jornada de Atualização em Informática (JAI), Congresso da SBC,* 2009.