UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO CIÊNCIAS EXATAS E NATURAIS
CURSO DE BACHARELADO EM CIÊNCIA E TECNOLOGIA

ANDERSON VIEIRA FERNANDES

ALGORITMO E DESENVOLVIMENTO DE UM PROTÓTIPO DE CADEIRA DE RODAS INTELIGENTE NO AUXÍLIO PARA DEFICIENTES

CARAÚBAS – RN

2016

ANDERSON VIEIRA FERNANDES

ALGORITMO E DESENVOLVIMENTO DE UM PROTÓTIPO DE CADEIRA DE RODAS INTELIGENTE NO AUXÍLIO PARA DEFICIENTES

Monografia apresentada a Universidade Federal Rural do Semiárido, como parte dos requisitos para obtenção do título de Bacharel em Ciência e Tecnologia

Orientador: Hugo Michel Camara de Azevedo Maia, Prof. Dr.

CARAÚBAS – RN

2016

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central Orlando Teixeira (BCOT)
Setor de Informação e Referência (SIR)

F363a Fernandes, Anderson Vieira.

ALGORITMO E DESENVOLVIMENTO DE UM PROTÓTIPO DE CADEIRA DE RODAS INTELIGENTE NO AUXÍLIO PARA DEFICIENTES / Anderson Vieira Fernandes. - 2016. 74 f. : il.

Orientador: Hugo Michel Camara de Azevedo Maia.

Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Bacharelado em Ciência e Tecnologia, 2016.

1. Protótipo. 2. Automação. 3. Cadeira de rodas. 4. Arduino.

I. Maia , Hugo Michel Camara de Azevedo, orient. II

Bibliotecário-Documentalista
Nome do profissional, Bib. Me. (CRB-15/10.000)

ANDERSON VIEIRA FERNANDES

ALGORITMO E DESENVOLVIMENTO DE UM PROTÓTIPO DE CADEIRA DE RODAS INTELIGENTE NO AUXÍLIO PARA DEFICIENTES

Monografia apresentada no Campus Caraúbas para a obtenção do título de Bacharel em Ciências e Tecnologia.

Defendida em: 20 / 05 / 2016 .

BANCA EXAMINADORA

Hugo Michel C. de Azevedo Maia

Hugo Michel de Azevedo Maia, Prof. Dr. (UFERSA)
Presidente

José Ailton L. Barbosa Júnior, Prof Ms. (UFERSA)
Membro Examinador

Tania Luna Laura, Prof. Dra. (UFERSA)
Membro Examinador

Eliel Poggi dos Santos

Eliel Poggi dos Santos, Prof. MS. (UFERSA)
Membro Examinador

Dedico,

A Deus, pela vida e por todos os aprendizados que me concedeu até hoje.

Aos meus pais, Alexandre Fernandes Neto e Consolação Vieira Fernandes.

A minha irmã Alessandra Vieira Fernandes.

E a todos os familiares.

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.” (Arthur Schopenhauer).

AGRADECIMENTOS

Agradeço a Deus, em primeiro lugar, por ter me guiado e proporcionado sabedoria ao decorrer do curso.

Aos meus pais, por todo o amor, educação e confiança. E por sempre mostrar que o estudo é a porta para o sucesso profissional.

A minha irmã Alessandra Fernandes, por ser meu espelho para vencer os desafios da vida.

A minha namorada Vivian de Araújo, por estar sempre ao meu lado mesmo eu estando distante e por apoiar tudo que faço.

Ao professor Dr. Hugo Michel Maia, por ter me orientado e transmitido o conhecimento para a realização deste trabalho.

A amiga Patrícia que se dispôs a ajudar na programação do protótipo. Ao amigo Alefjohn por ter me ajudado no desenvolvimento da parte mecânica do robô.

A todos os amigos e colegas da graduação que estiveram presentes na minha vida ao longo do curso.

RESUMO

Neste trabalho, foi desenvolvido um algoritmo computacional e um protótipo inteligente de cadeira de rodas para auxiliar pessoas com deficiência motora. Foram utilizados elementos e técnicas de computação, automação e eletrônica que são de grande importância para o desenvolvimento de um protótipo que possibilite desviar de possíveis obstáculos que possa causar colisão ao usuário. Primeiramente foi feito um contexto histórico da automação, bem como, suas principais características. Em seguida, corroborando com este contexto, abordou-se o desenvolvimento das cadeiras de rodas e a necessidade de sua evolução. Finalmente, foi descrito a implementação do protótipo utilizando a plataforma Arduino UNO, e os resultados com testes realizados com o protótipo para verificar a funcionalidade dos sensores e atuadores.

PALAVRAS-CHAVE: Protótipo, Automação, Cadeira de rodas, Arduino.

ABSTRACT

In this work, was developed an algorithm for implementation of prototype wheelchair intelligent, for to assist for locomotor disabilities people. There the use of elements and computational techniques, automation, and electronics that are great importantance for development of a prototype which allows avert of possible obstacles that may cause collision the user. First the all, is approached the historical context of automation, as well, yours main features. Then, corroborating this context, It is approached the development of wheelchairs and the need for evolution. Finally described is the prototype application using the Arduino, and results of tests carried out with the prototype to verify the functionality of the sensors and actuators.

Keywords: Prototype, Automatio, Wheelchairs, Arduino.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico - Digital
CI	Integrated Circuit (Circuito Integrado)
CC	Direct Current (Corrente Contínua)
CA	Alternat Current (Corrente Alternada)
I/O	Input/Output (Entrada/Saída)
IR	Infrared (Infrevermelho)
kHz	Kilohertz
MDF	Medium Density Fiberboard (Madeira de Média Densidade)
PSD	Position Sensing Device (Dispositivo de Monitoramento de Posição)
PWM	Pulse Width Modulation (Modulação por Largura de Pulso)
USB	Universal Serial Bus (Barramento Serial Universal)

LISTA DE SÍMBOLOS

T	Período da forma de onda
t_p	Duração do pulso em nível lógico
$V(t)$	Dependência temporal da tensão
V_{cc}	Tempo da onda em nível alto
V_{dc}	Tensão média de uma forma de onda
V_{pulso}	Tensão de pulso do sinal PWM

SUMÁRIO

1. INTRODUÇÃO

Automação teve início por volta da época de 3500 e 3200 a.C. com a utilização da roda, objetivando a simplificação do trabalho humano, de modo a substituir o esforço muscular por outros meios de mecanismo. Hoje em dia, à automação tem a função principal de melhorar a produtividade, bem como a qualidade da produção de grandes empresas nos processos considerados repetitivos (SILVA, 2013).

Atualmente a conjuntura da automação é criar novas habilidades de implementação, dispositivos que suportem maior capacidade computacional de processamento, bem como, ampliar o desenvolvimento de sistemas embarcados, além da criação em massa de dispositivos simples e de baixo custo, proporcionando várias soluções eficazes na resolução de problemas, surgindo uma vasta possibilidade de manipulação desses circuitos em diversas áreas.

1.1 Problema e Justificativa

No Brasil, 45,6 milhões de pessoas são portadoras de algum tipo de deficiência física. Em particular, a região nordeste se destaca por apresentar os maiores índices para todo o conjunto de deficiências: visual, motora, auditiva e mental. Esse número corresponde a 23,9% da população brasileira, sendo que 7% são pessoas portadoras de deficiência motora, ou seja, problemas intuitivos ou adquiridos que afetam a mobilidade e coordenação dos indivíduos (CENSO, 2010).

Ainda segundo o Censo (2010), dentre as diversas deficiências motoras, se destacam: a dificuldade de locomoção, correspondente a cerca de 13,3 milhões de pessoas ou 7% da população pesquisada; a deficiência motora severa que gera dificuldade e incapacidade de locomoção, afetando 4,4 milhões de pessoas, dentre as quais 734,4 mil não conseguem se movimentar ou subir escadas, por exemplo. Nesse sentido, essa deficiência torna as pessoas dependentes de auxílio ou instrumentos de locomoção e, atualmente, os sistemas de automação para deficientes físicos possui um alto custo, dificultando o seu acesso aos usuários de baixa renda.

A cadeira de rodas é um equipamento que foi desenvolvido para auxiliar no deslocamento de pessoas que apresentam impossibilidade, temporária ou definitiva, de deslocar-se utilizando os membros inferiores (MOREIRA, 2012).

Com o avanço na robótica industrial, as cadeiras de rodas vêm evoluindo progressivamente. Atualmente podem ser encontrados vários modelos de cadeiras de rodas, desde as manuais, adaptadas à prática de esportes, e as motorizadas (Figura 1), sendo esta última a mais desejada pelos deficientes devido ao seu fácil manuseio (SILVA; DEL'ACQUA, 2012).

A cadeira de rodas motorizada é geralmente indicada às pessoas que não possuem força ou coordenação para acionar manualmente uma cadeira de rodas convencional. Dessa forma, o uso da cadeira de rodas motorizada está diretamente ligado as condições motora e cognitiva do usuário, sendo restrito aquele que não possui uma coordenação adequada para a utilização desse equipamento. É sabido que se uma cadeira de rodas se desgovernar poderá causar acidente tanto para o usuário, quanto para as pessoas que estão em sua volta (MOREIRA, 2012).

Figura 1: Cadeira de rodas motorizada.

Fonte: http://www.rodaviva.com.br/cadeiras-de-rodas-motorizada.htm.

Segundo Moreira (2012), existe uma alta demanda na produção e potencialização de produtos que atendam às necessidades de portadores de deficiência motora, melhorando dessa forma, a qualidade de vida e consequentemente a integração no convívio social. Essa necessidade se abrange não somente para países desenvolvidos, mas está presente nos países em desenvolvimento como o Brasil. Portanto, torna-se necessário o uso do conhecimento técnico e científico para contribuir na melhoria da qualidade de vida dessa parte da população que na maioria das vezes é excluída da sociedade.

1.2 Objetivos

O objetivo desse trabalho é desenvolver um protótipo inteligente de uma cadeira de rodas que identifique possíveis obstáculos que possam comprometer seu deslocamento ou gerar uma possível colisão.

1.2.1 Objetivos Específicos

a) Desenvolver um *firmware* que seja compatível a uma cadeira de rodas inteligente;
b) Criar um protótipo de uma cadeira de rodas, para testar o software desenvolvido;
c) Desenvolver uma placa de potência para o protótipo.

1.3 Organização do Trabalho

Este trabalho foi organizado em 4 capítulos, como mostrado abaixo:

O capítulo 2 apresenta uma revisão bibliográfica sobre automação, bem como, sensores e atuadores que são componentes essenciais no processo de automatização.

No capítulo 3 será apresentado a plataforma Arduino, com ênfase na plataforma Arduino Uno, sistema embarcado, no qual será exposto as principais características tanto do ambiente de desenvolvimento do software, sua linguagem de programação, e a sua estrutura física da placa.

No capítulo 4 será feita a descrição da montagem do protótipo da cadeira de rodas inteligente, assim como os materiais que foram utilizados.

No capítulo 5 será apresentado os resultados experimentais tais como, testes no protótipo, para verificar a partir do *firmware* e do hardware implementado, o comportamento dos sensores e atuadores.

O capítulo 6 descreve a conclusão do trabalho, fazendo uma abordagem geral do que foi feito e associando os resultados alcançados frente às expectativas e às possíveis sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, será apresentado uma revisão bibliográfica das áreas na qual este trabalho está inserido, citando alguns aspectos históricos, bem como, exemplos das diversas aplicações inseridos na automação.

2.1 Automação

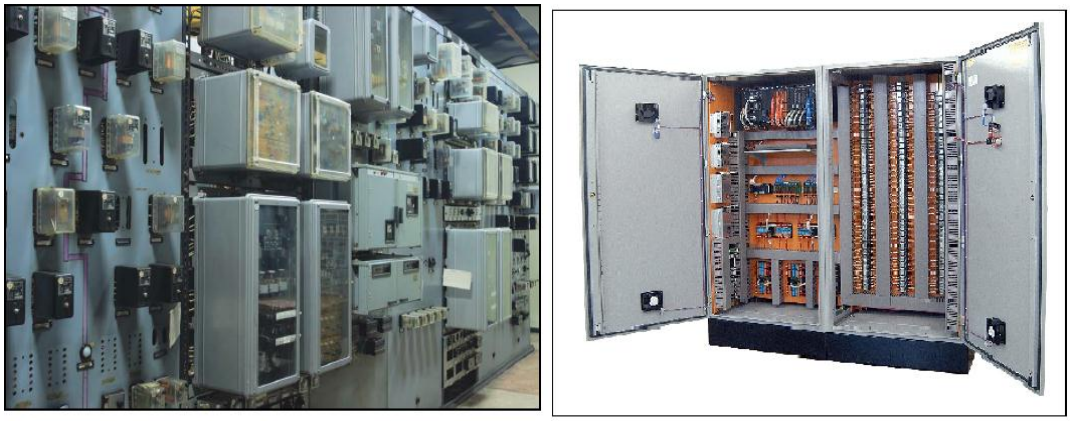
Automação é o conjunto que abrange as máquinas e as formas de como as operações serão executadas para aumentar a eficiência de produção em um determinado sistema. Esse termo pode ser definido também como sendo o controle eletrônico que substitui em parte, ou todo, o controle do cérebro humano (AIHARA, 2005).

O principal exemplo que pode ser usado nesse contexto é a automação industrial que surgiu no século XVIII, com a eclosão das grandes revoluções industriais. Segundo Moreira (2012), entende-se que o processo ocorre da seguinte forma: vários mecanismos de medidas de uma indústria enviam os sinais recolhidos para um computador, que por sua vez, compara esses sinais com outros ditos ideais e dessa forma, realiza uma série de cálculo enviando respostas para os equipamentos que tem a função de atuadores, com o intuito de alcançar uma produção de máxima eficiente.

Atualmente existem diversos dispositivos na automação industrial, porém o mais importante e consequentemente mais usado é o controlador lógico programável (CLP). Esse dispositivo veio a substituir os relés que desde então eram os principais mecanismos e execução, armazenado em grandes armários das industrias, interligando circuitos elétricos com vastas afiações (PAREDE; GOMES, 2011).

A Figura 2 mostra a proporção dos armários de relés frente aos armários de CLP presentes nas indústrias.

Figura 2: a) armário de relés; b) armário de CLPs.



a) b)

Fonte: a) Parede; Gomes, 2011; b) https://upload.wikimedia.org/wikipedia/commons/thumb/2/2f/Paineldeclp.jpg/220px-Paineldeclp.jpg.

2.2 Componentes da Automação

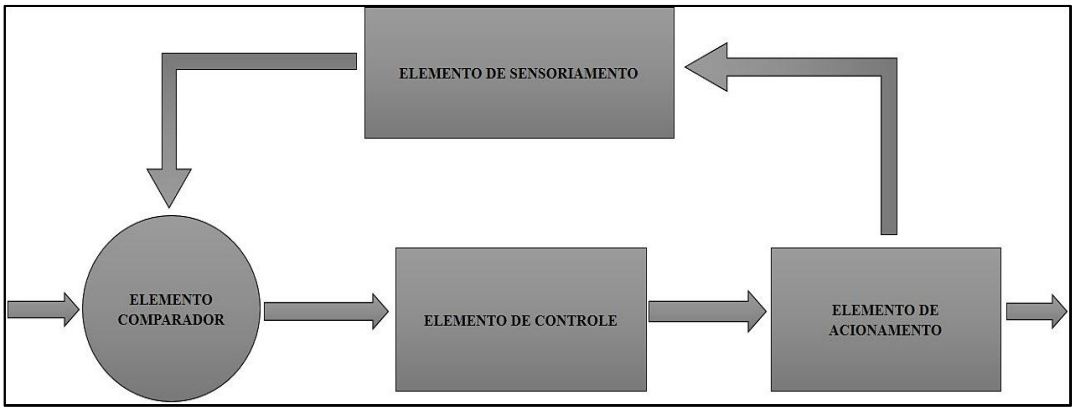
De acordo com Moreira (2012), os principais componentes de automação utilizados em grandes empresas como indústria automobilísticas, petroquímicas e até mesmo em supermercados, necessitam de várias repetições de tarefas que na maioria dos casos são bastante complexas. Portanto, qualquer sistema de automação se comporta da seguinte forma:

a) **Acionamento:** definido como sendo o dispositivo que recebe o sinal do controle e gera um sinal com potência suficiente para atuar sobre o sistema. Como motores elétricos e pistões;
b) **Sensoriamento:** calcula a eficiência do sistema de automação. Exemplo, termopares para medição de temperatura, *encooders* para medição de velocidade;
c) **Controle:** com base na informação dos sensores, o dispositivo de controle, adequa o funcionamento dos atuadores. Exemplo, uma sirene que é acionada quando algo está errado em um determinado ambiente;
d) **Comparador:** relaciona os valores recebidos com os valores pré-estabelecidos, mandando informação para o sistema. Como exemplo, os softwares de automação.

A Figura 3 exemplifica o comportamento do sistema de automação. O elemento comparador recebe os sinais medidos pelo elemento de sensoriamento e compara os valores recebidos com os valores pré-estabelecidos armazenado na memória do computador

programado pelo software. Desta forma, se for necessário ativar algum atuador, o elemento comparador envia um sinal para o elemento de controle, que regula os atuadores por meio do elemento de acionamento. Este ciclo permanece constante, com o elemento de sensoriamento que realimenta o circuito enviando os sinais com as informações encontradas no ambiente para o elemento comparador (MOREIRA, 2012).

Figura 3: Elementos da automação.



Fonte: Próprio do autor.

Ainda segundo Moreira (2012), existem várias classificações na automação que variam com as áreas de aplicações. Como por exemplo, automação comercial, industrial, de transporte, sendo que cada uma tem características diferentes e todas com o mesmo objetivo, maximizar a eficiência nas determinadas aplicações com a mínima intervenção do homem.

2.3 Sensores

Segundo Fonseca (2006), sensores são dispositivos que tem a capacidade de perceber uma grandeza física e transmitir para um indicador (termômetro, ponteiro de velocidade, etc.) ou para um controlador. Dessa forma, esses dispositivos são sensíveis a algum tipo de energia do ambiente, que detectam características de pressão, térmicas, luminosas, sonora, cinéticas, químicas e entre outras.

De acordo com Patsko (2006), sensores podem ser definido como “aquilo que sente”. No ramo da eletrônica, esses dispositivos podem serem vistos como qualquer componente ou

circuito eletrônico que permite a análise de uma determinada condição do ambiente, que possa ser simples como temperatura ou luminosidade, ou até mesmo complexas como detectar partículas subatômicas e radiação cósmica.

Os sensores agem relacionando informações recebidas pelo ambiente (como velocidade, aceleração, pressão, etc.) com sua própria sensibilidade. Em alguns casos, esses dispositivos não possuem características elétricas necessárias para ser utilizado em um sistema de controle, dessa forma, se faz necessário manipular o sinal de saída antes da sua leitura neste sistema. Por exemplo, quando o sensor apresenta uma tensão de saída muito baixa quando ativado por uma energia externa, se faz necessário à amplificação desse sinal para que o controlador possa ler tal informação (WENDLING, 2010).

Os sensores também podem ser definidos como sendo dispositivos que monitoram o ambiente e informam ao circuito eletrônico quando houver alguma perturbação no ambiente de controle. Portanto, a partir da informação dos sensores, o circuito eletrônico executa ações pré-determinadas para tal situação (MOREIRA, 2012).

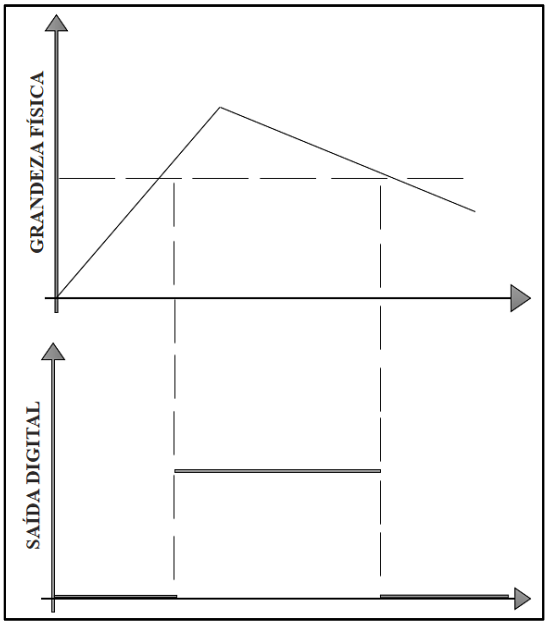
Ainda segundo Wendling (2010), existem dois tipos de sensores, os sensores analógicos e os digitais. O primeiro assume diversos valores no sinal de saída ao longo do tempo, desde que esteja dentro da sua faixa de operação. O segundo, só assume dois valores no sinal de saída (0 ou 1) ao longo do tempo. É sabido que não existem grandezas físicas que consigam atribuir-se a esses valores, 0 ou 1, porém são apresentadas dessa forma a um sistema de controle, depois de serem convertidas, geralmente por um comparador no circuito eletrônico.

Existe uma série de caraterísticas que devem ser levadas a fundo na hora da escolha do sensor que será utilizado para à determinada aplicação. As principais características dos sensores são: tipos de saída, linearidade, alcance e velocidade de resposta.

a) Tipos de Saída:

Digital ou Binária: a saída do dispositivo recebe apenas dois valores lógicos (0 ou 1). Esse tipo de saída só determina se uma grandeza física atingiu ou não um valor pré-determinado. A Figura 4 ilustra a saída de um sensor digital em função da variação de entrada ao longo do tempo.

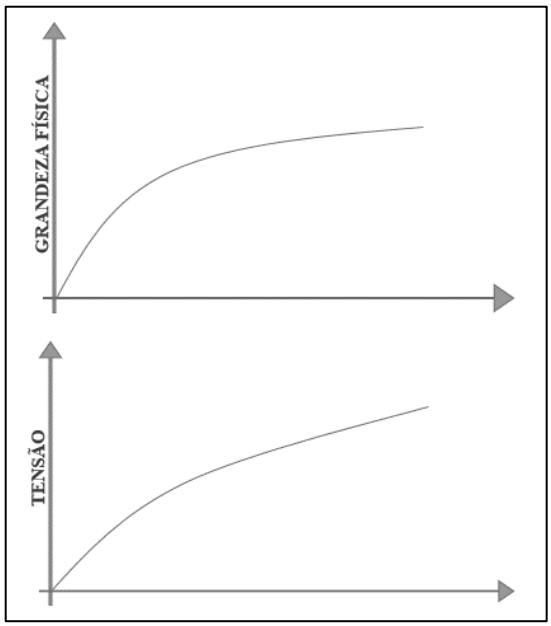
Figura 4: Saída Digital ou Binária de um Sensor.



Fonte: Próprio Autor.

Analógica: esse tipo de sensor pode assumir diversos valores ao longo do tempo, desde que esteja dentro da sua faixa de funcionamento. A Figura 5 ilustra um exemplo de sensor analógico, mostrando a tensão de saída, frente a uma grandeza física em uma faixa de tempo determinada.

Figura 5: Saída Analógica de um Sensor.

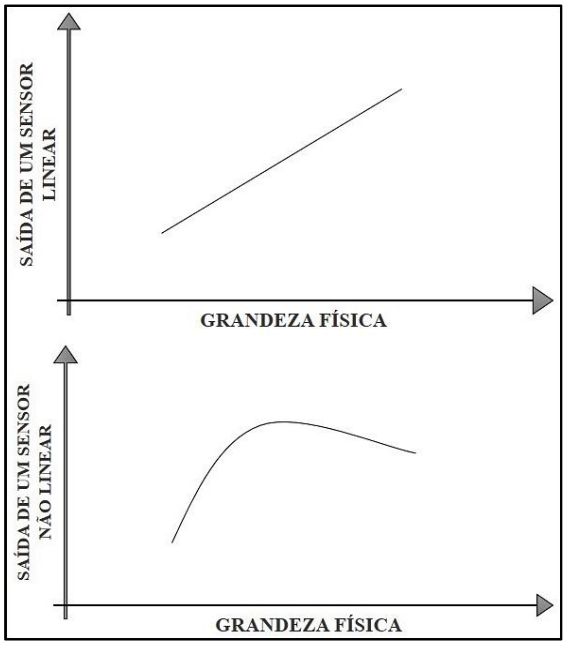


Fonte: Próprio Autor.

b) Linearidade:

Aplicado a sensores analógicos, a linearidade é uma curva de saída, a partir da grandeza medida. O objetivo é buscar respostas proporcionais as entradas, objetivando a facilitar a montagem do circuito de interface. Nem sempre a linearidade é possível, pois existem sensores não lineares. A Figura 6 mostra a diferença entre um sensor linear e um não linear.

Figura 6: Saída de um sensor linear e não linear.



Fonte: Próprio Autor.

c) Alcance:

Representa toda a faixa de valores de entrada do sensor. Dessa forma, quanto maior o alcance mais preciso será a resposta de saída do sensor.

d) Velocidade de Resposta:

É a diferença entre o tempo que o sensor percebe a variação do ambiente com o tempo de resposta. O ideal seria que essa diferença fosse nula, ou seja, o sensor teria uma resposta instantânea.

Portanto, existem vários sensores que são bastantes aplicados nas linhas de produções automatizadas como os sensores ultrassônicos e os infravermelhos. As principais aplicações

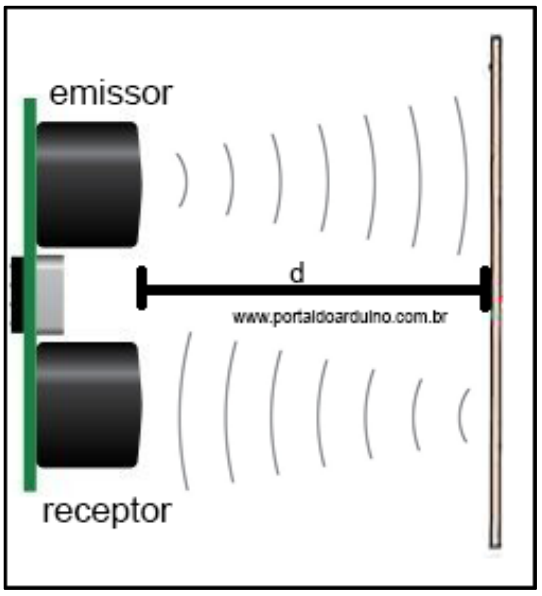
desses sensores são na mediação das dimensões de objetos, controle de posicionamento e verificação de danos ou falhas dos produtos (PATSKO, 2006).

2.3.1 Sensor Ultrassônico

Os sensores ultrassônicos são bastante utilizados na detecção de obstáculos a certa distância, desde que, os obstáculos tenham tamanho consideráveis e sejam capazes de refletir a radiação ultrassônica emitido por estes sensores (WENDLING, 2006).

O princípio de funcionamento se dá a partir de um oscilador que emite ondas eletromagnéticas com uma frequência em torno de 42 kHz. As ondas se propagam pelo meio e quando colidem com algum objeto são refletidas. Essas ondas refletidas são captadas pelo sensor, e com base no tempo de resposta desde sua incidência até o seu retorno, se torna possível obter informações sobre a distância do objeto que refletiu o sinal. A Figura 7 mostra o princípio de funcionamento de um sensor ultrassônico.

Figura 7: Funcionamento de um sensor Ultrassônico.



Fonte: Portal Arduíno, 2016.

Geralmente esses sensores são compostos por um gatilho (emissor) e um eco (receptor), funcionando como um sonar, ou seja, o sensor emiti pulsos ultrassônicos no meio

em que está inserido, e espera pelo retorno do seu eco. A distância do objeto ao senso está diretamente relacionada com o tempo da demora do eco. Quanto mais tempo demora o eco maior a distância, da mesma forma, quanto menor o tempo de demora do eco, menor a distância (MOREIRA 2012).

A velocidade do sinal ultrassônica é aproximadamente $340\ m/s$, dessa forma, se o sensor estiver a uma distância d do objeto, o sinal percorrerá uma distância $2d$ para sair e retornar ao sensor. Assim, pode-se calcular a distância de um objeto pela equação a seguir:

$$V = \frac{distância}{tempo} \tag{2.0}$$

$$V = \frac{2d}{t} \tag{2.1}$$

deixando o d em função do tempo, e sabendo que a velocidade do som é de $340\ m/s$, têm que a distância é de:

$$d = \frac{V\ x\ t}{2} \tag{2.2}$$

$$d = \frac{340\ x\ t}{2}$$

$$d = 170\ x\ t \tag{2.3}$$

São diversas as aplicações dos sensores ultrassônicos na automação como ferramenta de detecção sem contato, possibilitando que objetos de várias formas e cores, sejam percebidos no campo de atuação. Como por exemplo, detecção de nível de altura, medida de separação, medida de diâmetro em bobinas, detecção de objetos transparentes em ambiente com vapor e líquidos que são bastante utilizados para medir níveis de reservatórios.

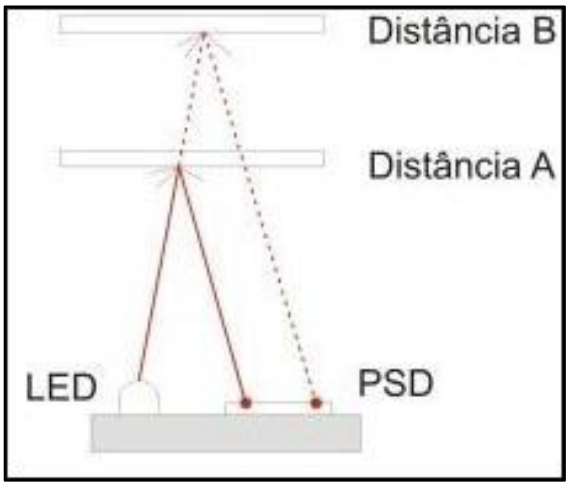
2.3.1 Sensor Infravermelho

Segundo Moreira (2012), o princípio de funcionamento do sensor infravermelho se baseia na triangulação de raios infravermelhos para medir a distância de um determinado objeto a ele. O funcionamento se dar da seguinte forma: um LED emite um feixe de luz infravermelho no ambiente, quando esse feixe de luz é refletido por um objeto o mesmo é

captado por um PSD (*Position Sensing Device* – Dispositivo de Monitoramento de Posição). A distância é calculada de acordo com a incidência do feixe refletido no PSD, que varia com a distância do objeto.

Como é visto na Figura 8, o raio captado pelo PSD varia de acordo com a distância do objeto.

Figura 8: Princípio de funcionamento de um sensor infravermelho.

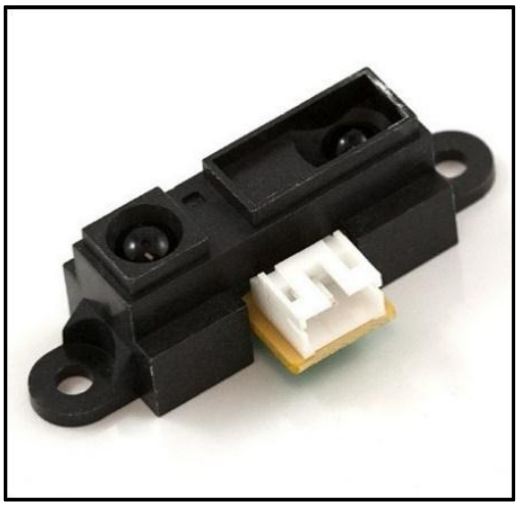


Fonte: Patsko, 2006.

Segundo Patsko (2006), o PSD é composto por diversos componentes sensíveis a luz, ou seja, são fotodiodos. Esse dispositivo é monitorado por um módulo de processamento que identifica o local em que a luz incidiu sobre o PSD. Como essa posição que incidiu no PSD está relacionada com a distância do objeto que refletiu, o módulo processa essa informação, dando resposta um sinal correspondente a essa distância. Portanto, a intensidade do sinal é diretamente proporcional à distância do objeto, ou seja, quanto mais perto estiver o objeto, maior será a intensidade do sinal, da mesma forma, quanto mais longe estiver o objeto, menor será a intensidade do sinal.

A Figura 9 mostra um exemplo de um sensor infravermelho que obedece o princípio de funcionamento descrito anteriormente.

Figura 9: Sensor Infravermelho.



Fonte: PID Electronics, 2016.

2.4 Atuadores

Segundo Wendling (2012), os atuadores são dispositivos que interferem no ambiente, alterando uma variável controlada. Esses dispositivos recebem um sinal gerado pelo controlador, agindo sobre o sistema controlado. Exemplos de alguns atuadores são: válvulas, relés, motores, etc.

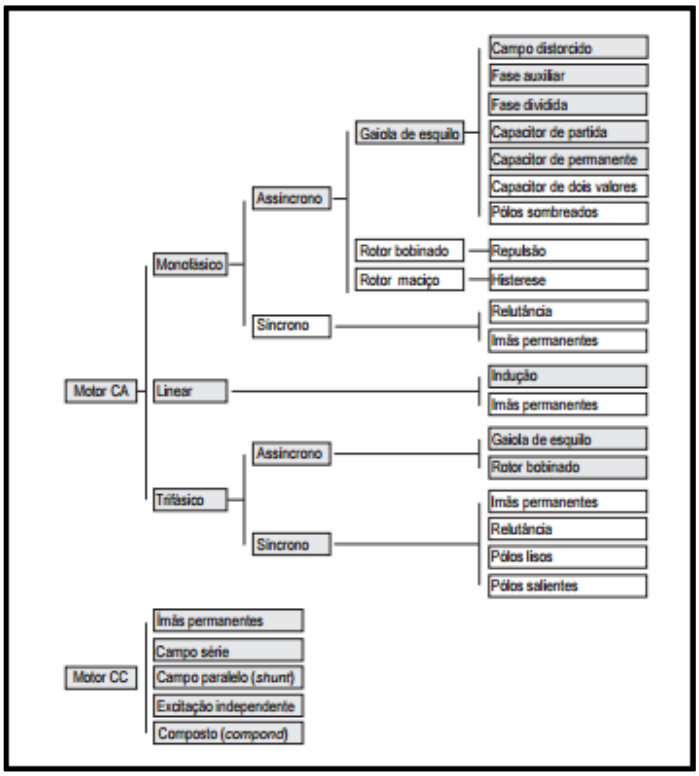
2.4.1 Motor Elétrico

O motor elétrico é um dispositivo que transformam energia elétrica em energia mecânica, sendo o mais usual de todos os motores, o motor elétrico combina as vantagens da utilização da energia elétrica que por sua vez possui um baixo custo, com sua simples construção (PROCEL, 2009).

Os motores elétricos podem ser divididos em duas grandes classes: motor de corrente contínua (CC) e motor de corrente alternada (CA), este podendo ser síncrono ou assíncrono, monofásico ou trifásico.

A Figura 10 apresenta as divisões de cada classe dos motores elétricos.

Figura 10: Divisão dos motores elétricos.

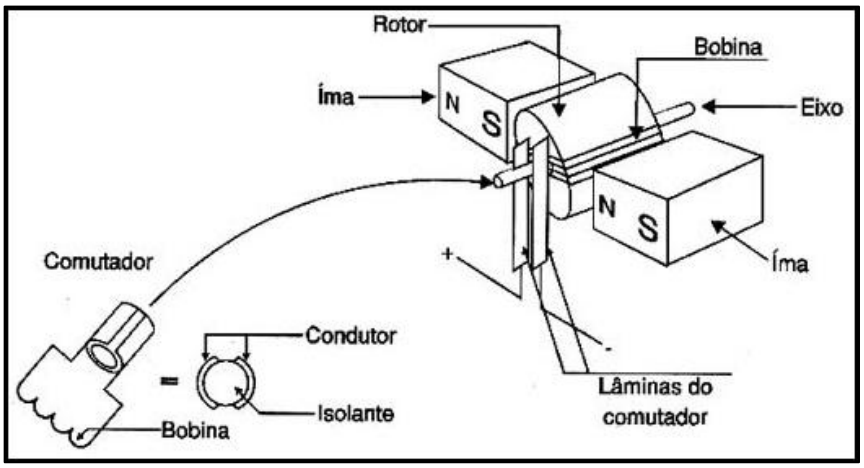


Fonte: Procel, 2009.

2.4.1.1 Motor acionado por corrente contínua (CC)

Segundo Moreira (2012), motores de corrente contínua (CC) são motores que utilizam pilhas, baterias ou uma fonte de alimentação que converte a corrente alternada em corrente contínua. Esses motores estão bastante presentes em brinquedos motorizados, como carrinhos de controle remoto, etc. Quando este dispositivo é energizado, o mesmo gira em um determinado sentido, e se inverter a polarização da fonte elétrica nestes terminais o motor gira no sentido oposto.

Figura 11: Motor de corrente contínua.



Fonte: Silva, 2013.

A Figura 11 mostra como funciona o motor de corrente contínua. Pode-se observa que o condutor gera um campo magnético, quando nele é aplicado uma corrente, que pode ser repelido ou atraído pelos ímãs permanentes. Os comutadores mudam de contato duas vezes por ciclo, invertendo dessa forma, o sentido da corrente na armadura, fazendo com que a armadura gire sempre no mesmo sentido, gerando uma rotação contínua enquanto se tem uma carga aplicada a escova. Esses motores possuem uma alta velocidade de giro, porém um pequeno torque (MOREIRA, 2012).

Existem outros motores elétricos que possuem mais de dois terminais, como por exemplo os motores de passo. Nestes motores a energia é distribuída de maneira mais organizada proporcionando um controle de ciclo interno do motor. Essa característica permite um controle preciso de posição em um determinado tempo, e um maior controle da velocidade de rotação (GIOPPO et al, 2009).

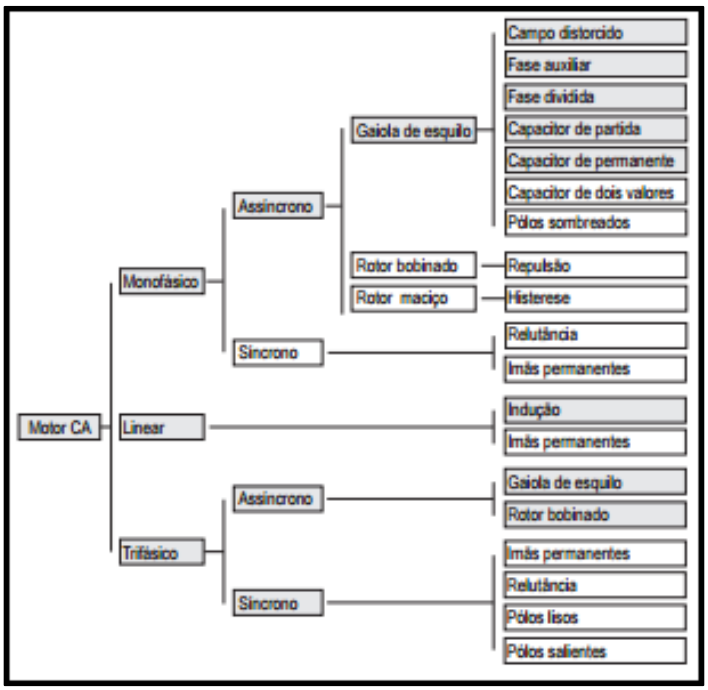
2.4.1.2 Motor acionado por corrente alternada (CA)

Os motores de corrente alternada (CA), são motores que funcionam em corrente alternada. Esses motores são utilizados em máquinas que são alimentados pela rede elétrica. A

velocidade dos motores CA é determinada pela frequência na qual opera a rede elétrica, o que permite boas condições de funcionamento e velocidades constantes (PROCEL, 2009).

Existem dois grandes grupos de motores de corrente alternada, que depende dos critérios usados em sua classificação. Podendo ser assíncronos ou síncronos, quando leva em consideração a rotação do motor, e pode ser monofásico ou trifásico quando leva em consideração a tensão elétrica fornecida pela rede de distribuição. A Figura 12 mostra como está dividia a classe do motores CA.

Figura 12: Classificação dos Motores CA.



Fonte: Procel, 2009.

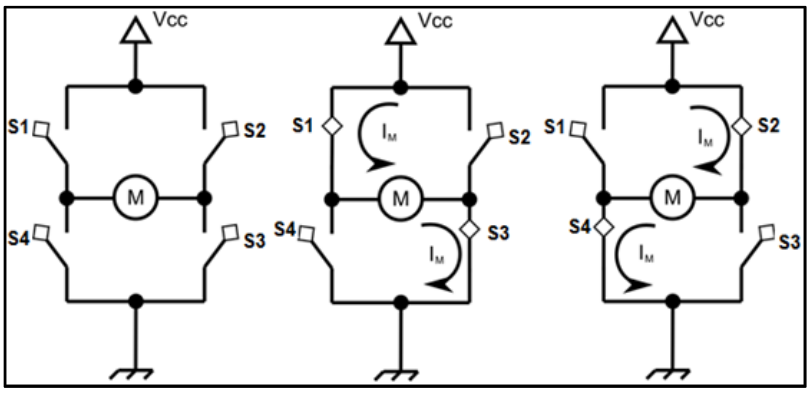
a) Motores Síncronos: o eixo do motor gira com a mesma frequência da rede elétrica, ou seja, com velocidade constante;
b) Motores Assíncronos: são os mais utilizados nas grandes industrias, possuem vantagens como elevada confiabilidade, e desvantagens como baixo rendimento e uma elevada corrente de partida;

2.5 Ponte H

Segundo Gioppo *et al* (2009), existe uma grande necessidade de controlar a rotação dos motores em determinados projetos eletrônico. Dessa forma, é preciso controlar o fluxo de corrente para que o motor inverta o sentido de rotação. Quem determina esse fluxo de corrente é a ponte H.

De acordo com Moreira (2012), a ponte H é um circuito que tem como característica principal controlar um motor de corrente contínua utilizando sinais gerados por um microcontrolador ou por um circuito digital. O controle se dá pela organização de seus componentes tornando fácil controlar a polaridade da tensão no motor, permitindo dessa forma, que o motor gire no sentido horário ou no sentido anti-horário, bem como, controlar a velocidade de giro através do sinal PWM. A Figura 13 mostra o formato de uma ponte H, na qual é composta por quatro chaves mecânicas ou eletrônicas posicionadas formando uma letra H.

Figura 13: Ponte H.



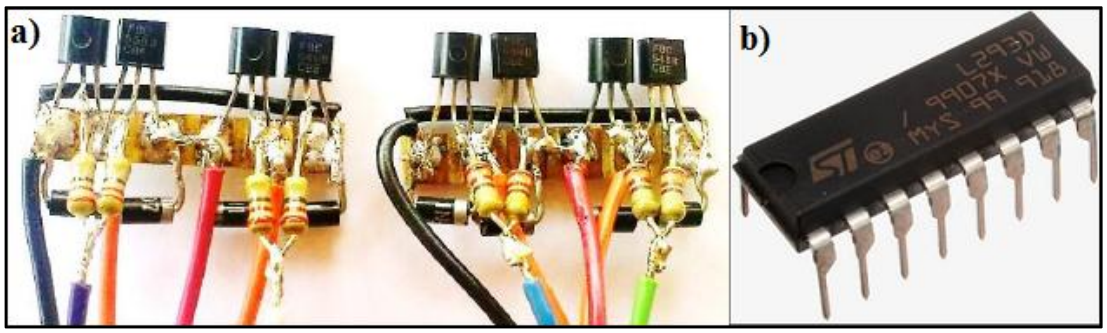
Fonte: http://aecxrobot.blogspot.com.br/p/aula-10-ponte-h.html.

Como é visto na Figura 13, o motor só irá funcionar quando duas chaves forem fechadas no sentido diagonal. Para o motor girar no sentindo contrário, basta abrir as duas chaves que estavam ligadas e ligar as duas chaves que estavam abertas. Caso seja necessário parar o motor, existem duas possibilidades, a primeira desligando todas as chaves, devido não se ter nenhuma corrente no circuito. A segunda possibilidade consiste em ligar duas chaves superiores ao positivo, ou duas chaves inferiores ao negativo da fonte de alimentação, isso

gera um freio eletrônico, devido o fluxo de corrente que passa pela bobina fazer com que o motor pare e não provoque giro. Se ligar as duas chaves em paralelos, ou ligar todas as chaves poderá provocar um curto circuito o que acarretará danos ao dispositivo.

Existem vários tipos de ponte H, podendo encontrar no formato de componentes discretos, ou no formato de circuito integrado. Os dois formatos possuem as mesmas funções, porém o que irá diferenciar é a potência da carga acionada.

Figura 14: Tipos de ponte H: a) construída a partir de materiais discretos; b) formato de circuito integrado.



Fonte: a) Moreira, 2012; b) http://www.arduinoecia.com.br/2014/04/controle-de-motor-cc-com-o-l293d-ponte-h.html.

2.6 *Pulse Width Modulation* (PWM)

Segundo Gioppo *et al* (2009), *Pulse Width Modulation* (PWM) ou modulação por largura de pulso é uma técnica que obtém um sinal analógico a partir de um sinal digital. O PWM é utilizado quando se quer controlar o valor médio de um sinal de tensão ou corrente de uma determinada carga. Neste trabalho, o PWM, foi utilizado para controlar a velocidade de rotação dos motores.

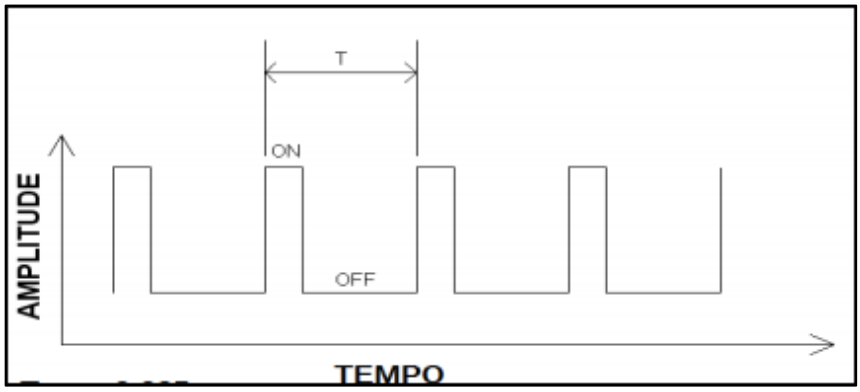
De acordo com Oliveira (2014), PWM é uma forma de codificar digitalmente níveis de sinais analógicos regulando a potência total entregue a uma carga. Essa técnica consiste no equilíbrio dos intervalos de tempo em que a carga recebe a potência máxima e mínima, de tal forma a atingir um valor intermediário desejado.

Ainda segundo Oliveira (2014), pode-se definir o intervalo de tempo em que a carga recebe potência nula t_{off} , e também pode-se definir o intervalo de tempo em que a carga recebe a potência máxima t_{on}. Portanto, o período de operação T é dado por:

$$T = t_{on} + t_{off} \tag{2.2}$$

pode-se observar esses três intervalos de tempo na Figura 15.

Figura 15: Forma de onda de um sinal PWM.



Fonte: Oliveira, 2014.

Dessa forma, se por exemplo, uma carga estiver ligada em dois pontos, em que as tensões valem 5 V e 0 V, a carga irá receber potência nula se chave estiver aberta entre os terminais, se a chave estiver fechada a carga receberá potência máxima. Ou seja, deixando a chave aberta durante todo o período de operação, a carga irá receber potência nula, permanecendo desligada, caso contrário, a carga receberá potência máxima. Se um determinado circuito necessitar de uma potência que esteja entre o valor mínimo e máximo, a solução é fazer com que a chave mude rapidamente, permanecendo um determinado período de operação aberta e um determinado período de operação fechada.

A tensão média de uma forma de onda é dada por:

$$V_{dc} = \frac{1}{T}\int_{0}^{T} V(t)dt \tag{2.4}$$

onde T é o período da forma de onda, como já explicado anteriormente, e $V(t)$ é a função da tensão, que é dado pela amplitude, ao longo do tempo. Portanto,

$$V(t) = \begin{cases} V_{pulso}, & 0 \leq t \leq t_p \\ 0, & t_p < t \leq T \end{cases} \tag{2.5}$$

onde, t_p é a duração do pulso em nível lógico 1, ou seja t_p e V_{pulso} é o tempo e a tensão de pulso do sinal PWM. Portanto,

$$V_{dc} = \frac{1}{T}\left(\int_0^{t_p} V_{pulso}\,dt + \int_{t_p}^{T} 0\;dt\right) \tag{2.6}$$

logo:

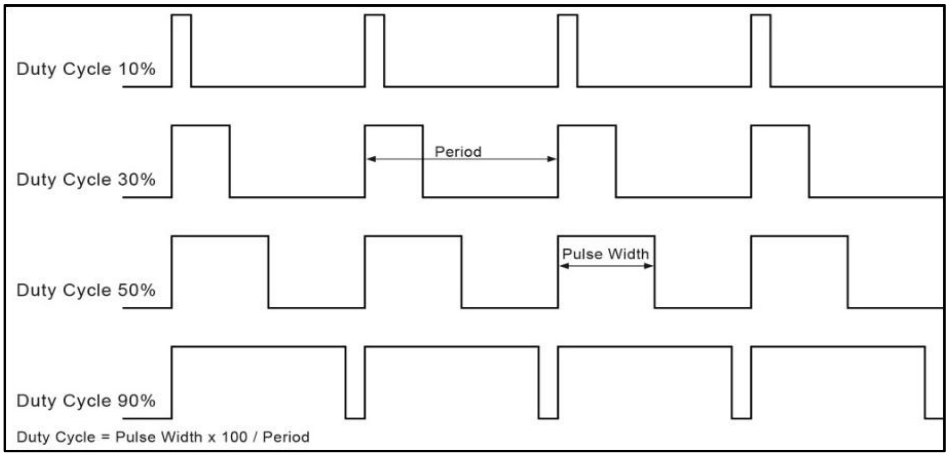
$$V_{dc} = \frac{t_p}{T} V_{pulso} \tag{2.7}$$

A razão entre o tempo em que a carga recebe a potência máxima com período de operação é chamado de ciclo ativo ou *duty cycle.*

$$duty\ cycle = \frac{t_{on}}{T}\ x\ 100\% \tag{2.8}$$

O *duty cycle*, representa a fração da potência total que é entregue a carga durante o período de operação T. Variando a frequência, pode-se obter diferentes valores para o *duty cycle*, como pode ser visto na figura 16.

Figura 16: Formas de ondas com diferentes *Duty Cycle*.



Fonte: http://www.protostack.com/blog/2011/06/atmega168a-pulse-width-modulation-pwm/.

Portanto, esta técnica permite variar a intensidade média da corrente no motor, bem como, alimentando-os com pulsos e controlando a sua duração. Dessa forma, a tensão em cada pulso é a tensão máxima da fonte de alimentação, mas o valor aplicado ao motor será o valor determinado pelo *duty cycle*. Aumentando a duração dos pulsos, o motor receberá alimentação por um tempo mais longo e na média terá uma tensão correspondente maior, o

que faz com que o motor gire mais rápido. Caso contrário, reduzindo a largura do pulso, o motor receberá uma alimentação por um tempo mais curto, reduzindo a tensão e consequentemente diminuindo o giro do motor.

3 PLATAFORMA ARDUINO

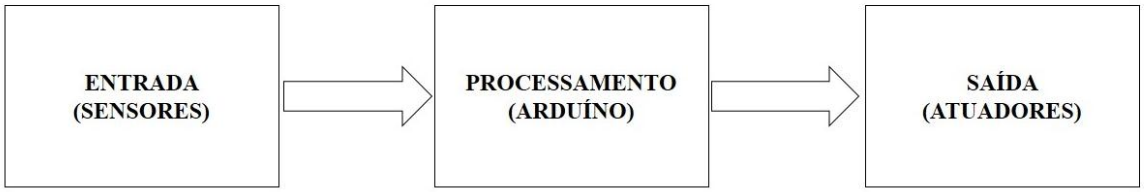
Segundo Fonseca e Beppu (2010), o conceito de Arduino surgiu na Itália em 2005, com a necessidade de criar um dispositivo que possa controlar projetos e/ou protótipos construído de maneira mais interativa e de baixo custo, do que outros sistemas já disponíveis no mercado.

A plataforma Arduino é um sistema embarcado, união entre *hardware* e *software*, baseado em uma plataforma de computação física, que são sistemas digitais interligados a sensores e atuadores permitindo construir sistemas intuitivos que percebam a realidade e respondam a ações, baseado em uma placa que possui entradas e saídas microcontroladas, desenvolvido a partir de uma biblioteca baseada na linguagem de programação C/C++ (FONSECA; BEPPU, 2010).

Ainda segundo Fonseca e Beppu (2010), a plataforma Arduino pode ser usado para desenvolver diversos projetos, pois é uma placa capaz de interpretar variáveis no ambiente e transformá-las em sinais elétricos correspondentes. Esses sinais são interpretados pelo microcontralador através dos sinais gerados pelos sensores aos seus terminais de entrada, e que por sua vez, atua no controle de outro dispositivo ligado no terminal de saída.

A Figura 17 mostra os principais elementos do sistema embarcado.

Figura 17: Diagrama de Blocos da plataforma Arduino.



Fonte: Próprio Autor.

A plataforma Arduino é baseado em um microcontrolador (Atmega), logicamente programável, ou seja, se torna possível criar programas utilizando a sua própria linguagem baseada em C/C++. Dessa forma, a plataforma Arduino é uma ferramenta *open-source*, isto é, uma ferramenta para pessoas que gostam de programar e criar projetos nas diversas áreas da engenharia (FONSECA; BEPPU, 2010).

3.1 Características da plataforma Arduino UNO

O plataforma Arduino vem se desenvolvendo desde seus primórdios. Já foram criadas várias versões da sua plataforma, como Arduino Nano, Arduino Mega, Arduino UNO, e entre outras versões (ARDUINO, 2016). Na Tabela 1, pode-se observar as principais características técnicas da plataforma Arduino Uno.

Tabela 1: Características técnicas do Arduino UNO.

Microcontralador	ATmega328P
Tensão de operação	5 V
Tensão de entrada (recomendado)	7-12 V
Tesão de entrada (limite)	6-20 V
Pinos E/S digitais	14 (6 oferecem saídas PWM)
Pinos de entrada analógica	6
Corrente DC por pino de E/S	20mA
Corrente DC para pino 3.3V	50mA
Memória Flash	32 KB (ATmega328P), dos quais 0,5 KB são utilizados pelo gerenciador de boot.
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)

Velocidade de Relógio	16 MHz
Comprimento	68,6 mm
Largura	53,4 mm
Peso	25 g

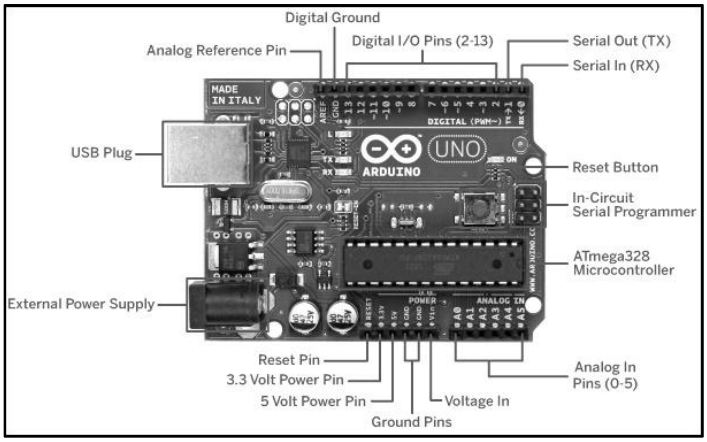
Fonte: Arduíno, 2016.

Como é visto na tabela 1, a plataforma Arduino UNO pode ser alimentado tanto de um fonte externa, como baterias por exemplo, quanto de um fonte provinda pela conexão USB, sendo a fonte de energia selecionado automaticamente (MOREIRA, 2012).

A plataforma Arduino UNO possui 14 pinos digitais, nos quais 6 oferecem saídas PWM. Esses pinos operam com 5 volts e podem ser utilizados tanto como entrada ou saída, que varia de acordo com a programação feito no microcontrolador (ARDUINO, 2016).

Segundo Arduino (2016), além dos 14 pinos digitais, a plataforma possui ainda 6 entradas analógicas, com 10 bits de resolução. O conversor A/D da plataforma Arduino Uno são de 10 bits, ou seja, ele pode ler informações entre 0 (0 V) a 1023 (5 V). A Figura 18 mostra detalhadamente como está dividida a plataforma Arduino UNO.

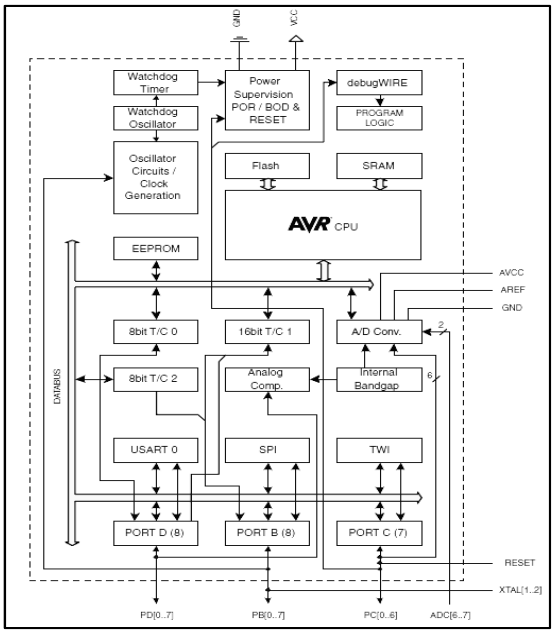
Figura 18: Componentes da plataforma Arduino UNO.



Fonte: Arduino, 2016.

A Figura 19 mostra o diagrama de blocos da arquitetura do microcontrolador ATmega328 e seus respectivos periféricos.

Figura 19: Diagrama de Blocos do microcontrolador ATmega328.

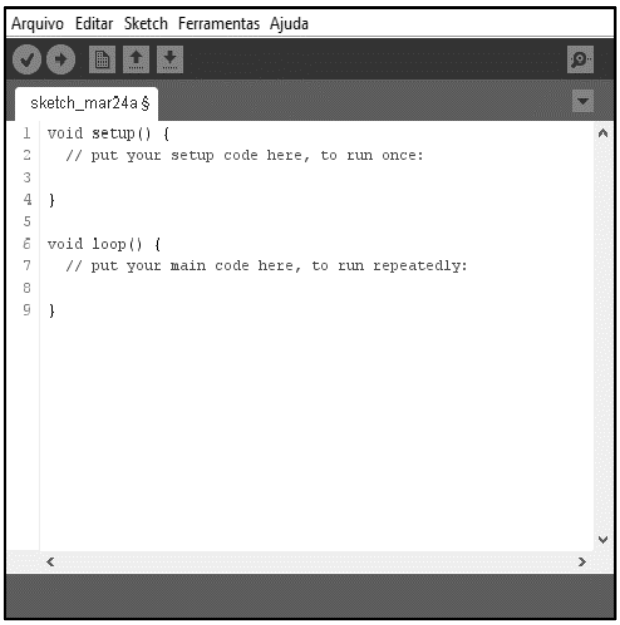


Fonte: Arduino, 2016.

3.2 IDE da plataforma Arduino

O Ambiente de Desenvolvimento Integrado IDE (*Integrated Development Enviroment*), pode ser baixado gratuitamente pela WEB. É nesse ambiente que será realizada toda a programação feita pelo usuário. A Figura 20 mostra a aparência da sua interface.

Figura 20: IDE da plataforma Arduino.

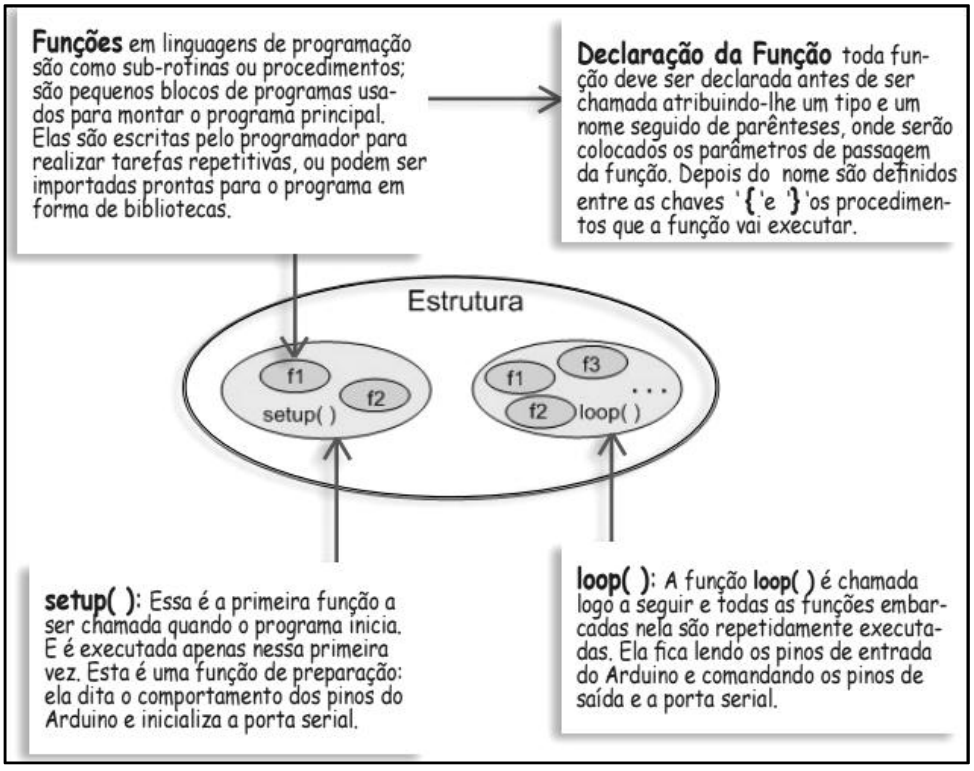


Fonte: Arduino, 2016.

3.2.1 Linguagem de programação da plataforma Arduino

Segundo Silveira (2012), a estrutura de programação do Arduino é formada por dois blocos de funções que carregam outros blocos de funções escritas na linguagem C/C++. A Figura 21 mostra a estrutura da programação da plataforma Arduino.

Figura 21:Estrutura da programação da plataforma Arduino.



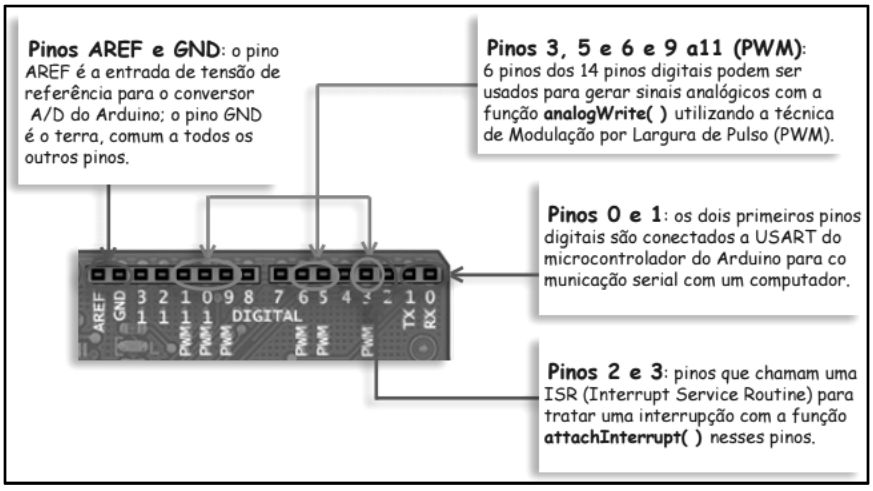
Fonte: Silveira, 2012.

3.2.1.1 Principais funções dos pinos digitais da plataforma Arduino UNO

- pinMode(): função de direcionar o fluxo de informação em qualquer pino digital. Nessa função, deve ser informado qual pino irá usar, e qual a sua função, se é entrada ou saída das informações PinMode (Pino, Modo). Está função é sempre escrita na função setup;
- digitalRead(): essa função lê as informações presentes na função PinMode(), e pode ser armazenado em qualquer outra variável descrita pelo usuário;
- digitalWrite(): essa função envia um nível lógico para qualquer pino digital do Arduino. Nessa função, deve-se informar dois parâmetros, o número do pino digital e o estado lógico que o pino deve permanecer, alto ou baixo (HIGH ou LOW).

A Figura 22 mostra detalhadamente onda estão localizados os pinos digitais da plataforma Arduino UNO, bem como, o que representa cada um deles.

Figura 22: Pinos Digitais da plataforma Arduino UNO.



Fonte: Silveira, 2012.

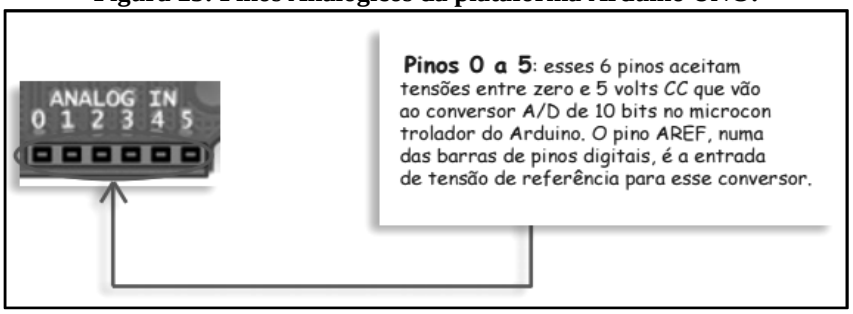
3.2.1.2 Principais funções dos pinos analógicos da plataforma Arduino UNO

- analogRead: da mesma forma do digitalRead(), essa função lê as informações enviadas pelo pino analógico, e pode ser armazenada em qualquer outra variável descrita pelo usuário.

- analogWrite: função de gerar tensões analógicas em 6 dos pinos digitais, utilizando PWM. Dois parâmetros devem ser informados o número do pino em que a tensão deve ser gerada, e a amplitude da tensão que pode variar de 0 (0 V) e 255 (5 V).

A Figura 23 mostra onde estão localizados os pinos analógicos da plataforma Arduino UNO, e suas principais características.

Figura 23: Pinos Analógicos da plataforma Arduino UNO.



Fonte: Silveira, 2012.

4 DESCRIÇÃO DO PROTÓTIPO E MATERIAIS UTILIZADOS

Neste capítulo será abordado a descrição do protótipo de uma cadeira de rodas inteligente, assim como, os materiais utilizados e o fluxograma de controle.

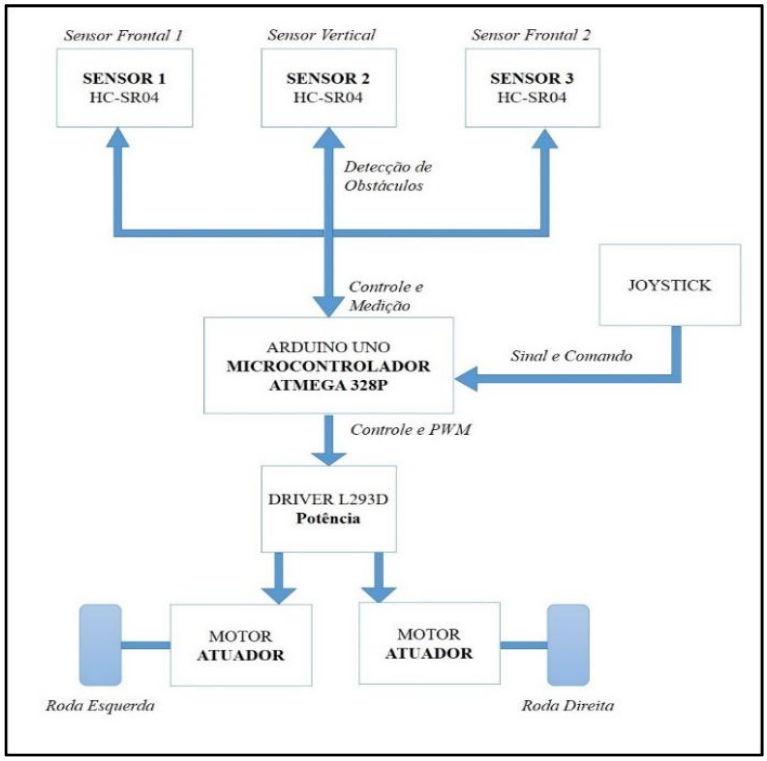
4.1 Diagrama geral do protótipo

Este tópico abordará os diagramas em bloco e elétricos do protótipo proposto neste trabalho.

4.1.1 Diagrama em Blocos

A Figura 24 apresenta o diagrama em blocos do protótipo, que é um conjunto de sensores, atuadores, circuitos de potência e microcontrolador.

Figura 24: Diagrama de blocos do protótipo.



Fonte: Próprio Autor.

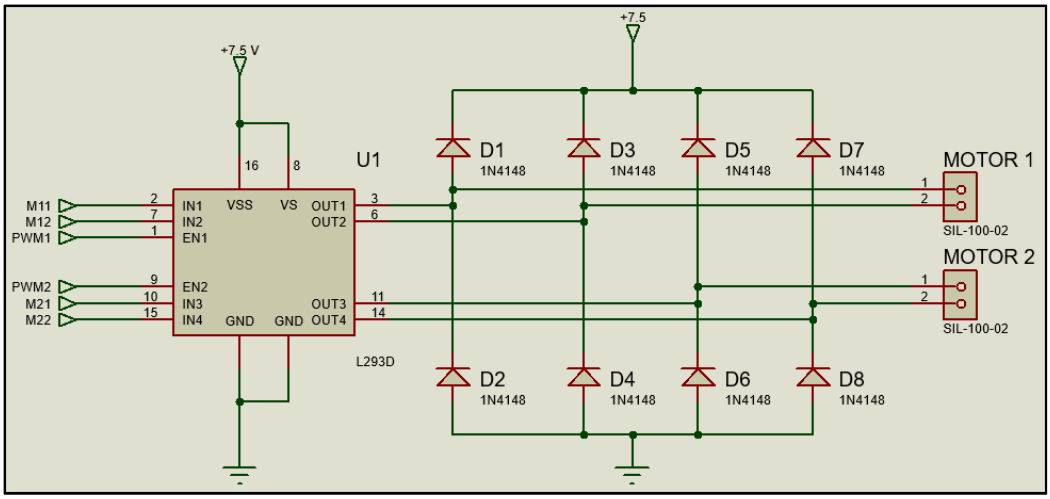
Como pode ser visto na Figura 24, o protótipo possui um *joystick*, que manda informação para o microcontrolador para movimentar o protótipo através do circuito de potência e da ponte H. O mesmo ainda possui um conjunto de três sensores ultrassônicos que tem a função de detectar obstáculos para que o algoritmo tome um determinada decisão.

Cada sensor ultrassônico está ligado a um pino "I/O" do microcontrolador da plataforma Arduino UNO, sendo que todos estes, estão ligados nas portas digitais oferecidas por essa placa. Portanto, o microcontrolador interpreta os sinais recebidos por esses sensores, por meio do estado dos pinos digitais, e envia informação para os atuadores.

4.1.2 Diagrama Elétrico

A figura 25 mostra o diagrama eletrônico do circuito de potência das rodas de tração, que é composto por uma ponte H, L293D, e dois atuadores (servo motor). As saídas para os motores possui diodos D1, D2, D3, D4, D5, D6, D7 e D8, que são usados para proteger os pinos do C.I devido a corrente reversa gerada pela indutância dos motores.

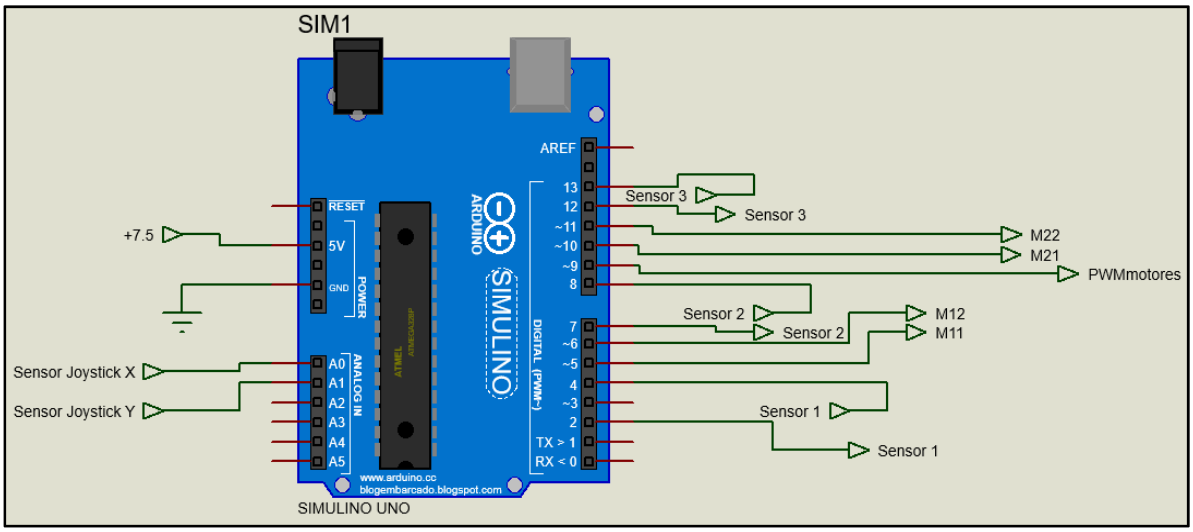
Figura 25: Diagrama elétrico do circuito de potência das rodas de tração do protótipo.



Fonte: Próprio Autor.

Na Figura 26, tem-se o diagrama elétrico do microcontrolador (ATmega328P) utilizado na plataforma do Arduino UNO. A plataforma realiza a leitura dos sensores ultrassônicos e do *joystick* e gera sinais de PWM para o acionamento de motores.

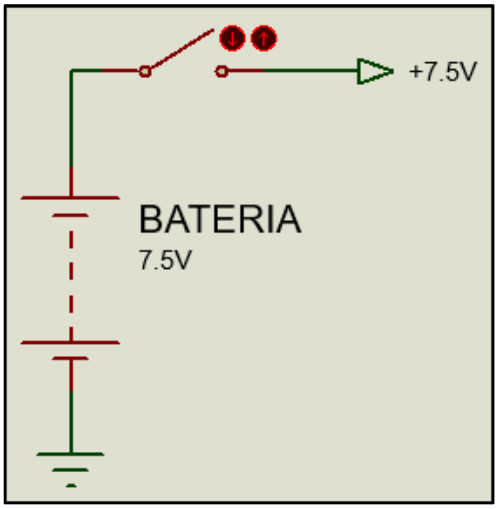
Figura 26: Diagrama elétrico da plataforma Arduino UNO utilizado no protótipo.



Fonte: Próprio Autor.

Na Figura 27, tem-se o circuito de alimentação do protótipo, que nada mais é do que cinco pilhas conectadas em série, fornecendo um total de 7,5 V para todo o sistema.

Figura 27: Diagrama elétrico do circuito de alimentação do protótipo.



Fonte: Próprio Autor.

4.2 Componentes do protótipo

A Tabela 2 mostra todos os componentes que foram utilizados para a construção do protótipo, bem como, o custo total da sua montagem.

Tabela 2: Componentes do Protótipo.

Descrição	Nomenclatura	Referência	Quantidade	Custo (R$)
Arduino Uno	-	R3	1	50,00
Ponte H (CI)	L293D	-	1	07,00
Diodos	D1, D2, D3, D4, D5, D6, D7, D8	1N4148	8	4,00
Servo Motor	Lnb		2	64,00
Sensor Ultrassônico	HC-SR04	-	3	32,40
Joystick	-	-	1	10,00
Suporte para 5 pilhas AA	-	-	1	07,90
Pilhas	AA	-	5	10,00
Jumpers	Macho-Macho/ Fêmea-Fêmea	-	50	12,50
Total				**197,80**

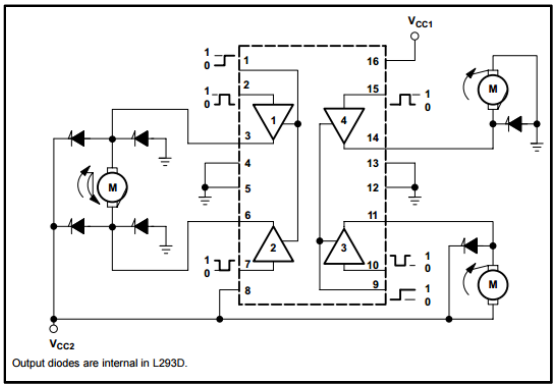
Fonte: Próprio Autor.

4.2.1 *Driver* L293D

O driver L293D, nada mais é do que uma ponte H dupla em formato de CI. Dessa forma, esse circuito pode controlar até dois motores. O controle dos motores é feito através do controle do fluxo de corrente. As suas principais vantagens são no pequeno espaço que ocupa na placa de circuitos e sua simplicidade na montagem. A Figura 14 (b) do capítulo 2 ilustra o formato do C.I.

A Figura 28 apresenta o diagrama de blocos do C.I L293D.

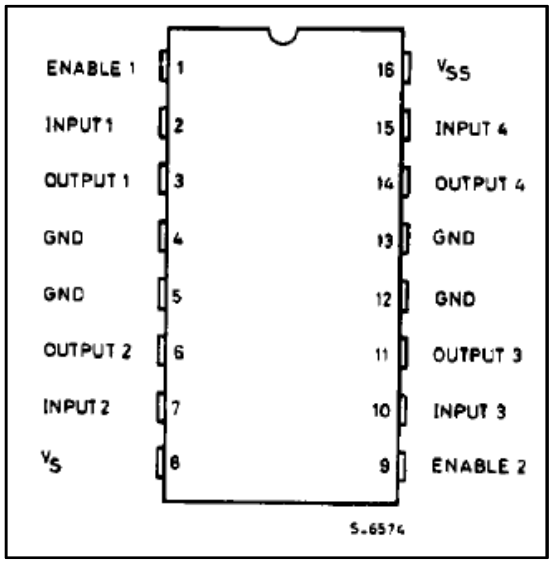
Figura 28: Diagrama de Blocos do C.I L293D.



Fonte: http://www.ti.com/lit/ds/symlink/l293.pdf.

A Figura 29 ilustra os pinos do L293D e como eles estão nomeados.

Figura 29: Pinos do C.I L293D.



Fonte: http://www.ti.com/lit/ds/symlink/l293.pdf.

Os pinos *enables* (1 e 9) são entradas (compatível com o nível TTL). Nível baixo desabilita a ponte A e B e nível alto habilita a ponte A e B. *Outputs* (1 e 2) e *outputs* (3 e 4) são as saídas da ponte A e B, dessa forma, a corrente que flui através desses motores conectados nesses pinos, é monitorado pelo pino 1 e 9 respectivamente. Os pinos 2, 7, 10 e 15 são os *inputs*, entradas das pontes A e B, compatível ao nível TTL.

O C.I L293D foi usado para controlar a rotação dos motores, que manipula a entrada da plataforma Arduino UNO como forma binária. O motor 1 (M1.1 e M1.2) está conectado nos pinos 3 e 6 e o motor 2 (M2.1 e M2.2) está conectado nos pinos 11 e 14 do C.I L293D.

A tabela verdade para o acionamento dos motores pode ser observada na Tabela 3.

Tabela 3: Condições de funcionamento dos motores.

Condições de Funcionamento	Ponte A		Ponte B	
	M1.1	M1.2	M2.1	M2.2
Frente	1	0	1	0
Ré	0	1	0	1
Direita	1	0	0	0
Esquerda	0	0	1	0
Parar	0	0	0	0

Fonte: Próprio Autor.

4.2.2 Módulo *Joystick*

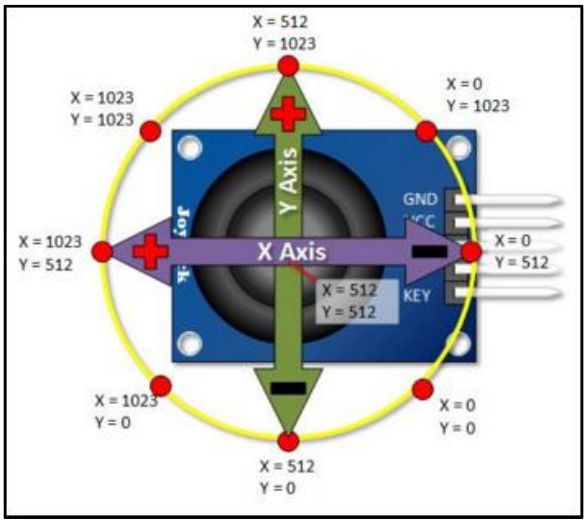
O *joystick* é um módulo de controle em duas dimensões compatível com a plataforma Arduino. Esse módulo permite ler dados nos eixos X e Y através da manipulação de uma alavanca que está fisicamente conectada. A Figura 30 ilustra o módulo *joystick* que foi utilizado nesse trabalho (FAGUNDES *et al*, 2016).

Figura 30: Módulo Joystick.

Fonte: https://www.fasttech.com/product/1320202-5-pin-2-way-arduino-compatible-ps2-joystick-game.

O *joystick* pode ser definido também como sendo um sensor analógico, dessa forma, o mesmo possui uma sensibilidade que varia de acordo com sua inclinação. Portanto, ao inclinar a alavanca em um determinado sentido, o *joystick* pode fornecer valores que variam entre 0 (0V) a 1023 (5V) nos eixos X e Y. A Figura 31 ilustra os valores de X e Y em cada posição do *joystick* (CASADO, 2016).

Figura 31: Valores de X e Y no Joystick.



Fonte: http://www.aprendetecnologia.es/wp-content/uploads/2016/02/f695088bbce6c7b79d3b1108c6b78dfe1.jpg.

A Tabela 4 mostra como conectar o módulo *joystick* na plataforma Arduino UNO.

Tabela 4: Módulo *Joystick* e plataforma Arduino Uno.

Joystick	Plataforma Arduino Uno
GND	GND
+5V	5V
VRx	Porta Analógica
VRy	Porta Analógica
SW	Porta Digital

Fonte: http://www.aprendetecnologia.es/wp-content/uploads/2016/02/joy.png.

4.2.3 Sensor HC-SR04

O sensor HC-SR04 é bastante utilizado na detecção de objetos. Esse sensor utiliza sinais ultrassônicos (40 kHz, acima da capacidade de audição do ouvido humano), para determinar a distância entre o objeto ao sensor. O HC-SR04 pode medir distâncias entre 2 cm a 4 m, com precisão de 3 mm. O ângulo de detecção é de 15 graus (DATASHEET HC-SR04, 2016).

A Tabela 5 mostra as principais características do sensor.

Tabela 5: Características do sensor HC-SR04.

Tensão de Trabalho	DC 5-7 V
Corrente de Trabalho	15mA
Frequência de Trabalho	40kHz
Máximo Alcance	4 m
Mínimo Alcance	2 cm
Ângulo de detecção	15 *degree*
Sinal de entrada de disparo	10us
Sinal de saída de disparo	Sinal de alavanca TTL de entrada e o intervalo em proporção.
Dimensão	45*20*15mm

Fonte: http://www.micropik.com/PDF/HCSR04.pdf.

A Figura 32 mostra o sensor HC-SR04 e seus pinos.

Figura 32: Sensor HC-SR04.

Fonte: http://buildbot.com.br/blog/wp-content/uploads/2015/01/Modulo-HC-SR04-Pinagem.jpg

O módulo possui quatro pinos, sendo o Vcc a sua alimentação em 5V, o *Trigger* emissor da onda ultrassônica, o *Echo* receptor da onda refletida pelo objeto, e o GND o potencial 0V.

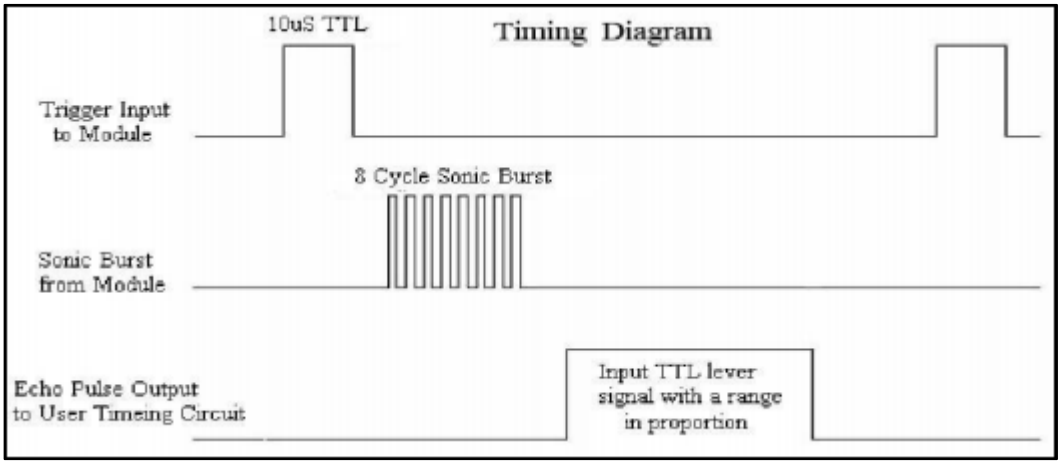
O sensor funciona da seguinte forma: primeiro é enviado um sinal com duração de 10 microssegundos ao pino *trigger* para indicar que a medição se iniciará. Em seguida, o módulo envia 8 pulsos de 40 kHz e o pino *echo* aguarda o retorno do sinal pelo receptor. Se houver o retorno do sinal em nível alto, a distância pode ser calculada usando a equação:

$$distancia = \frac{(Pulso\ em\ nível\ alto\ x\ velocidade\ do\ som\ )}{2} \quad (4.1)$$

é preciso dividir por 2 pois é contado o tempo de ida e volta do sinal.

A Figura 33 ilustra o diagrama de tempo do sensor HC-SR04.

Figura 33: Diagrama de tempo do sensor HC-SR04.



Fonte: http://www.tato.ind.br/produto/Sonar-HC%252dSR04.html.

4.2.4 Servo Motor

Os motores de corrente contínua possuem uma alta velocidade de giro e um pequeno torque, como foi visto no capítulo 2, o que se torna inviável utilizar esse dispositivo no protótipo, visto que o mesmo, deverá se locomover com uma baixa velocidade, simulando dessa forma uma cadeira de rodas motorizada.

Dessa forma, utilizou-se servo motores, por possuir uma caixa de redução de giro e um alto torque. Os servos motores utilizado nesse trabalho são os mesmo usados para redirecionar a polarização e seletores de canais nas antenas parabólicas. A Figura 34 ilustra esse dispositivo.

Figura 34: Servo Motor.

Fonte: http://bistel.com.br/12623-servo-motor-p-antena-parabolica.html.

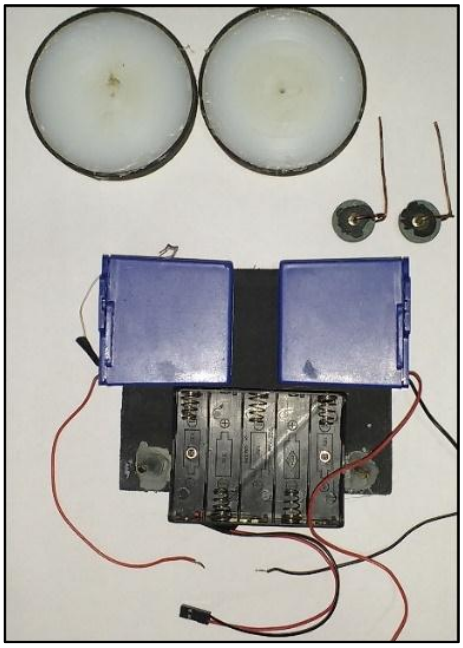
Esses dispositivos possuem uma limitação em seu torque, girando apenas 180^{o}. Dessa forma, se faz necessário algumas modificações no interior da sua estrutura para que o mesmo gire continuamente 360^{o}. Portanto, após essas modificações, como a retirada de uma trava que possui nas engrenagens, o sevo motor, passa a ser um motor de corrente contínua com uma redução de giro acoplado, o que era necessário para o protótipo.

4.3 Estrutura física do protótipo

4.3.1 Base do protótipo

O chassi do protótipo foi montado utilizando um perfil de madeira (MDF) onde foi encaixado os servos motores, o suporte de pilhas e as rodas dianteiras e traseiras. As rodas traseiras foram revestidas com um material emborrachado para aumentar a força de tração. A Figura 35 ilustra como foi montado a base do protótipo.

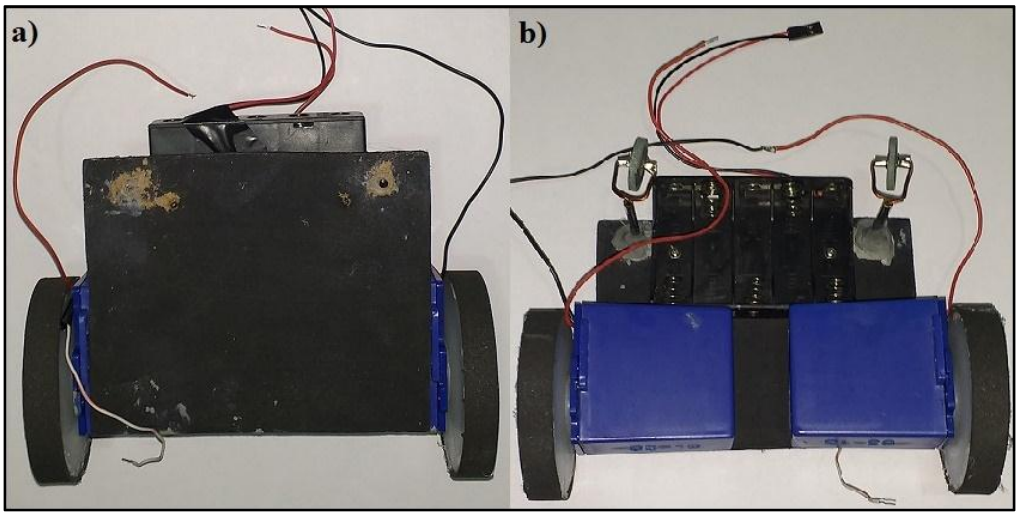
Figura 35: Base do protótipo.



Fonte: Próprio Autor.

A Figura 36 mostra a vista superior e inferior da base do protótipo.

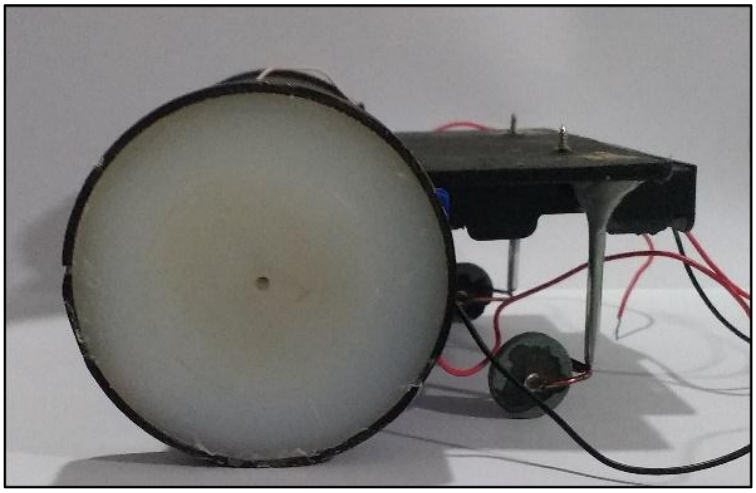
Figura 36: Estrutura física do protótipo: a) vista superior; b) vista inferior.



Fonte: Próprio Autor.

A Figura 37 apresenta a vista lateral do protótipo.

Figura 37: Vista lateral da base protótipo.



Fonte: Próprio Autor.

A cadeira foi modelada com fio de cobre e estofada com um material do tipo E.V.A. A Figura 38 mostra o formato da cadeira.

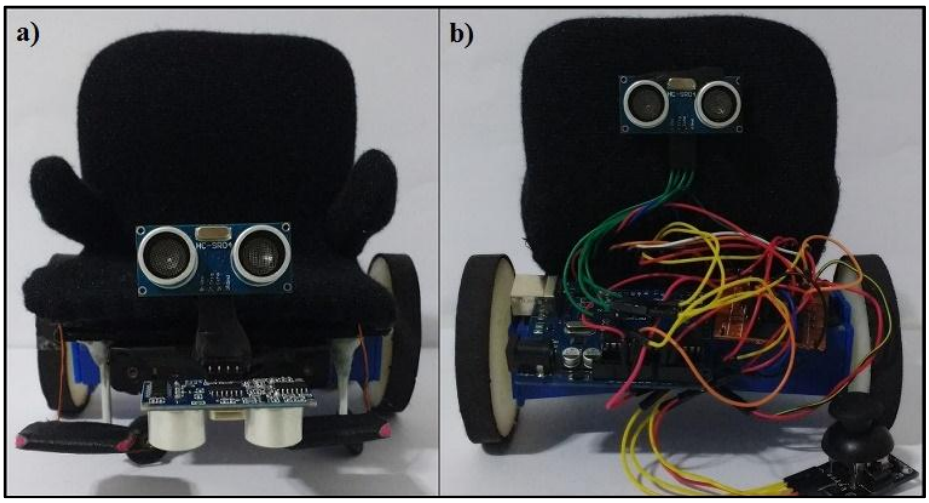
Figura 38: Cadeira do protótipo.

Fonte: Próprio Autor.

4.3.2 Protótipo de uma cadeira de rodas inteligente

Na Figura 39 tem-se a estrutura física do protótipo de uma cadeira de rodas inteligente desenvolvido neste trabalho.

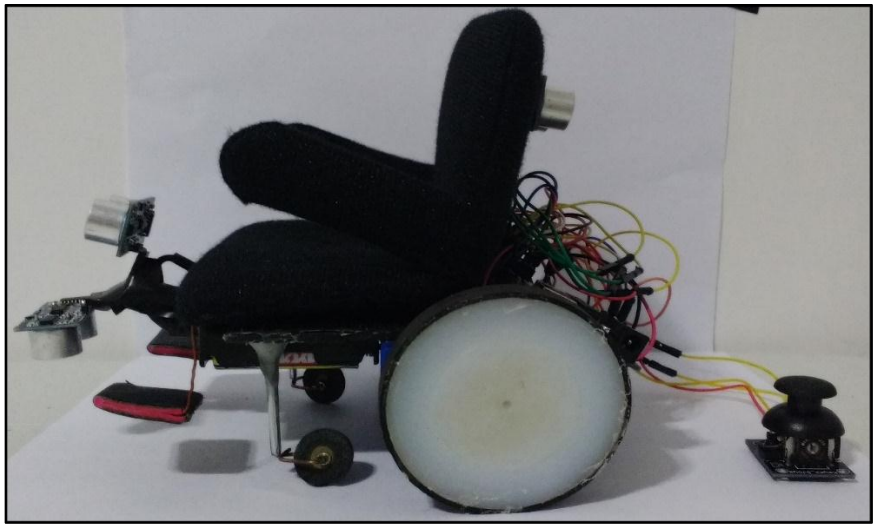
Figura 39: Estrutura física do protótipo: a) visão frontal; b) visão posterior.



Fonte: Próprio Autor.

A Figura 40 mostra a visão lateral do protótipo de uma cadeira de rodas inteligente.

Figura 40: Estrutura física do protótipo: visão lateral.



Fonte: Próprio Autor.

4.4 Fluxograma de controle do protótipo

Para facilitar a compreensão do funcionamento de controle do protótipo, a Figura 40 ilustra o seu fluxograma.

Dessa forma, o programa inicia com a declaração das variáveis e configuração dos pinos I/O, definido como entradas ou saídas e o seu estado inicial. Mais adiante, habilita os PWM de acionamento de cada motor e em seguida o *joystick* inicia sua leitura. Como foi visto anteriormente, existe diferentes valores para cada inclinação realizada no *joystick*. Dessa forma, se fez necessário mapear cada posição do *joystick* (x e y) e após o mapeamento, esses valores foram inseridos no programa, de tal forma, que dependendo da inclinação do *joystick*, o protótipo responderia com alguma movimentação (parado, frente, ré, direita ou esquerda). Após a leitura do sensor analógico, o protótipo recebia leituras dos sensores ultrassônicos, que tinha como objetivo, alertar o mesmo de possíveis obstáculos que estivesse no caminho e consequentemente impedindo uma possível colisão.

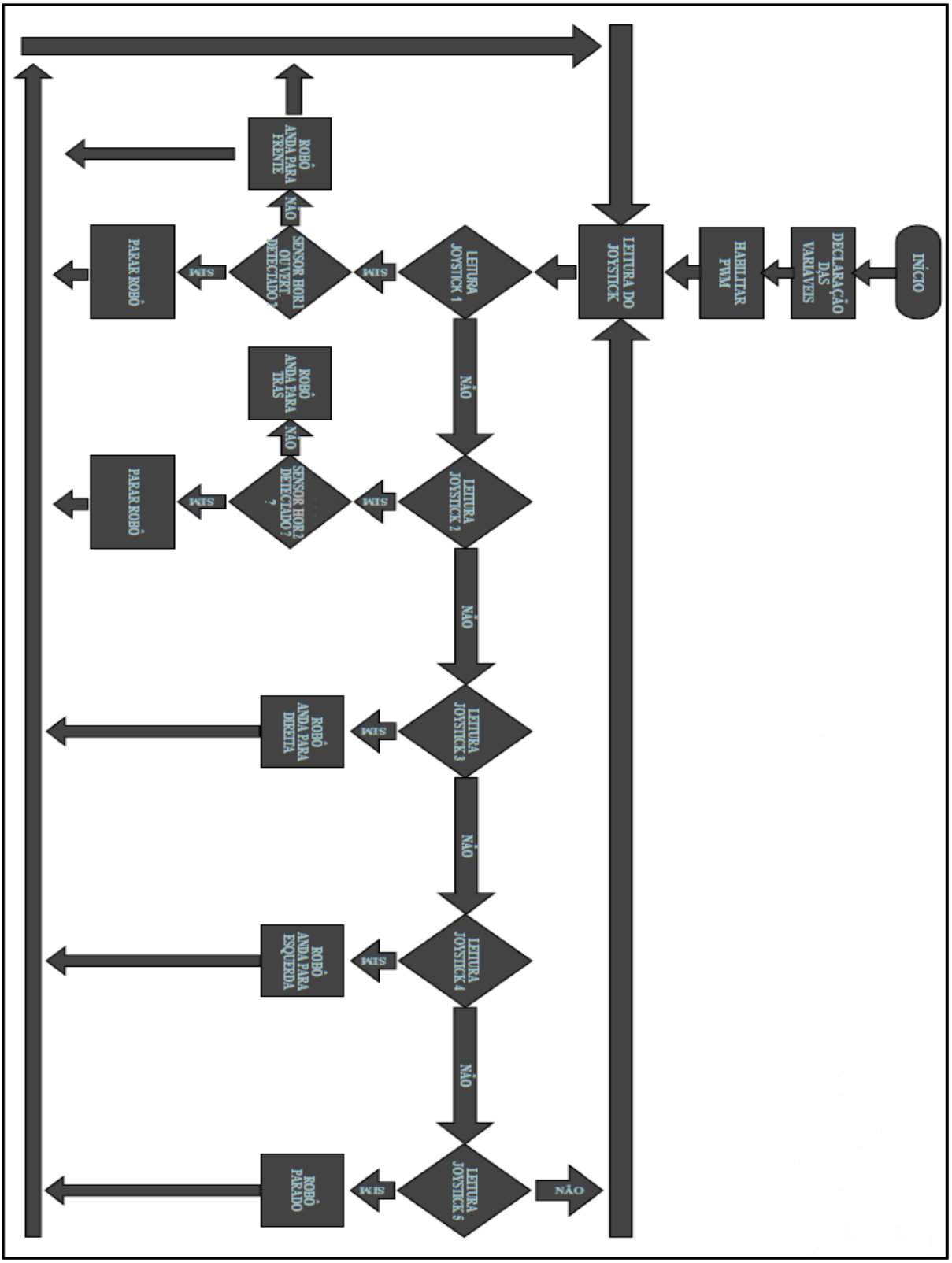
A Tabela 6 apresenta as condições para movimentar o protótipo, a partir dos valores fornecido pelo *joystick*.

Tabela 6: Condições para cada leitura do *Joystick*.

Leitura Joystick 1	(leituray >= 205) && (leituray <= 255)) && ((leiturax >= 100) && (leiturax <= 150)
Leitura Joystick 2	(leituray >= 0) && (leituray <= 20)) && ((leiturax >= 100) && (leiturax <= 150)
Leitura Joystick 3	(leituray >= 100) && (leituray <= 150)) && ((leiturax >= 0) && (leiturax <= 20)
Leitura Joystick 4	(leituray >= 100) && (leituray <= 150)) && ((leiturax >= 205) && (leiturax <= 255)
Leitura Joystick 5	(leituray >= 100) && (leituray <= 150)) && ((leiturax >= 100) && (leiturax <= 150)

Fonte: Próprio Autor.

Figura 41: Fluxograma do protótipo.



Fonte: Próprio Autor.

5 RESULTADOS

No sentido de verificar o funcionamento do *firmware* e do protótipo desenvolvido, realizou-se testes para analisar o comportamento do algoritmo sobre a parte física criada nesse estudo.

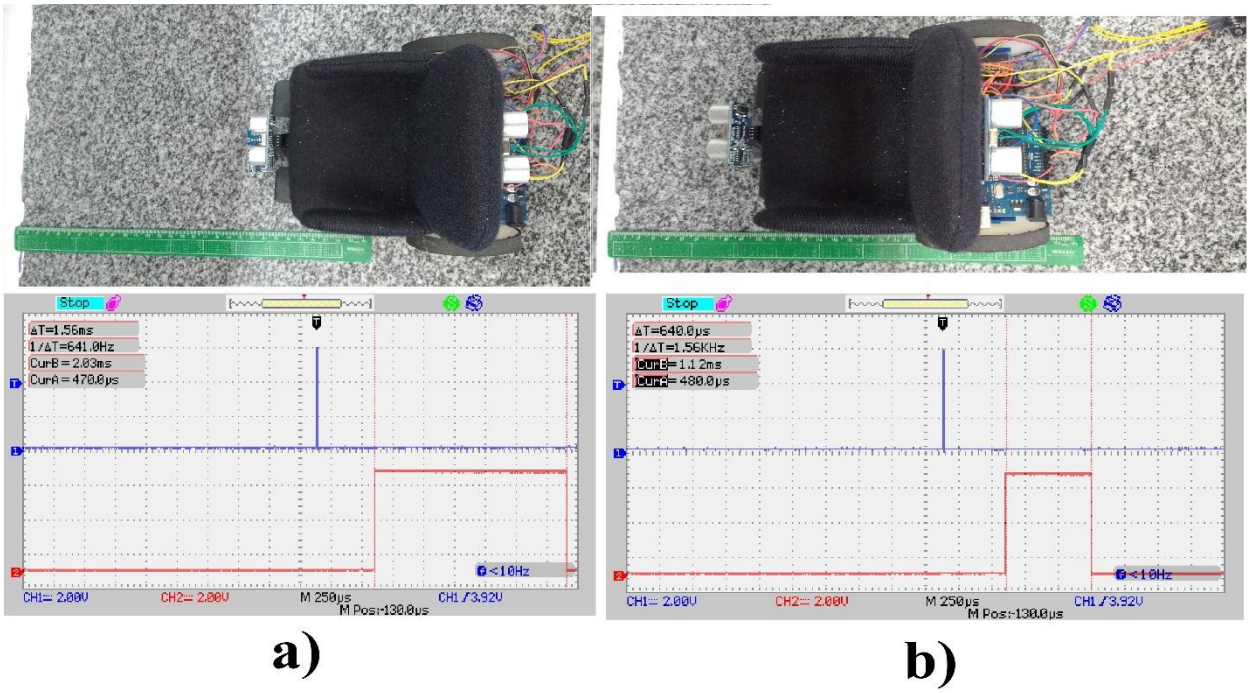
5.1 Teste com o Protótipo

As principais funções do *firmware* desenvolvido foram testadas, desde as mais simples como mover a cadeira para frente, trás, direita ou esquerda, até as mais complexas, como na identificação de objetos, bem como, desníveis no ambiente que possam comprometer a rota do protótipo gerando possíveis colisões.

No primeiro teste, analisou-se o desempenho do protótipo ao se deslocar em vários sentidos, para frente e para trás, ou até mesmo curvas para direta ou para esquerda em um ambiente sem obstáculos ou desníveis. O teste obteve resultados eficientes, pois o controle realizou as operações propostas.

O segundo teste consistiu na capacidade do protótipo em detectar obstáculos no ambiente onde estava se locomovendo. Desta forma, posicionou-se horizontalmente um sensor detector de objetos (HC-SR04) na parte frontal do protótipo. Neste teste, foi colocado um obstáculo que simulava uma parede, uma pessoa ou qualquer outro objeto, por exemplo, que impedisse o seu movimento gerando uma colisão frontal. A Figura 42 ilustra este teste.

Figura 42: Detecção de obstáculos frontal: a) obstáculo a uma distância de 10 cm; b) obstáculo a uma distância maior do que 10 cm.

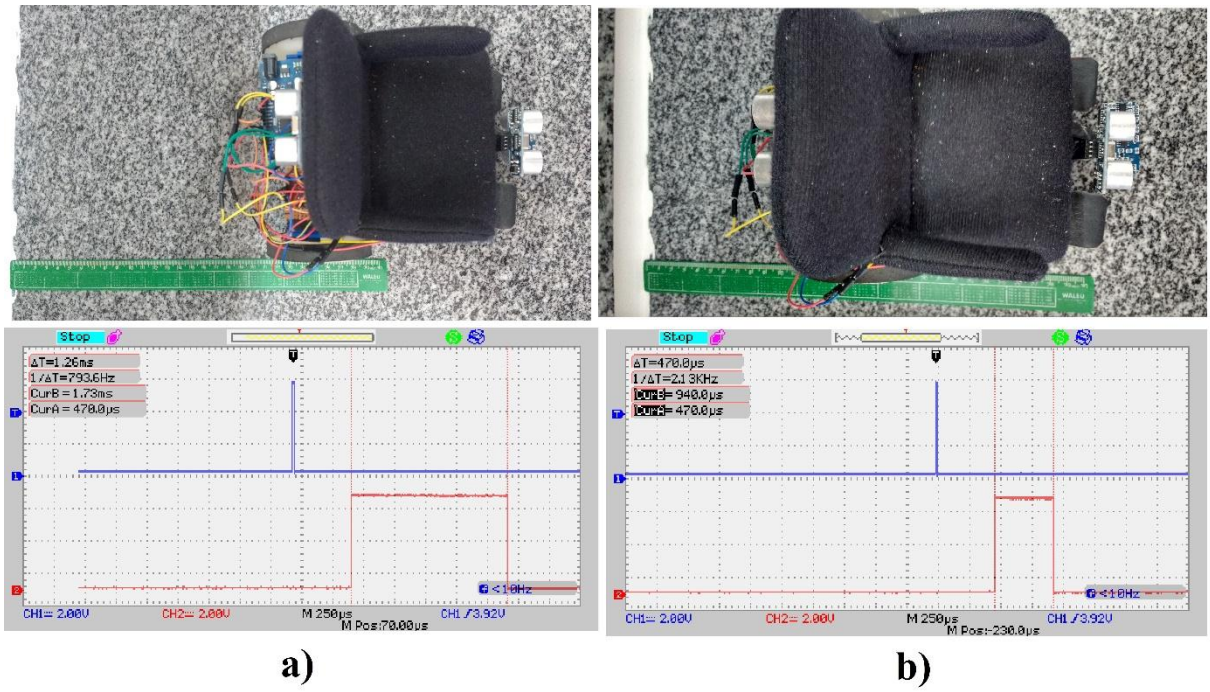


Fonte: Próprio Autor.

O protótipo estava configurado para detectar obstáculos na posição frontal, fazendo com que o mesmo parasse a uma distância pré-determinada de dez centímetros e em seguida se movimentasse no sentido contrário com o intuito de se afastar do obstáculo a uma distância maior do que a determinada pelo código. Pode-se observar neste teste a forma de onda do sensor detector com e sem a presença de obstáculo. Em azul, o *trigger* enviava um pulso de 10 microssegundos para que o sensor iniciasse a leitura. Em seguida o módulo HC-SR04 enviava pulsos de 40 kHz e aguardava o retorno do sinal pelo receptor *echo*, em vermelho, que permanecia em nível lógico alto por um período de tempo igual ao período de propagação da onda. Dessa forma, verificou-se que o protótipo obedeceu de forma eficiente o *firmware* desenvolvido.

No terceiro teste realizado posicionou-se um segundo sensor detector de obstáculos (HC-SR04) na parte posterior do protótipo. Este teste tinha o mesmo objetivo do teste anterior, que foi detectar objetos, pessoas ou qualquer outro obstáculo, agora que impedisse a movimentação do protótipo quando o mesmo se movimentasse para trás. A Figura 43 mostra como foi feito este teste.

Figura 43: Detecção de obstáculo posterior: a) obstáculo a uma distância de 20 cm; b) obstáculo a uma distância maior do que 20 cm.



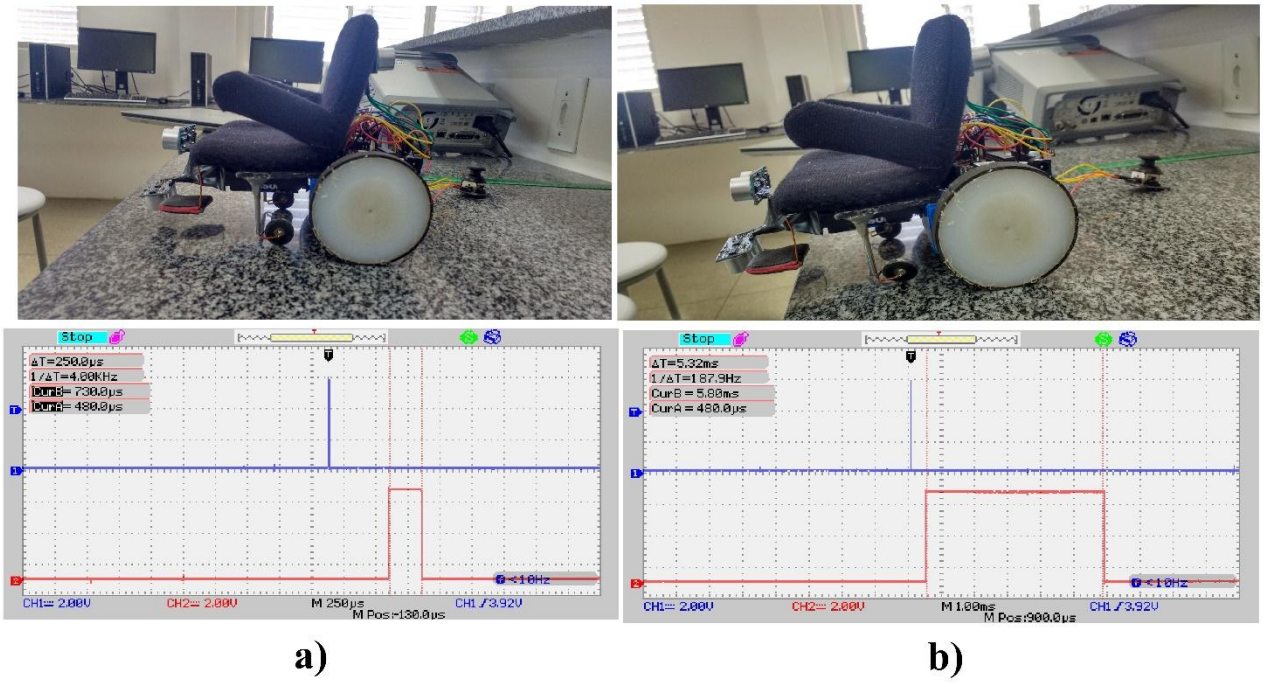
Fonte: Próprio Autor.

O protótipo estava configurado para detectar obstáculos na posição posterior, obedecendo à mesma função do sensor frontal, ou seja, fazendo com que o mesmo parasse a uma distância pré-determinada de dez centímetros e em seguida se movimentasse no sentido contrário com o intuito de se afastar do obstáculo a uma distância maior do que a determinada pelo código. A forma de onda do sensor detector de obstáculo mostra que o pino *echo* (vermelho) permaneceu em nível lógico alto igual ao período de tempo de propagação da onda, ou seja, quando o protótipo está a uma distância menor do que dez centímetros, o tempo de propagação da onda é mais rápida do que a uma distância maior que dez centímetros. Dessa forma, verificou-se que o protótipo obedeceu mais uma vez de forma eficiente o *firmware* desenvolvido.

O último teste realizado consistia em verificar a capacidade do protótipo em detectar níveis do ambiente, como buracos, calçadas ou qualquer outro nível que pudesse comprometer o seu movimento. Para isso, foi utilizado um terceiro sensor detector de obstáculos (HC-SR04) que possuía as mesmas características dos sensores anteriores. Dessa forma, o sensor três foi posicionado na parte frontal do protótipo, no sentido vertical.

Neste teste colocou-se o protótipo em uma mesa para detectar níveis maiores do que o piso, simulando a presença de calçadas. A Figura 44 ilustra o teste.

Figura 44: Detecção de níveis; a) nível igual do piso; b) nível maior que o piso.



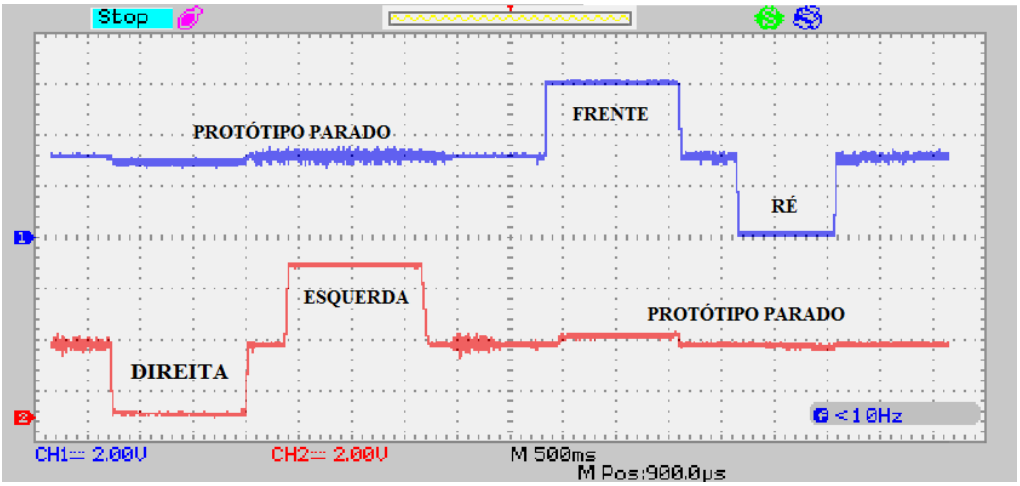
Fonte: Próprio Autor.

O protótipo estava configurado para detectar níveis maiores do que o do piso no qual ele estava inserido. Logo, se a distância do nível do piso ao sensor vertical fosse maior do que dez centímetros, o protótipo se movimentava na direção oposta com o objetivo de se afastar o máximo do nível detectado para não gerar um possível acidente. Dessa forma, o sensor vertical trabalha de forma oposta dos outros sensores. Como a distância do sensor ao piso é considerada muito pequena, o pino *echo* (vermelho) permanece em nível lógico alto por um curto intervalo de tempo. Quando o protótipo encontra um desnível, a distância do novo ‘’nível’’ do piso ao sensor se torna maior, fazendo com que o pino *echo* do sensor permaneça em nível lógico alto por um período de tempo igual ao período de propagação da onda refletida. Portanto, o protótipo respondeu de forma eficiente o *firmware* implementado.

5.2 Teste com *Joystick*

Para verificar o funcionamento do joystick, realizou-se testes dos sinais fornecidos pelo módulo do sensor. A Figura 45 mostra a forma de onda do sensor para cada movimentação do protótipo: direita, esquerda, frente e ré.

Figura 45: Forma de onda do sensor Joystick.

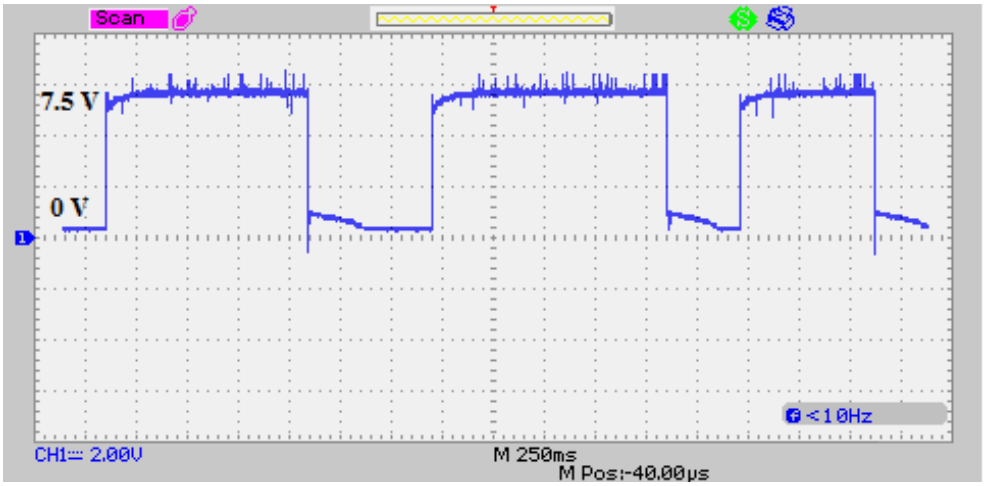


Fonte: Próprio Autor.

5.3 Teste dos PWM

Para verificar o funcionamento dos motores de tração, realizou-se testes dos sinais PWM oferecidos pela plataforma Arduino Uno. Como visto no capítulo 3, a plataforma Arduino Uno possui uma resolução de 8 bits nas portas PWM, ou seja, o valor máximo que essas portas oferecem é de 255 que corresponde a tensão da fonte de alimentação. O protótipo foi configurado para receber o valor máximo do sinal PWM (255 que corresponde a 7.5V) para todos os casos vistos no tópico 5.1. Portanto, as formas de ondas dos sinais PWM para os motores podem ser observadas nas Figuras 46 e 47.

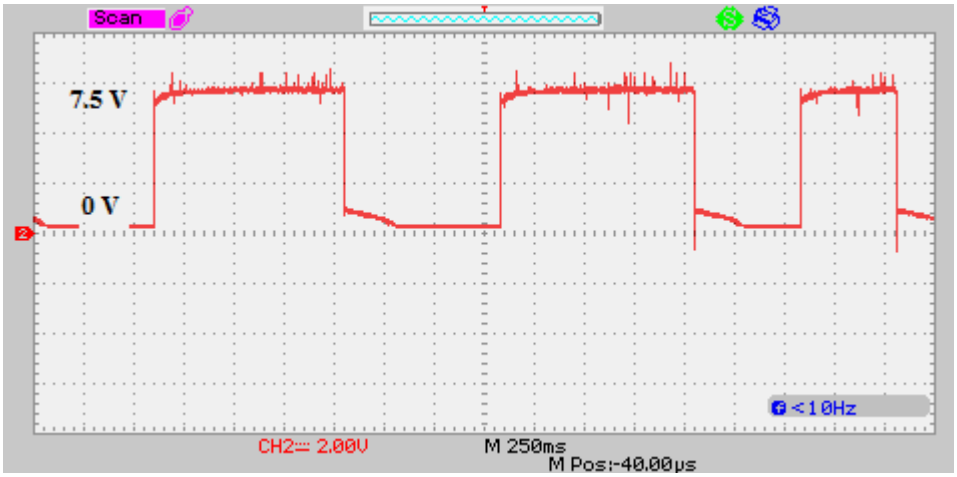
Figura 46: Forma de onda do sinal PWM para o primeiro motor.



Fonte: Próprio Autor.

Quando os motores eram acionados por meio do comando do joystick, o formato da onda do sinal PWM deixava de ser 0 (0V), protótipo parado, e passava a ser 255 (7.5V), protótipo em movimento. A Figura 47 mostra a forma de onda do sinal PWM para o segundo motor de tração.

Figura 47: Forma da onda do sinal PWM para o segundo motor.



Fonte: Próprio Autor.

Pode-se observar que no sinal PWM existe a presença de ruídos devido ao seu chaveamento.

Portanto, a realização dos testes teve como objetivo verificar o funcionamento do protótipo desenvolvido, para constatando as práticas de eletrônica e computação desenvolvidas ao longo deste trabalho.

6 CONCLUSÃO

O uso de tecnologias voltadas para pessoas que possuem necessidades especiais é bastante importante, pois é a partir do conhecimento técnico-científico que são desenvolvidas e/ou aprimoradas técnicas para melhorar a qualidade de vida dessa parte da população que na maior parte dos casos é excluída do convívio social.

Neste trabalho foi desenvolvido uma automação implementada em um protótipo de uma cadeira rodas para deficientes utilizando sensores detectores de obstáculos e microcontrolado pela plataforma Arduino UNO. Assim, conseguiu-se cumprir todos os objetivos, desenvolvendo um *firmware* para controlar o protótipo em questão.

Portanto, em qualquer projeto de automação devem-se ter as prioridades bem definidas, um pré-planejamento de execução ao objetivo geral e objetivo específico, bem como, das dimensões, custo, benefícios, aplicabilidade e usabilidade dentro de um contexto cotidiano em que a "robótica" reintegra o bem estar dos portadores de deficiência motora.

Para trabalhos futuros propõe-se: utilização de mais sensores para tornar o protótipo mais independente do usuário; implementação de um filtro capacitivo para filtrar o ruído gerado pelo chaveamento do sinal PWM; utilização do eletro-oculografia para deficiente que tenham paralisia total do corpo; implementação de um protocolo de comunicação para mapeamento de obstáculos utilizando o filtro de Kalman.

REFERÊNCIAS

ARDUINO. **Arduino Uno.** Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardUno>.

CASADO, Manuel. **Prácticas com Arduino-Visualino: Joystick analógico que controla cuatro LEDs**. 2016. Disponível em: <http://www.aprendetecnologia.es/practicas-con-arduino-visualino-joystick-analogico-keyes/>.

FAGUNDES, Sylviane Beck Ribeiro; GOMES, Luiz Vidal Negreiros; MEDEIROS, Ligia Maria Sampaio de. **Joystick: Uma tendência nas máquinas florestais.** Disponível em: <http://www.abepro.org.br/biblioteca/enegep1998_art092.pdf>.

FONSECA, Fabrício Ramos da. **Sensores**. 2006. Disponível em: <http://www.adororobotica.com/Sensores.pdf>.

FONSECA, Erika Guimara˜es Pereira da; BEPPU, Mathyan Motta. **Apostila Arduino.** 2010. Disponível em: <http://www.sr.ifes.edu.br/~eduardomax/arquivos/Tut_Arduino.pdf>.

FUTIDA, Ivo Takao; ROMERO, Roseli Aparecida Francelin**. Desenvolvimento mecânico e de Controle PWM para sistema robótico.** Disponível em: <http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_331.pdf>.

GIOPPO, Lucas Longen et al. **Robô seguidor de linha**. 2009. Disponível em: <http://paginapessoal.utfpr.edu.br/msergio/portuguese/ensino-de-fisica/oficina-de-integracao-ii/oficina-de-integracao-ii/Monog-09-2-Seguidor-de-linha.pdf>.

GUIMARÃES, Fábio de Almeida**. Desenvolvimento de um robô móvel utilizado para exploração de ambientes hostis.** 2007. Disponível em: <http://maua.br/files/dissertacoes/desenvolvimento-de-robo-movel.pdf>.

INDÚSTRIA, Procel. **Motor Elétrico**. 2009. Disponível em: <http://arquivos.portaldaindustria.com.br/app/conteudo_18/2014/04/22/6281/Motor_eletrico.pdf>.

MOREIRA, Aldeir de Souza. **Cadeira de rodas inteligente: O uso de técnicas de programação, automação e eletrônica para maior autonomia e qualidade de vida dos cidadãos de mobilidade reduzida.** 2012. Disponível em: <https://drive.google.com/folderview?id=0B44hbd16QxHSX0JOMk5lNXhRaTg&usp=sharing>.

PAREDE, Ismael Moura; GOMES, Luiz Eduardo Lemes. Eletrônica: **Automação Industrial.** 2011. Disponível em: <http://eletro.g12.br/arquivos/materiais/eletronica6.pdf>.

OLIVEIRA, Gustavo C.. **Acionamento de Motores: PWM e Ponte H.** 2014. Disponível em: <https://www.assembla.com/spaces/warthog-dc/documents/dclHqyPner46_cacwqEsg8/download/dclHqyPner46_cacwqEsg8>.

PATSKO, Luís Fernando. **Tutorial, Aplicações, Funcionamento e Utilização de Sensores.** 2006. Disponível em: <http://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_aplicacoes_e_funcionamento_de_sensores.pdf>.

SILEIRA, Leonardo; LIMA, Weldson Q.. **Um breve histórico conceitual da Automação Industrial e Redes para Automação Industrial**. 2013. Disponível em: <http://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_13.pdf>.

SILVEIRA, João Alexandre da. **Arduíno: Cartilha para programação.** 2012. Disponível em: <http://ordemnatural.com.br/pdf-files/CartilhadoArduino_ed1.pdf>.

SILVA, Igor Thyerry Vieira da. **Maquinas de corrente contínua.** 2013. Disponível em: <http://www.ebah.com.br/content/ABAAAgbCEAK/maquinas-corrente-continua#>.

SILVA, Otto Marques da; DEL'ACQUA, Ricardo José. 2012. **Cadeiras de rodas e sua evolução histórica.** Disponível em: <http://www.crfaster.com.br/Cadeira Rodas.htm>.

WENDLING, Marcelo. **Sensores.** 2010. Disponível em: <http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf>.

APÊNDICE

Código completo do protótipo inteligente de cadeira de rodas no auxílio para deficientes.

```
#include <Ultrasonic.h>                          //Carrega a biblioteca do sensor ultrassonico

Ultrasonic ultrasonicHor1(2, 4);                 //Trig 2; Echo 4
Ultrasonic ultrasonicHor2(7, 8);                 //Trig 7; Echo 8
Ultrasonic ultrasonicVer(12, 13);                //Trig 12; Echo 13

int calibraHorizontal1 = 10;          //Variável calibraHorizontal1 inicializado como valor 10
int calibraHorizontal2 = 10;          //Variável calibraHorizontal2 inicializado como valor 10
int calibraVertical = 10;             //Variável calibraVertical inicializado como valor 10

float leitura1 = 0;                   //Variável leitura1 inicializado em nível baixo
float leitura2 = 0;                   //Variável leitura2 inicializado em nível baixo
float leitura3 = 0;                   //Variável leitura3 inicializado em nível baixo

float cmMsec1 = 0;                    //Variável cmMsec1 inicializado em nível baixo
float cmMsec2 = 0;                    //Variável cmMsec2 inicializado em nível baixo
float cmMsec3 = 0;                    //Variável cmMsec3 inicializado em nível baixo

long   leituraHorizontal1 = 0;        //Variável leituraHorizontal1 inicializado em nível baixo
long   leituraHorizontal2 = 0;        //Variável leituraHorizontal2 inicializado em nível baixo
long   leituraVertical = 0;           //Variável leituraVertical inicializado em nível baixo

const int JoyX = A0;                  //Joystick X conectado ao Pino A0 do Arduino
const int JoyY = A1;                  //Joystick Y conectado ao Pino A1 do Arduino

int JoyX1 = 0;                        //Variável JoyX1 inicializado em nível baixo
int JoyY1 = 0;                        //Variável JoyX2 inicializado em nível baixo

int leiturax = 0;                     //Variável leiturax inicializado em nível baixo
int leituray = 0;                     //Variável leituray inicializado em nível baixo

const int Motor1Pin1 = 11;            //Pino 2 do L293D no Pino 11 do Arduino
const int Motor1Pin2 = 10;            //Pino 7 do L293D no Pino 10 do Arduino
const int Motor2Pin1 = 5;             //Pino 15 do L293D no Pino 7 do Arduino
const int Motor2Pin2 = 6;             //Pino 10 do L293D no Pino 8 do Arduino
const int PWMmotores = 9;       //Enables ativado Pino 9 do L293D no Pino 9 do Arduino
```

ROTINA DE CONFIGURAÇÃO

```
void setup() {

  pinMode(JoyX, INPUT);               //Joystick eixo X configurado como entrada
  pinMode(JoyY, INPUT);               //Joystick eixo Y configurado como entrada
```

```
  pinMode(Motor1Pin1, OUTPUT);                    //Motor 1.1 configurado como saída
  pinMode(Motor1Pin2, OUTPUT);                    //Motor 1.2 configurado como saída
  pinMode(Motor2Pin1, OUTPUT);                    //Motor 2.1 configurado como saída
  pinMode(Motor2Pin2, OUTPUT);                    //Motor 2.2 configurado como saída
  pinMode(PWMmotores, OUTPUT);                  //PWM(motor) configurado como saída
  analogWrite(PWMmotores, 255);                                 //Habilita PWM em 100%

}
```

ROTINA PRINCIPAL

```
void loop()
{

  while (true)
  {

    for (int x = 0; x < 5; x++) {                    // Executa 5 vezes para obter uma média
      JoyX1 = analogRead(JoyX);              //Armazena os valores de leitura do Joystick eixo
X
      delay(1);                                        //Aguarda um tempo de 1 milissegundo
      JoyY1 = analogRead(JoyY);              //Armazena os valores de leitura do Joystick eixo
Y
      delay(1);                                        //Aguarda um tempo de 1 milissegundo

      leiturax = (leiturax + JoyX1);                //Armazena os valores da varivável JoyX1
      leituray = (leituray + JoyY1);                //Armazena os valores da varivável JoyY1

    }

    leiturax = (leiturax / 5);                                  //Obtém a média do JoyX
    leituray = (leituray / 5);                                  //Obtém a média do JoyY

    leiturax = (leiturax >> 2);                              //Habilita sensor JoyX em 8 bits
    leituray = (leituray >> 2);                              //Habilita sensor JoyY em 8 bits
```

TESTA SE O JOYSTICK FOI DETECTADO

```
    //Botão frente
    if (((leituray >= 205) && (leituray <= 255)) && ((leiturax >= 100) && (leiturax <= 150)))
    {
      detectaFrente();                                        //Sensores frontais são ativados
      frente();                                                  // Protótipo anda pra Frente
    }

    //Botão Ré
    if (((leituray >= 0) && (leituray <= 20)) && ((leiturax >= 100) && (leiturax <= 150)))
    {
```

```
    detectaRe();                                                    //Sensor traseiro é ativado
    re();                                                           //Protótipo anda pra trás

   }

   //Botão direita
   if (((leituray >= 100) && (leituray <= 150)) && ((leiturax >= 0) && (leiturax <= 20)))
   {
    direita();                                                      //Protótipo anda pra direita
   }

   //Botão esquerda
   if (((leituray >= 100) && (leituray <= 150)) && ((leiturax >= 205) && (leiturax <= 255)))
   {
    esquerda();                                                     //Protótipo anda pra esquerda
   }

   //Parado
   if (((leituray >= 100) && (leituray <= 150)) && ((leiturax >= 100) && (leiturax <= 150)))
   {
    parar();                                                        //Protótipo permanece parado
   }

   JoyX1 = 0;                                                       //Zera variável JoyX1
   JoyY1 = 0;                                                       //Zera variável JoyY1
   leiturax = 0;                                                    //Zera variável leiturax
   leituray = 0;                                                    //Zera variável leituray

  }
}
```

ROTINA DE CALIBRAÇÃO DOS SENSORES FRONTAIS

```
void detectaFrente() {

   boolean valor = true;
   while(valor){

     for (int y=0; y<3; y++) {                                  //Executa 3 vezes para obter uma média

      leituraHorizontal1 = ultrasonicHor1.timing();       //Ativa a leitura do sensor Horizontal 1

      //Lê as informações do sensor Horizontal 1 em centímetros.
      cmMsec1 = ultrasonicHor1.convert(leituraHorizontal1, Ultrasonic::CM);

       delay(1);                                                //Aguarda tempo de 1 milissegundos

      leituraVertical = ultrasonicVer.timing();                 //Ativa a leitura do sensor Vertical

      //Lê as informações do sensor Vertical em centímetros.
```

```
      cmMsec3 = ultrasonicVer.convert(leituraVertical, Ultrasonic::CM);

      delay(1);                                          //Aguarda tempo de 3 milissegundos

      leitura1 = (leitura1 + cmMsec1);                  //Guarda os valores do sensor Horizontal
1
      leitura3 = (leitura3 + cmMsec3);                   //Guarda os valores do sensor Vertical

      }

      leitura1 = (leitura1 / 3);   //Obtém a média dos valores oferecidos pelo sensor Horizontal
1
      leitura3 = (leitura3 / 3);            //Obtém a média dos valores oferecidos pelo sensor
Vertical
```

CONDIÇÕES PARA O MOTOR PARAR

```
       if ((leitura3 <= calibraVertical) || (leitura1 <= calibraHorizontal1))
       {
         parar();                                                    //Parar Protótipo
       }
       else
       {
        valor = false;
        break;
       }


       cmMsec1 = 0;                                              //Zera a variável cmMsec1
       cmMsec3 = 0;                                              //Zera a variável cmMsec2
       leituraHorizontal1 = 0;                                 //Zera a variável leituraHorizontal1
       leituraVertical = 0;                                     //Zera a variável leituraVertical
       leitura1 = 0;                                               //Zera a variável leitura1
       leitura3 = 0;                                               //Zera a variável leitura3
   }
}
```

ROTINA DE CALIBRAÇÃO DO SENSOR TRASEIRO

```
 void detectaRe(){

   boolean valor = true;
   while(valor){

     for (int y=0; y<3; y++) {                          //Executa 3 vezes para obter uma média

      leituraHorizontal2 = ultrasonicHor2.timing();      //Ativa a leitura do sensor Horizontal 2

      //Lê as informações do sensor Horizontal1 em centímetros.
```

```
      cmMsec2 = ultrasonicHor2.convert(leituraHorizontal2, Ultrasonic::CM);

      delay(1);                                          //Aguarda tempo de 3 milissegundos

      leitura2 = (leitura2 + cmMsec2);                  //Guarda os valores do sensor Horizontal 2
            }

      leitura2 = (leitura2 / 3);   //Obtém a média dos valores oferecidos pelo sensor Horizontal
2
```

CONDIÇÕES PARA O MOTOR PARAR

```
     if ((leitura2 <= calibraHorizontal2))
       {
        parar();                                                    //Parar o Protótipo
       }
       else
       {
        valor = false;
        break;
       }
      cmMsec2 = 0;                                                  //Zera a variável cmMsec2
      leituraHorizontal2 = 0;                                     //Zera a variável leituraHorizontal2
      leitura2 = 0;                                                 //Zera a variável leitura2
    }
  }
```

ROTINA PROTÓTIPO PARADO

```
void parar() {
   digitalWrite(Motor1Pin1, LOW);                  //Habilita Motor 1.1 para o nível baixo
   digitalWrite(Motor1Pin2, LOW);                  //Habilita Motor 1.2 para o nível baixo
   digitalWrite(Motor2Pin1, LOW);                  //Habilita Motor 2.1 para o nível baixo
   digitalWrite(Motor2Pin2, LOW);                  //Habilita Motor 2.2 para o nível baixo
   delay(5);                                      //Aguarda um tempo de 5 milissegundos
  }
```

ROTINA PROTÓTIPO ANDA PRA FRENTE

```
  void frente() {
   digitalWrite(Motor1Pin1, HIGH);                  //Habilita Motor 1.1 para o nível alto
   digitalWrite(Motor1Pin2, LOW);                  //Habilita Motor 1.2  para o nível baixo
   digitalWrite(Motor2Pin1, HIGH);                  //Habilita Motor 2.1 para o nível alto
   digitalWrite(Motor2Pin2, LOW);                  //Habilita Motor 2.2  para o nível baixo
   delay(5);                                      //Aguarda um tempo de 5 milissegundos
  }
```

ROTINA PROTÓTIPO ANDA PRA TRÁS

```
void re() {
  digitalWrite(Motor1Pin1, LOW);                 //Habilita Motor 1.1  para o nível baixo
  digitalWrite(Motor1Pin2, HIGH);                  //Habilita Motor 1.2  para o nível alto
  digitalWrite(Motor2Pin1, LOW);                 //Habilita Motor 2.1  para o nível baixo
  digitalWrite(Motor2Pin2, HIGH);                  //Habilita Motor 2.2  para o nível alto
  delay(5);                                     //Aguarda um tempo de 5 milissegundos
}
```

ROTINA PROTÓTIPO ANDA PRA DIREITA

```
void direita() {
  digitalWrite(Motor1Pin1, HIGH);                  //Habilita Motor 1.1  para o nível alto
  digitalWrite(Motor1Pin2, LOW);                 //Habilita Motor 1.2  para o nível baixo
  digitalWrite(Motor2Pin1, LOW);                 //Habilita Motor 2.1  para o nível baixo
  digitalWrite(Motor2Pin2, LOW);                 //Habilita Motor 2.2  para o nível baixo
  delay(5);                                     //Aguarda um tempo de 5 milissegundos
}
```

ROTINA PROTÓTIPO ANDA PRA ESQUERDA

```
void esquerda() {
  digitalWrite(Motor1Pin1, LOW);                 //Habilita Motor 1.1 para o nível baixo
  digitalWrite(Motor1Pin2, LOW);                 //Habilita Motor 1.2 para o nível baixo
  digitalWrite(Motor2Pin1, HIGH);                  //Habilita Motor 2.1 para o nível alto
  digitalWrite(Motor2Pin2, LOW);                 //Habilita Motor 2.2 para o nível baixo
  delay(5);                                     //Aguarda um tempo de 5 milissegundos
}
```