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

ANTÔNIO GEORGE DE SOUSA CRUZ

DESENVOLVIMENTO DE ROBÔ DE SUMÔ PARA COMPETICÃO

CARAÚBAS

2016

ANTÔNIO GEORGE DE SOUSA CRUZ

DESENVOLVIMENTO DE ROBÔ DE SUMÔ PARA COMPETICÃO

Monografia apresentada à Universidade Federal Rural do Semi-Árido - UFERSA, campus Caraúbas como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Orientador: Prof. Dr. Hugo Michel Camara de Azevedo Maia – UFERSA.

Co-orientador: Prof. Ms. José Ailton L. Barboza Júnior – UFERSA.

CARAÚBAS

2016

C957d Cruz, Antônio George de Sousa .

DESENVOLVIMENTO DE ROBÔ DE SUMÔ PARA COMPETICÃO / Antônio George de Sousa Cruz. – 2016.

82 f. : il.

Orientador: Hugo Michel Camara de Azevedo Maia.

Coorientador: José Ailton L. Barboza Júnior.

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

1. Robô. 2. Sumô. 3. Protótipo. 4. Programação. I. Maia, Hugo Michel Camara de Azevedo , orient.

II. Júnior, José Ailton L. Barboza , co-orient. III. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (USP) e gentilmente cedido para o Sistema de Bibliotecas da Universidade Federal Rural do Semi-Árido (SISBI-UFERSA), sendo customizado pela Superintendência de Tecnologia da Informação e Comunicação (SUTIC) sob orientação dos bibliotecários da instituição para ser adaptado às necessidades dos alunos dos Cursos de Graduação e Programas de Pós-Graduação da Universidade.

ANTÔNIO GEORGE DE SOUSA CRUZ

DESENVOLVIMENTO DE ROBÔ DE SUMÔ PARA COMPETICÃO

Monografia apresentada à Universidade Federal Rural do Semi-Árido - UFERSA, campus Caraúbas como requisito para obtenção do título de Bacharel em Ciência e Tecnologia.

Defendida em: 09 / 11 / 2016

BANCA EXAMINADORA

[redacted]

Prof. Dr. Hugo Michel de Azevedo Maia – UFERSA
Presidente

[redacted]

Prof. Ms. José Ailton L. Barboza Junior - UFERSA
Primeiro Membro

[redacted]

Prof. Dr. Francisco de Assis Brito Filho – UFERSA
Segundo Membro

[redacted]

Prof. Ms. Eliel Poggi dos Santos – UFERSA
Terceiro Membro

Em Memória,

Meus avós paternos João Onofre Cardoso e Maria Leonila e meu Avô Materno Alderi Pio, pelas saudades eternas.

Dedico,

A Deus, pelo dom da vida, pela proteção divina.

Aos Meus Pais, Francisco Gilvan da Costa Cruz e Francisca Medeiros de Sousa Cruz.

Aos meus irmãos: Francisco Gilvan da Costa Cruz Junior, Gilfran de Sousa Cruz e Francisca Georgiana de Sousa Cruz. E às minhas Cunhadas, Cunhado e Sobrinhos.

À minha namorada Mirley Priscila Lopes da Costa.

À minha avó Materna Maria Lurdes.

E a todos os amigos e familiares.

AGRADECIMENTOS

Primeiramente agradeço a Deus, pela oportunidade da vida em que ele me deu, inclusive de poder enfrentar mais esse desafio.

Aos meus pais, pelo amor, dedicação, perseverança, sabedoria de vida que me tem concedido a cada ano que se passa, procurando me guiar sempre no caminho correto, honesto e responsável da vida, me mostrando que a educação é o melhor caminho para uma vida de conquistas.

Aos meus Irmãos, que me ajudaram e me apoiaram sempre em minhas escolhas, mesmo as mais difíceis possíveis.

À minha namorada, pela paciência, por sempre me mostrar motivação quando não a enxergava mais e por sempre me dando a mão nas minhas falhas.

Ao meu orientador professor Dr. Hugo Maia, por ter me ajudado de braços abertos com seu conhecimento neste trabalho.

Ao meu co-orientador professor Ms. Ailton Junior, por toda ajuda que proporcionou e por ter me mostrado que todo desafio sem esforço e dedicação não tem vitória.

Ao meu amigo Isael Calistrato, por ter mostrado que amizade não está só nas conquistas, mas também nas dificuldades, me ajudando com seu conhecimento neste trabalho.

À todos meus amigos e familiares que me apoiaram de alguma forma, diretamente ou indiretamente.

Vencer na Vida não é questão de quem é mais inteligente, mais forte ou mais habilidoso, e sim de ser dedicado, de ser determinado e audacioso. Enfrente o impossível.

RESUMO

Neste trabalho foi desenvolvido um protótipo de robô de sumô para competição. Há o uso de elementos e técnicas de computação, automação e eletrônica que são de grande importância para o desenvolvimento de um protótipo que possibilite desviar de possíveis oponentes que causem colisão ao robô. Inicialmente, é feita uma abordagem dentro de todo o contexto da robótica educacional de aplicações nesta área com o uso da robótica. Após, é apresentado no capítulo suas principais características, bem como o seu desenvolvimento ao longo da história. Em seguida, é abordado o desenvolvimento do protótipo. Finalmente, é descrita a montagem do protótipo utilizando, sua programação no apêndice e o tempo de resposta nos resultados.

Palavras-chave: Robô. Sumô. Protótipo. Programação.

ABSTRACT

In this work a prototype sumo robot for competition was developed. There is the use of elements and techniques of computation, automation and electronics that are of great importance for the development of a prototype that allows to deviate of possible opponents that can cause a collision to the robot. Initially an approach is made within the entire context of educational robotics of applications in this area with the use of robotics. Afterwards, the chapter presents its main characteristics as well as its development throughout history. Then we talk about the development of the prototype. Finally we describe the assembly of the prototype using, its programming in the appendix and response time in the result.

Key-words: Robot. Sumo. Prototype. Programming.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CA	*Alternat Current* (Corrente Alternada)
CC	*Direct Current* (Corrente Contínua)
CI	*Integrated Circuit* (Circuito Integrado)
CLP	*Programmable Logic Controller* (Constrolador Lógico Programável)
CPU	*Central Processing Unit* (Unidade Central de Processamento)
I/O	*Input/Output* (Entrada/Saída)
IR	*Infrared* (Infrevermelho)
kHZ	*Kilohertz*
MOSFET	*Metal Oxide Semiconductor Fiel Effect Transistor* (Transistor de Efeito de Campo Metal Óxido Semicondutor)
PCB	*Printed Circuit Board* (Placa de Circuito Impresso)
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)
PSD	*Position Sensing Device* (Dispositivo de Monitoramento de Posição)
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
VGS	Tensão do Gate ao Source

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais, a Robótica é uma realidade crescente a cada dia. É comum observar o uso de robôs em diversos segmentos da sociedade, realizando tarefas domésticas como aspirar de pó e limpar dutos, bem como robôs para competição, robôs educacionais, e outros com tarefas mais complexas como exploração subaquática e espacial, limpeza de lixo tóxico, cirurgias médicas, mineração, busca de minas terrestres, montagem de automóveis entre outros. A robótica traz soluções rápidas e eficientes, proporcionando, entre outros benefícios, um retorno rentável, mesmo que, inicialmente apresente um custo relativamente alto para sua implementação. Pode-se dizer que um dos grandiosos passos da Robótica foi a sua inserção na educação, a chamada Robótica Educacional.

No Brasil, principalmente a partir da década de 90, ela vem sendo implementada como uma ferramenta de auxílio para a assimilação de conceitos teóricos de diversas disciplinas. Tem-se buscado caminhos através do Ensino da Robótica para oferecer ao estudante um melhor preparo para enfrentar a competitividade do mundo globalizado. A intensão da inserção da robótica na educação é ajudar de forma mais fácil a construção do conhecimento, através de experiências práticas voltadas para realidade, incentivando a criatividade e inteligência dos estudantes.

Atualmente, um dos objetivos da Robótica é criar novas tecnologias de implementação, criar novos componentes eletrônicos que suportem maior capacidade e velocidade computacional de processamento, ampliar o desenvolvimento de sistemas embarcados, proporcionar fabricação em massa de componentes simples e de baixo custo, proporcionar a criação de várias soluções eficazes na resolução de problemas, possibilitar a existência de uma gama possibilidades de manipulação dessas tecnologias em distintas áreas.

1.1 Problema e justificativa

Com o desenvolvimento de competições de robótica, como o RoboCup, UFC de máquinas e desafio de inteligência de robôs, esse projeto tem em vista direcionar a Universidade Federal Rural do Semi-Árido para tipo de pesquisa, desafio e conquista.

Atualmente o Brasil tem boas referências internacionalmente com esse tipo de atividade acadêmica. No ano de 2015, os alunos de engenharia da PUC-Rio conquistaram 6 medalhas, sendo 3 de ouros, nas olímpiadas de robôs na Califórnia – EUA. Em 08 de julho desse ano, a equipe RioBotz, uma equipe de robótica da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), com seu robô Minotaur, 100% brasileiro, venceu um robô norte-americano no UFC de máquinas promovido pelo programa de BattleBots, no canal de TV ABC dos EUA.

Com o crescimento da UFERSA e deste tipo de estudo acadêmico, a Instituição tem bons motivos de incentivar esse desenvolvimento, como forma de proporcionar novas pesquisas na área de robótica, abrangendo tanto a área da eletrônica, da programação de microcontroladores e também uma boa parte da mecânica.

1.2 Objetivo

O objetivo desse trabalho é desenvolver um protótipo de robô totalmente autônomo de no máximo 3Kg que se mova por esteira, e que esteja preparado para competição de robô de sumô, com todo sistema embarcado.

1.2.1 Objetivos específicos

a) Desenvolver um firmware que tenha estratégia para competição de robô de sumô, utilizando sensores e atuadores;
b) Criar um protótipo mecânico de robô totalmente autônomo de no máximo 3Kg que se mova por esteira;
c) Confeccionar a placa para o microcontrolador, que será a placa principal, a placa dos sensores e da ponte H para controlar os atuadores.

Neste capítulo, serão apresentadas as regras gerais do robô de sumô, conforme Regras de Sumô por RoboCore.

1.3 Organização do trabalho

Para tornar mais compreensível, esse trabalho foi organizado em 7 capítulos, como mostrado abaixo:

Capítulo 2, breve explanação sobre a robótica, automação e os dispositivos periféricos, sensores e atuadores.

Capítulo 3, explanação sobre a competição de robô de Sumô.

Capítulo 4, breve explicação sobre a metodologia adotada para o desenvolvimento desse projeto.

Capítulo 5, apresentação do hardware, com todos os circuitos e componentes aplicados.

Capítulo 6, o firmware, onde serão apresentadas a estratégia de combate, o diagrama de blocos do algoritmo e a programação do mesmo.

Capítulo 7, abordagem dos resultados da pesquisa obtidos através da prática ocorrida no projeto, levantando possíveis problemas e soluções.

2 ROBÓTICA

Leis da Robótica: Asimov (REZENDE, 1992), criador do termo “Robótica” elaborou três leis que até hoje são utilizadas como referência para a criação de novas tecnologia nessa área, permanecendo vivas nas regras da comunidade científica. São elas:

- **Primeira Lei:** Um robô não pode ferir um ser humano ou, por omissão, permitir que um ser humano sofra algum mal;
- **Segunda Lei:** Um robô deve obedecer às ordens que lhe sejam dadas por seres humanos, exceto nos casos em que tais ordens contrariem a Primeira Lei;
- **Terceira Lei:** Um robô deve proteger sua própria existência desde que tal proteção não entre em conflito com a Primeira e Segunda Lei.

2.1 Automação

Não há como falar da Robótica, sem antes ter uma boa explanação sobre a automação, já que a robótica também se complementa de algumas características suas, se não a maioria.

Automação (do latim *Automatus*, que significa mover-se por si) é um sistema automático de controle pelo qual os mecanismos verificam seu próprio funcionamento, efetuando medições e introduzindo correções sem a necessidade da interferência do homem. Em seu uso moderno, a automação pode ser definida como uma tecnologia que utiliza comandos programados para operar um dado processo, combinados com retroação de informação para determinar que os comandos sejam executados corretamente Frequentemente utilizada em processos antes operados por seres humanos, é a aplicação de técnicas computadorizadas ou mecânicas para diminuir o uso de mão-de-obra em qualquer processo, especialmente o uso de robôs nas linhas de produção. A automação diminui os custos e aumenta a velocidade da produção (LACOMBE 2004).

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 modelo que pode ser empregado nesse assunto é a automação industrial que passou a existir no século XVIII, com a eclosão das grandes revoluções industriais. Segundo Moreira (2012), têm-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 cálculos enviando respostas para os equipamentos que tem a função de atuadores, com o intuito de alcançar uma produção de máxima eficiente.

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

Figura 1: a) armário de relés; b) armário de CLPs.

Fonte: a) Parede; Gomes, 2011; b) Wikiwand, 2015.

O avanço da automação está ligado com a eclosão da indústria microeletrônica e da informática, mais precisamente com o surgimento dos CLPs (Controlador Lógico Programável), que substituam os relés, dispositivos mecânicos utilizados na época de 60. Os CLPs são dispositivos capazes de controlar máquinas através de comandos gravados em sua memória programável. Esse dispositivo vem evoluindo com o tempo, adquirindo funções como execução, sequenciamento, temporização e entre outras (MOREIRA, 2012). Na figura 1, podemos verificar fisicamente a diferença de um sistema de relés para o de CLP.

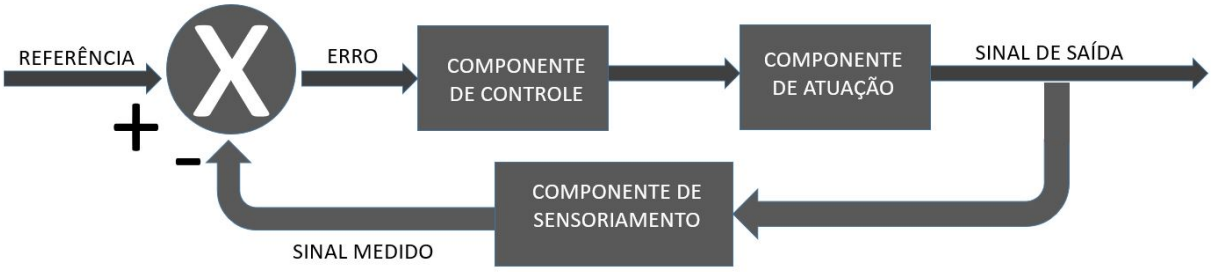
2.2 Componentes da automação

De acordo com Moreira (2012), os fundamentais elementos de automação utilizados em grandes empresas, como, indústria automobilística, petroquímica e até mesmo em supermercados carecem de múltiplas repetições de tarefas, que na maior parte dos casos são bastante complexas. Portanto, qualquer sistema de automação tem estrutura básica da seguinte forma:

a) **Acionamento:** conhecido também como atuadores, esses componentes são responsáveis por colocar em prática a função estabelecida. Como motores elétricos e pistões;
b) **Sensoriamento:** é responsável pelo reconhecimento do ambiente hostil. Exemplo: sensor infravermelho e sensor fim-de-curso;
c) **Controle:** com base na informação do comparador, o dispositivo de controle, comanda o funcionamento dos atuadores. Exemplo: Ponte H;
d) **Comparador:** compara os valores lidos pelos sensores com os valores pré-estabelecidos, mandando informação para o controle. Exemplo, os softwares de automação.

O componente comparador recebe os sinais medidos pelo componente de sensoriamento da referência e compara os valores recebidos com os valores pré-estabelecidos gravados na memória da CPU programada pelo software. Desta forma, se for necessário ativar algum atuador, o componente comparador envia um sinal para o Algoritmo de controle, que seleciona e comanda o componente de atuação. Esse ciclo continua com o componente de sensoriamento que realimenta o circuito enviando os sinais medidos com as informações encontradas no ambiente hostil para o componente comparador. A figura 2 mostra o comportamento do sistema de automação, através da equação do cálculo de erro.

Figura 2: Elementos de automação.



Fonte: Próprio do autor.

Segundo Moreira (2012), existem vários arranjos na automação que se modificam com as áreas de aplicações, como, automação industrial, comercial, de transporte e residencial. Cada uma tem particularidades e têm com o mesmo objetivo a serem alcançados: elevar ao máximo a eficiência nas aplicações determinadas com a menor ajuda possível, ou sem a intervenção, do homem.

2.3 Sensores

Segundo Fonseca (2006), sensores são dispositivos que têm a capacidade de perceber uma grandeza física e comunicar-se com um indicador, dispositivo responsável por indicar visualmente a grandeza medida, ou com um controlador. Dessa forma, esses dispositivos são sensíveis a algum tipo de energia do ambiente, podendo ser luminosa, térmica e cinética.

De acordo com Patsko (2006), sensores podem ser definidos como “aquilo que sente”. Na área da eletrônica, esses dispositivos podem ser observados como qualquer componente ou circuito eletrônico que possibilite a apreciação de uma determinada situação do ambiente, que pode ser básica como temperatura ou luminosidade, ou complexa como detectar partículas subatômicas e radiação cósmicas.

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 serem utilizadas em um sistema de controle, dessa forma, se faz necessário manipular o sinal de saída antes da sua leitura no sistema de controle. Por exemplo, quando o sensor apresenta uma tensão de saída

muito baixa quando ativado por uma energia externa, se faz necessário a amplificação desse sinal para que o controlador possa ler tal informação (WENDLING 2010).

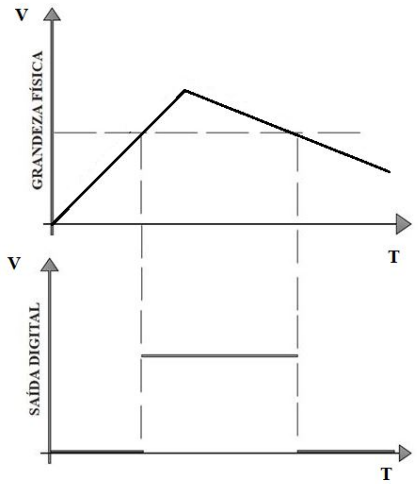
Ainda segundo Wendling (2010), existem dois tipos de sensores, os sensores analógicos e os digitais. O primeiro assume diferentes valores no sinal de saída ao decorrer do tempo, desde que esteja incluso na sua faixa de trabalho. O segundo só admite dois valores no sinal de saída (0 ou 1) ao decorrer do tempo. É conhecido que não existem grandezas físicas que consigam atribuir-se a esses valores, 0 ou 1, porém são expostas nessa configuração a um sistema de controle, após serem convertidas, normalmente por um comparador no circuito eletrônico.

Existe uma gama de caraterísticas que necessitam ser levadas em consideração na hora da seleção do sensor que será empregado para uma particular aplicação. As fundamentais 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 somente dois valores lógicos (0 ou 1). Esse tipo de saída só determina se uma grandeza física alcançou ou não um valor pré-estabelecido. A figura 3 esboça a saída de um sensor digital em função da variação de entrada ao decorrer do tempo.

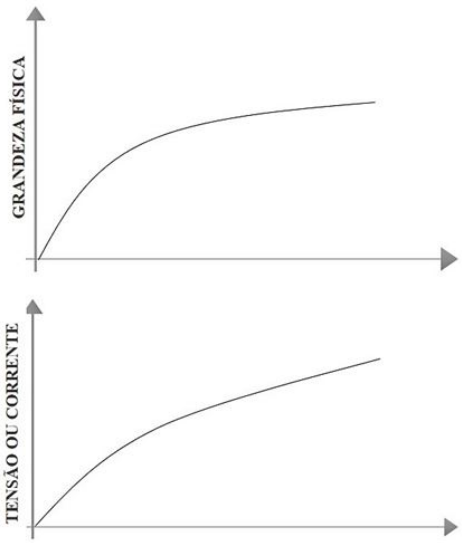
Figura 3: Saída digital ou binária de um sensor.



Fonte: Anderson, 2015.

Analógica: esse tipo de sensor pode admitir múltiplos valores ao decorrer do tempo, desde que se encontre dentro da sua faixa de trabalho. A figura 4 esboça um modelo de sensor analógico, representando tensão ou corrente de saída versus grandeza física em uma faixa de tempo determinada.

Figura 4: Saída analógica de um sensor.

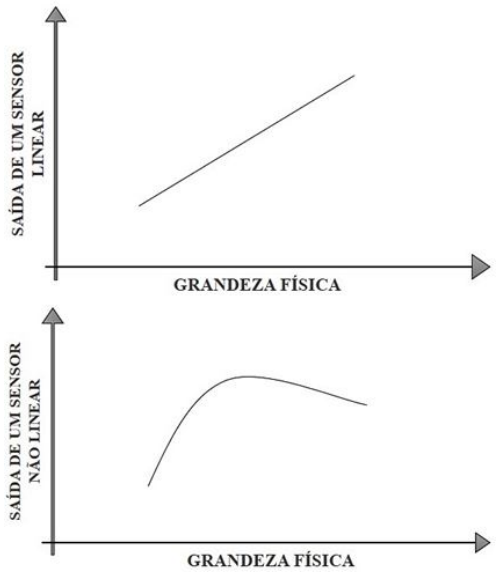


Fonte: Anderson, 2015.

b) Linearidade:

Aplicada a sensores analógicos, a linearidade é uma curva de saída, a partir da grandeza medida. O objetivo é procurar respostas medidas as entradas, objetivando promover com facilidade a montagem do circuito de interface. Nem sempre a linearidade pode ser obtida, pois existem sensores não lineares. A figura 5 ilustra a diferença entre um sensor linear e um não linear.

Figura 5: Saída de um sensor linear e não linear.



Fonte: Anderson, 2015.

c) Alcance:

Representa todas as possibilidades de valores de entrada do sensor. Dessa forma, a precisão da resposta de saída do sensor dependerá do máximo alcance atingido.

d) Velocidade de resposta:

É a diferença entre o tempo que o sensor lê a mudança da situação do ambiente, e o tempo de resposta. O ideal seria que essa diferença fosse o mais próximo possível de zero, ou seja, o sensor teria uma resposta imediata. Sensores de baixa velocidade podem depreciar o sistema de controle.

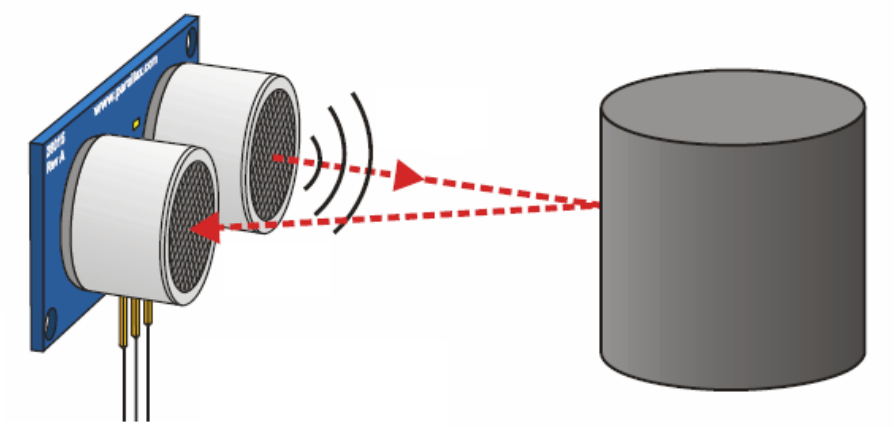
Portanto, existem diversos sensores que são bastante aplicados nas linhas de produções automatizadas como os sensores ultrassônicos e os infravermelhos. As principais aplicações desses sensores são na mediação das dimensões de objetos, controle de posicionamento e verificação de danos ou falhas dos produtos (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 considerável e sejam capazes de refletir a radiação ultrassônica emitida 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 42kHZ. As ondas se propagam pelo meio e quando colidem com algum objeto são refletidas. As ondas refletidas são captadas pelo sensor e, com base no tempo de resposta desde sua incidência até seu retorno, se torna possível obter informações sobre a distância do objeto que refletiu o sinal. A figura 6 mostra o princípio de funcionamento de um sensor ultrassônico.

Figura 6: Funcionamento de um sensor ultrassônico.



Fonte: Portal vidadesilicio http://blog.vidadesilicio.com.br/arduino/sensor-ultrassom-hc-sr04/

Geralmente esses sensores são compostos por um gatilho (emissor) e um eco (receptor), funcionando como um sonar, ou seja, um pulso ultrassônico é incidido no meio esperando pelo retorno de um eco. A distância do objeto ao sensor 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 ultrassônico é aproximadamente 340m/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:

$$V = \frac{distância}{tempo}$$

$$V = \frac{2d}{t} \tag{2.0}$$

Deixando o $2d$ em função do tempo e sabendo que a velocidade do som é de 340 m/s, temos que a distância é de:

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

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

$$d = 170\ x\ t \tag{2.1}$$

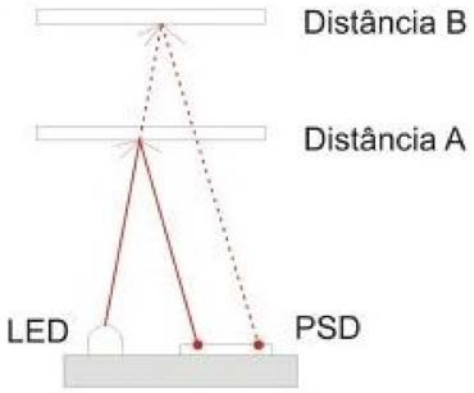
São diversas as aplicações dos sensores ultrassônicos na automação como ferramenta de detecção sem contato, possibilitando que objetos de várias formas e cores, sejam percebidos no campo de atuação. Como detecção de nível de altura, medida de separação, medida de diâmetro em bobinas e detecção de objetos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

2.3.2 Sensor infravermelho

Segundo Moreira (2012), o princípio de funcionamento desses sensores se baseiam na triangulação de raios infravermelhos para medir a distância de um determinado objeto a ele. O funcionamento acontece 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. 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 7, o raio captado pelo PSD pode ser diferente de acordo com a distância do objeto.

Figura 7: Princípio de funcionamento de um sensor infravermelho.

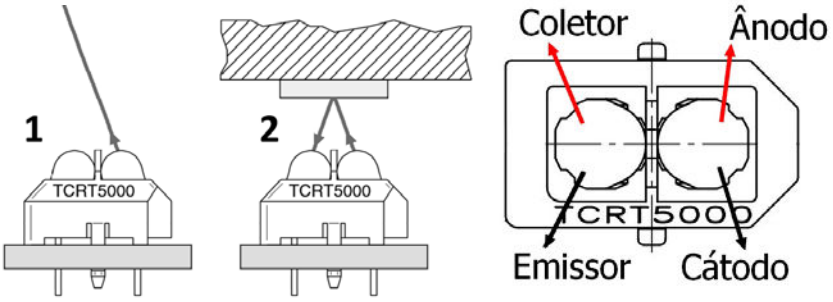


Fonte: Patsko, 2006.

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 como 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 tiver o objeto, menor será a intensidade do sinal.

A figura 8 mostra um exemplo de um sensor infravermelho que obedece ao princípio de funcionamento descrito anteriormente.

Figura 8: Sensor infravermelho.



Fonte: Eletronite http://www.eletronite.com.br/explore/como-utilizar-o-sensor-optico-tcrt5000-no-arduino-uno.html.

2.4 Atuadores

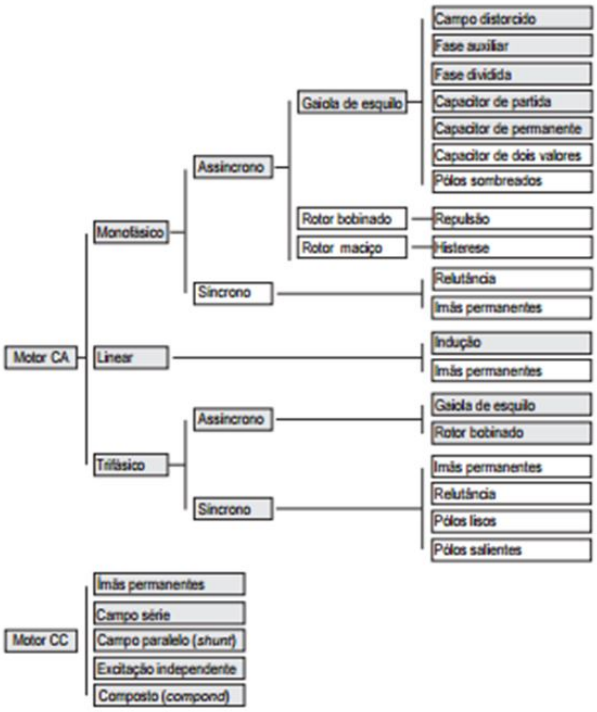
Segundo Wendling (2012), os atuadores são dispositivos que interferem no ambiente, alterando uma variável controlada. Esses dispositivos recebem um sinal gerado pelo controlador, agindo sobre o sistema controlado. Exemplos de alguns atuadores são: válvulas, relés, motores e etc.

2.4.1 Motor elétrico

Os motores elétricos são dispositivos que transformam energia elétrica em energia mecânica, 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 9 esboça a classificação dos motores.

Figura 9: Divisão dos motores elétricos.

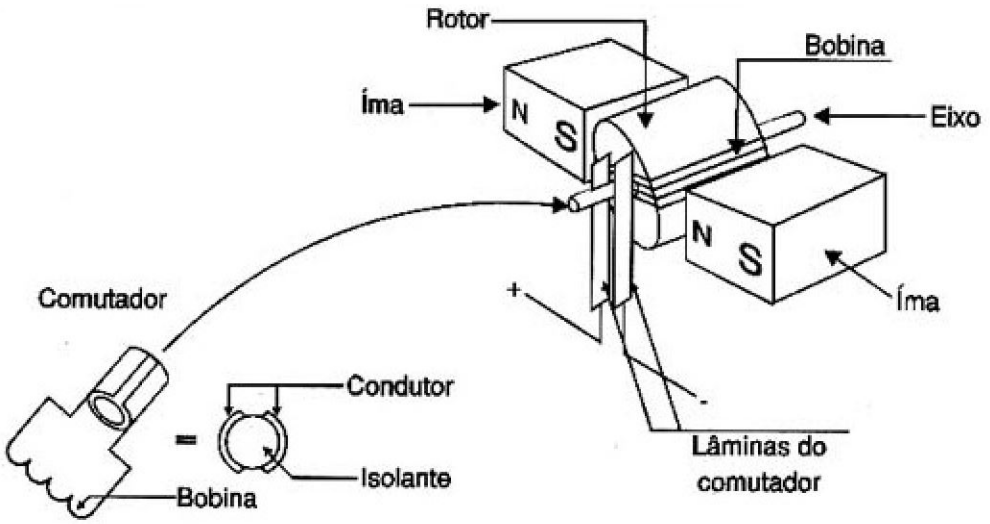


Fonte: Procel, 2009.

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 esses dispositivos são energizados, o motor gira em um determinado sentido e se invertem a polarização da fonte elétrica nestes terminais, o motor gira no sentido oposto.

Figura 10: Motor de corrente contínua.



Fonte: Silva, 2013.

Na Figura 10, pode-se observar que o condutor gera um campo magnético quando nele é aplicada uma corrente, que pode ser repelido ou atraído pelos ímãs permanentes. Os comutadores mudam de contato duas vezes por ciclo, dessa forma invertem, 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 alimentadas 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 dependem dos critérios usados em sua classificação. Podendo ser assíncronos ou síncronos, quando levada em consideração a rotação do motor, e pode ser monofásico ou trifásico quando levada em consideração a tensão elétrica fornecida pela rede de distribuição.

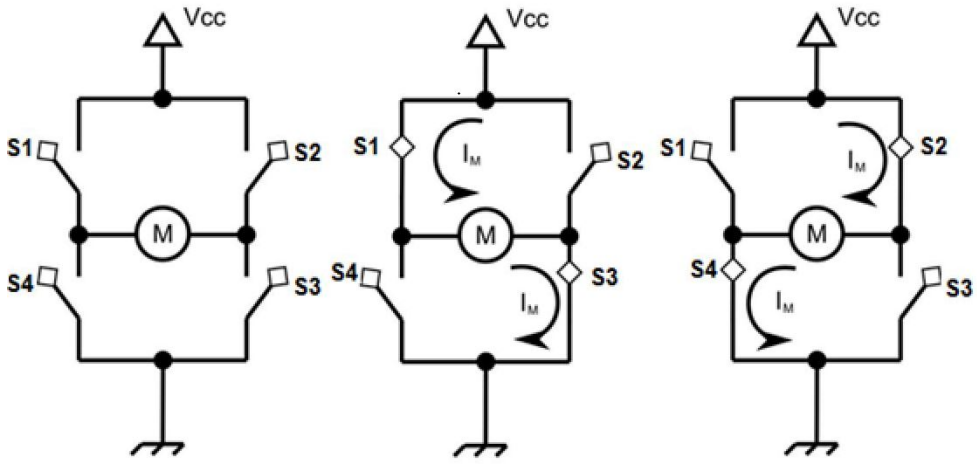
a) Motores Síncronos: o eixo do motor gira com a mesma frequência da rede elétrica, ou seja, com velocidade constante;
b) Motores Assíncronos: são os mais utilizados nas grandes industrias, possuem vantagens como elevada confiabilidade, e desvantagens como baixo rendimento e uma elevada corrente de partida.

2.5 Ponte H

Segundo Gioppo *et al* (2009), existe uma grande necessidade de controlar a rotação dos motores em determinados projetos eletrônicos. 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. 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 11 mostra o formato de uma ponte H, composta por quatro chaves mecânicas ou eletrônicas posicionadas formando uma letra H.

Figura 11: Ponte H.

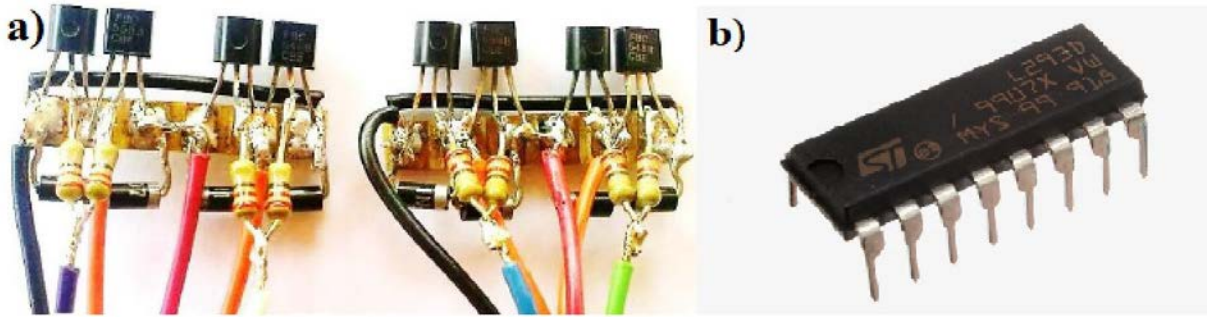


Fonte: AECX - Robótica.

Como pode ser observado na figura 11, 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 se for necessário para o motor, existem duas possibilidades: a primeira desligando todas as chaves, devido não se ter nenhuma corrente no circuito, a segunda possibilidade, é ligando duas chaves superiores ao positivo ou duas chaves inferiores ao negativo da fonte de alimentação, isso gera um freio eletrônico, devido ao fluxo de corrente que passa pela bobina, que faz 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, formato de componentes discretos ou no formato de circuito integrado. Os dois formatos na figura 12 possuem as mesmas funções, porém o que irá diferenciar na sua aplicação.

Figura 12: Tipos de ponte H: a) construída a partir de materiais discretos; b) formato de circuito integrado.



Fonte: a) Moreira, 2012; b) Arduino e Cia, 2014.

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 a velocidade de um motor para que o mesmo execute uma determinada função, como, um giro de um robô móvel.

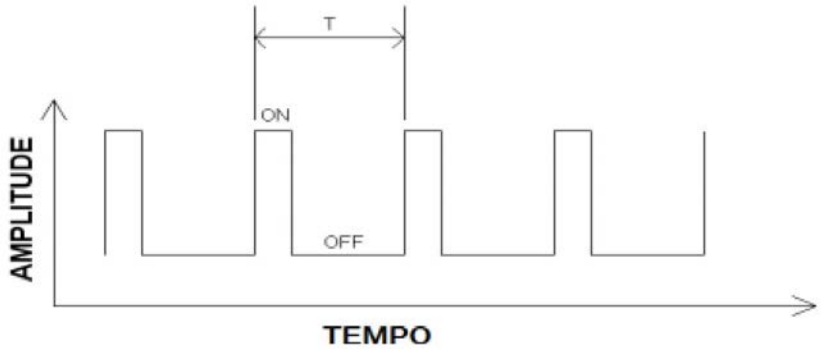
De acordo com Oliveira (2014), PWM é uma forma de codificar digitalmente níveis de sinais analógicos regulando a potência total entregue a uma carga. Essa técnica consiste no equilíbrio dos intervalos de tempo em que a carga recebe a potência máxima e mínima, de tal forma a atingir um valor intermediário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

Ainda segundo Oliveira (2014), pode-se definir o intervalo de tempo em que a carga recebe potência nula no T em OFF, t_{off}, e também pode-se definir o intervalo de tempo em que a carga recebe a potência máxima no T em ON, 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 13.

Figura 13: Forma de onda de um sinal PWM.



Fonte: Oliveira, 2014.

Se por exemplo, uma carga estiver ligada em dois pontos, em que as tensões valem 5 V e 0 V, ela irá receber potência nula se a 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:

$$V_{dc} = \frac{1}{T}\int_{0}^{T} V(t)dt \tag{2.3}$$

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

$$V(t) = \begin{cases} V_{pulso}, & 0 \leq t \leq t_p \\ 0, & t_p < t \leq T \end{cases} \tag{2.4}$$

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

$$V_{dc} = \frac{1}{T}\left(\int_0^{t_p} V_{pulso}dt + \int_{t_p}^{T} 0\ dt\right) \tag{2.5}$$

logo:

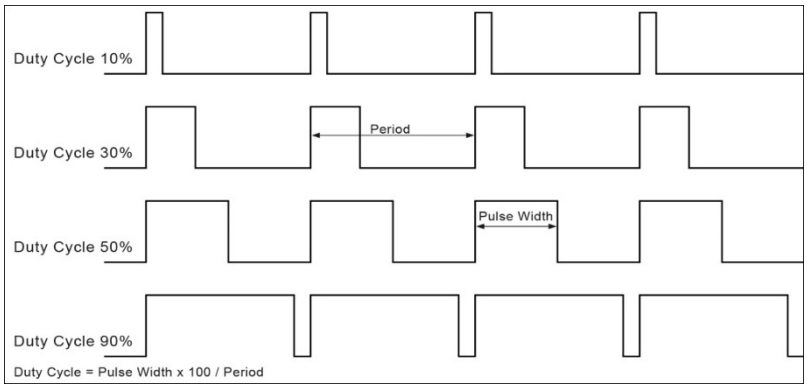
$$V_{dc} = \frac{t_p}{T}V_{pulso} \tag{2.6}$$

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*.

$$duty\ cycle = \frac{t_{on}}{T}\ x\ 100\% \tag{2.7}$$

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 14.

Figura 14: Forma de onda com diferentes Duty Cycle.



Fonte: Protostack, 2016.

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 pulso. 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 ROBÔ DE SUMÔ

Com a evolução tecnológica na área da robótica em geral, nesses últimos anos a competição de robôs se tornou cada vez mais competitiva e popular. Uma dessas competições são os robores de sumôs.

O robô de Sumô é uma competição que segue a linha de regra da luta do sumô, porém com outras regras estabelecidas pelo evento, onde um robô deve colocar o seu adversário, outro robô para fora de uma arena circular, onde essa ação de golpe resultará na pontuação chamado de Yuko.

Portanto este é um novo trabalho dentro da instituição, para a incentivar novos protótipos para este tipo de competição, que está em crescimento. As equipes que participam cada vez mais procuram melhorar seus lutadores, tanto no hardware como no firmware. E com isso surgem e se desenvolvem melhores projetos que podem ser aplicados no cotidiano dos seres humanos.

Neste trabalho haverá uma tentativa de construir um protótipo que estará de acordo com as regras das competições que ocorrem na PUC-RJ.

A competição tem regras para permitir a participação do robô, as principais são:

- O Robô só pode ter no máximo 3 Kg;
- A dimensões do robô é de no máximo de 20 cm de largura por 20 cm de comprimento, sem limitação de altura, até o início da luta, onde poderá se expandir até no máximo 10cm;
- O robô deve ser totalmente autônomo.

O objetivo é ganhar ponto em melhores de 3 rounds colocando o adversário totalmente para fora da arena, sem deixar que o oponente empurre o do competidor para fora da arena. Também se ganha pontos se o adversário sair da arena por motivo próprio.

3.1 Especificação do robô

- O robô deverá caber em um cubo ou quadrado com as dimensões referentes à sua classe;
- A massa total do robô no início da partida deverá ser menor ou igual ao peso designado para sua respectiva classe, de acordo com a tabela 1;

Tabela 1: Categoria de competição

Modalidade	Altura	Comprimento	Largura	Peso
3Kg Sumô	Ilimitado	20 cm	20 cm	3000 g

Fonte: Próprio Autor

- Robôs de 3 kg poderão ser tanto autônomos quanto rádio-controlados. O que está sendo desenvolvido é autônomo;
- Para robôs autônomos, qualquer método de controle poderá ser empregado, desde que esteja contido no robô e que não interaja com um sistema de controle externo (humano ou máquina);
- Os robôs autônomos deverão entrar em operação automaticamente em não menos do que cinco (5) segundos após autorização do juiz e comando dado por um membro da equipe;
- É obrigatória a fixação do nome do robô em uma superfície visível, permitindo que os espectadores e organizadores do evento possam identificar facilmente os robôs envolvidos na partida;
- Os robôs poderão expandir seu tamanho após o início da partida, porém não será permitido se separar fisicamente devendo continuar como um único robô. A violação desta regra implicará na perda da partida. O desprendimento de peças cujo o somatório de suas massas seja inferior a 10g não implicará na perda da partida. Caso um robô seja prejudicado por uma peça que tenha se desprendido de seu adversário, a ele será dado um ponto de Yuko.

3.2 Restrições

- Dispositivos para interferência, porém não limitados a sistemas de LEDs infravermelhos (IR), com intenção de saturar os sensores dos oponentes, não serão permitidos;
- Peças que possam quebrar ou danificar o ringue não serão permitidas. Tais peças serão avaliadas pelos juízes na inspeção de segurança, podendo ou não serem liberadas para o uso. Não utilize peças que tenham, ou não, a intenção de danificar o robô adversário, seu operador e/ou ringue. Impactos e colisões normais não serão considerados como danos intencionais;
- Dispositivos que possam armazenar líquido, pó, gás ou outras substâncias com intenção de lançá-las no oponente não serão permitidos.
- Nenhum dispositivo inflamável será permitido;
- Dispositivos que lancem quaisquer objetos no oponente não serão permitidos;
- Substâncias para melhorar a tração não serão permitidas. Pneus e outros componentes do robô que entrem em contato com a arena não deverão ser capazes de pegar e segurar um cartão de 80 x 130 mm, com gramatura 180gr (cartolina) por mais de 2 (dois) segundos;
- Dispositivos para aumentar a força normal, tais como bombas de vácuo ou imãs, não serão permitidos.

3.3 Dojô

- O interior do Dojô é a superfície onde são realizadas as partidas, circulada por uma linha de borda, que também é parte integrante do Dojô. Qualquer lugar fora dessa área delimitada é chamado de parte exterior;
- O Dojô terá formato circular, com diâmetro de 154,0 cm, espessura de 5,0 mm, o material deverá ser uma placa de aço coberta por poliuretano;

- As Shikiri (linhas de início) consistem em duas linhas marrons (ou equivalentes para absorção de luz infravermelha - IR) centradas no ringue com largura de 2 cm e comprimento de 20 cm. A distância de 20 cm de separação entre as linhas é medida pelos limites externos das mesmas;
- A linha de borda é uma faixa circular com a largura de 5 cm na extremidade externa da superfície de disputa.

3.4 A partida

- A partida será disputada por duas equipes;
- Uma partida consistirá em 3 (três) *rounds*, cada *round* terá um tempo nominal de 1 (um) minuto, podendo ser acrescentado, a critério do juiz, 30 (trinta) segundos totalizando um tempo total de 1 (um) minuto e 30 (trinta) segundos (1’ 30’’). O tempo limite de uma partida não poderá exceder 4 (quatro) minutos e 30 (trinta) segundos;
- A equipe que ganhar 2 (dois) *rounds* ou receber 2 (dois) pontos de Yukô primeiro, dentro do tempo limite, será declarada vencedora. Uma equipe recebe um ponto de Yukô quando vence um *round*. Caso o tempo limite seja atingido antes de uma equipe atingir dois pontos de Yukô e uma das equipes tenha recebido um ponto de Yukô, enquanto a outra não tiver ganhado pontos, esta será a vencedora. Caso a partida não seja vencida por nenhuma equipe dentro do tempo limite, a decisão será realizada pelos juízes, por meio de pontuação seguindo critérios;
- Será permitido ao competidor um tempo máximo de 5 (cinco) minutos para troca de baterias dos robôs entre duas partidas;
- Será permitido ao competidor alterar a programação de seus robôs entre duas partidas, porém, será proibida qualquer tipo de alteração durante a partida, ou seja, entre cada um dos *rounds;*
- *Os rob*ôs deverão ser posicionados tangenciando a borda mais externa da linha de Shikiri. Os dois robôs deverão estar paralelos, porém com as frentes posicionadas em sentidos opostos (180°);

- Quando o juiz principal anunciar o início do *round*, o robô deverá ser ativado (ligar) e após uma pausa de 5 (cinco) segundos os robôs poderão começar a funcionar;
- A partida será paralisada ou retomada conforme os anúncios dos juízes;
- A partida terminará quando anunciado pelo juiz principal. Então as duas equipes recolherão os seus respectivos robôs da área do Dojô.

4 METODOLOGIA

A partir do momento que o projeto foi escolhido, foi pesquisado e estudado de forma que, tivesse o maior custo benefício possível.

Primeiro, foi estudado como seriam a os componentes físicos, escolhendo a princípio qual o atuador que seria trabalhado, que deveria atuar com baixa tensão, mas tendo força para atuar na ação. Assim, foi escolhido o motor fácil acesso, especificamente motor de atuação para vidro elétrico de carro. Após isso, foram escolhidos os sensores seguindo o custo benefício, tendo assim o infravermelho para detecção do dojô e seus limites e o ultrassônico para localização do adversário.

Feitas as escolhas de atuadores e sensores, foram desenvolvido o firmware e o hardware paralelamente.

Para o firmware, foi desenvolvida primeiramente a estratégia que o robô adotaria em ação, repassada para o diagrama de blocos e após isso para um algoritmo. Por fim a programação, nessa última foi utilizado o programa Mikro C PRO for PIC, o mesmo já vem acompanhado com uma placa para simulação ou testes, assim facilitando a programação.

O Hardware foi iniciado pelo dimensionando dos componentes para trabalhar com o sensor infravermelho e também para atuação dos motores. No caso do segundo foi feita simulação no programa Proteus 8. Após as simulações e dimensionamento foram feitas os testes no protoboard. Já para placa principal, do micrcontrolador, já existe um circuito padrão. Para a construção das placas, foi feito o circuito no programa Altium Design, desenhando primeiro o esquemático, para poder depois passar para o PCB, e a partir daí fazer a impressão do circuito e confeccionar nas placas de fenolite, para então montagem das mesmas.

5 HARDWARE

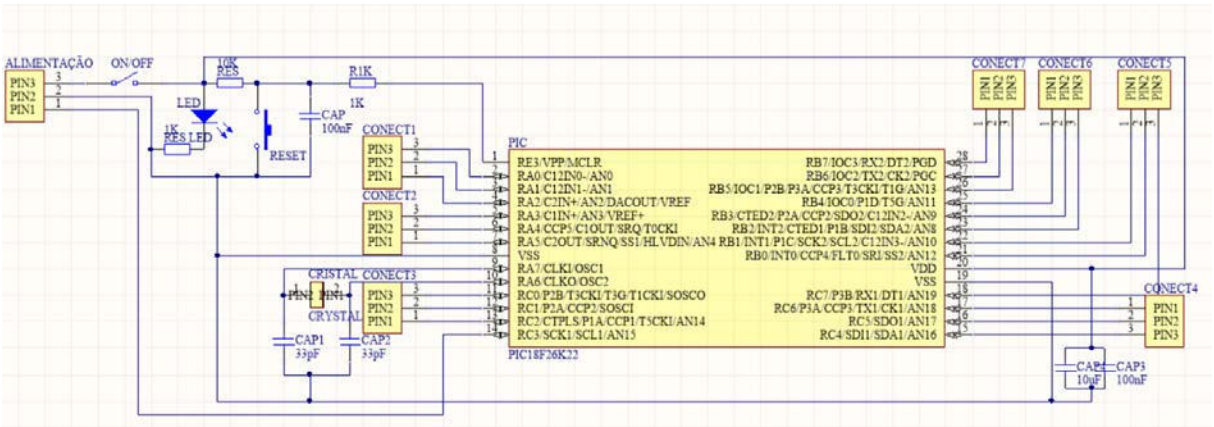
O Hardware é todo conjunto de equipamentos físicos responsáveis pelo processamento e execução das instruções dadas pelo firmware através de seu componente de comando, nesse caso, o microcontrolador.

5.1 Placa principal

A Placa Principal foi desenvolvida para ser responsável por alojar o microcontrolador e seus periféricos principais: a chave ON/OFF, o LED de sinalização de alimentação presente e o botão de reset de programação.

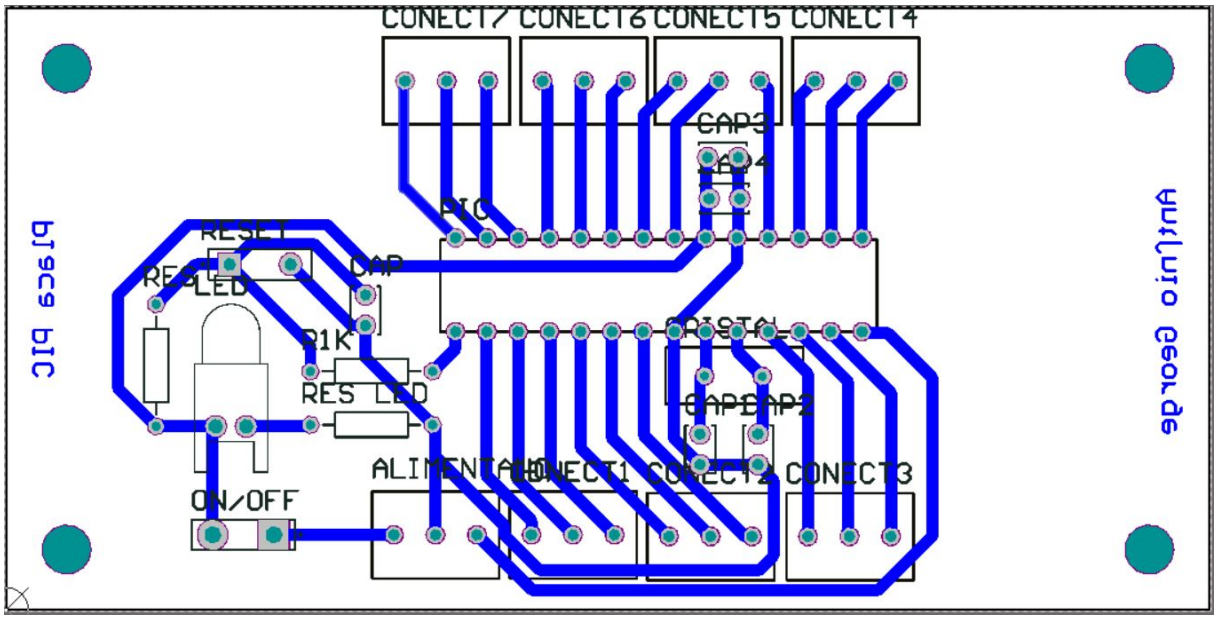
A placa será alimentada pelo conector de alimentação pelo PIN3, passando pela chave ON/OFF (Liga/Desliga). O LED sinalizara que a placa está alimentada, que a mesma está em paralelo com o botão de Reset e seu Capacitor. Os capacitores CAP3 e CAP4 estarão ligados entre o V_{dd}, entrada de alimentação do Microcontrolador, e com o GND representado pelo PIN2 do Conector Alimentação, para filtrar possíveis ruídos ocasionados pelas outras placas. As figuras 15 e 16, ilustram o esquemático e o PCB da placa principal.

Figura 15: Esquema da placa principal.



Fonte: Próprio Autor.

Figura 16: PCB da placa principal.



Fonte: Próprio Autor.

5.1.1 Microcontrolador

São componentes eletrônicos de tamanho reduzido adequados para comandar máquinas e equipamentos eletro-eletrônicos através de firmwares. São componentes que congregam, em um único circuito integrado, distintos componentes de um sistema computacional simples. Em outras palavras, podemos alegar que o microcontrolador é um pequeno microcomputador integrado em um único chip. Por se tratar de um componente programável é bem versátil, podendo ser empregado em aplicações das mais diversas (CORTELETTI 2006)

Microcontrolador é um tipo exclusivo de circuito integrado, pois vem com a probabilidade de ser programado para satisfazer tarefas particulares. Este tipo de circuito integrado contém um núcleo de processador, memória e periféricos programáveis de entrada e saída. Responsável pelo cumprimento das instruções da lógica, firmware, reconhecendo o ambiente através dos sensores e aplicando as ações necessárias através dos atuadores.

É utilizado especialmente na área de automação industrial e em sistema embarcados, por seu custo, tamanho, e também pelo seu consumo de energia relativamente baixo. Normalmente na casa dos miliwatts, possui capacidade para entrar em modo de espera (Sleep ou Wait), esperando por uma interrupção ou evento externo. Em modo de espera pode

aproximar-se na casa dos nanowatts, tornando-os adequados para aplicações onde a necessidade de baixo consumo de energia é um fator determinante para o sucesso do projeto. Esse grande benefício faz com que o microcontrolador hoje seja usado fundamentalmente em eletrônicos, como celulares, televisões, aparelhos de sons e também nos próprios meios de transporte, para poder monitorar o funcionamento dos mesmos. Esta aplicabilidade dos microcontroladores se deve ao fato dele ser mais versátil e funcional, já que em um minúsculo chip dispõe de todos os elementos necessários para fazer uma máquina funcionar (ZELENOVSKY e MENDONÇA, 2010).

Os microcontroladores também estão restritos à sua memória, disposição de processamento e ao número de instruções, pendente do modelo, e ao fim que se destina tal microcontrolador.

Os primeiros microcontroladores necessitavam de componentes externos para funcionar, tornando despesa total do conjunto elevado sendo inviável, economicamente, informatizar dispositivos, o primeiro foi o 4-bit Intel 4004 lançado em 1971, com um tempo, foram criados microcontroladores mais desenvolvidos como o Intel 8008 e outros.

Em 1975, a Intel fabricou um chip (Intel 8048) com memória RAM e ROM introduzidas, o que foi vastamente aproveitado em diversas aplicações. Os microcontroladores tinham duas memórias nomeadamente EPROM, apagável, em um processo dispendioso, e PROM, que podia ser programado somente uma única vez. Em 1993, a memória EEPROM foi embutida nos microcontroladores, ela era eletricamente apagável e tinha um preço de fácil acesso.

O microcontrolador tornou-se conhecido depois que a Intel Corporation lançou uma versão 8-bit, em 1981, chamado de 8051. Intel consentiu para que outros fabricantes fizessem variantes alternativas de 8051, garantindo múltiplas versões do mesmo no mercado. Alguns destes controladores tinham distantes velocidades com múltiplas ROMs dispostas num único chip.

A família de microcontroladores 8051 marcou uma revolução eletrônica com o usuário final a colher os benefícios da tecnologia e da ciência.

Conforme o passar dos anos, os microcontroladores foram crescendo e proporcionando muito mais para os usuários finais e empresas.

5.1.1.1 PIC

O PIC é um microcontrolador fabricado pela Microchip Technology Inc. É um circuito integrado que contém todos os dispositivos imprescindíveis para concretizar um completo sistema digital programável. O PIC pode ser observado como um circuito integrado TTL e CMOS habitual, mas internamente dispõe de todos os dispositivos peculiares necessários para um microprocessador: CPU (é o cérebro do microcontrolador, onde está o "set", conjunto de instruções que o microcontrolador reconhece e sabe executar: onde lê, compara e decide qual instrução executar), memórias ROM (para armazenar permanentemente a lógica da programação) e RAM (para armazenar temporariamente os dados das variáveis), os timers (que são usados para determinam os tempos de execução dos conjuntos de instruções, dependendo do modelo várias famílias de I/O onde faz o interfaceamento do microcontrolador com o mundo exterior).

O PIC contém uma gama de modelo, para melhor adequar em certos tipos exigências especificadas pelo projetista.

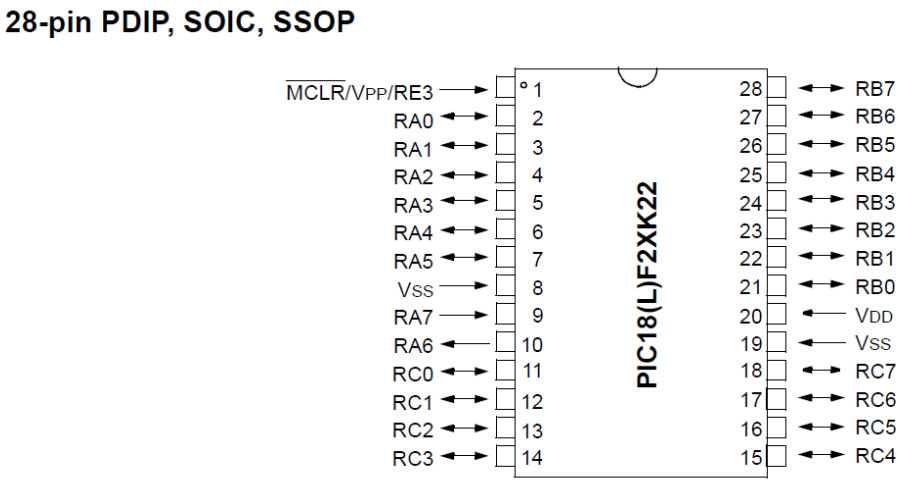
O PIC é um microcontrolador de arquitetura Harvard modificada pela sua fabricante, que processa dados de 8 até 32 bits. É derivado do PIC1650 inicialmente desenvolvido pelo setor de microeletrônica da General Instrument. Seu nome é oriundo de "Programmable Interface Controller" (Controlador de Interface Programável), antes conhecido como "Peripheral Interface Controller" (Controlador de Interface Periférica), atualmente conhecido apenas como PIC. Conta com ampla variedade de modelos e periféricos internos. Possui alta velocidade de processamento devido a sua arquitetura e conjunto de instruções RISC que significa Reduced Instruction Set Computer (Computador Com Conjunto de Instruções Reduzido). Neste tipo de arquitetura, o microcontrolador faz tudo usando poucas instruções básicas que são combinadas de acordo com o que se deseja que ele faça, com recursos de programação por Memória flash, EEPROM e OTP. Os microcontroladores PIC têm famílias com núcleos de processamento de 12 bits, 14 bits e 16 bits e trabalham em velocidades de 0kHz (ou DC) a 48MHz ou velocidades de 16 MIPS em alguns modelos. Há o reconhecimento de interrupções tanto externas como de periféricos internos. Funcionam com tensões de alimentação de 1.8 a 6V e os modelos possuem encapsulamento de 6 a 100 pinos em múltiplos formatos (SOT23, DIP, SOIC, TQFP, etc).

5.1.1.2 O PIC 18F26K22

O Microcontrolador PIC 18F26K22 foi escolhido para ser o cérebro do sistema embarcado por sua facilidade de compra no mercado, pela opção de poucos pinos (28 pinos) e por sua alta velocidade de processamento. Tendo tensão de operação de 5.5V, baixo consumo de corrente, chegando a consumir em modo de suspensão 20nA (nano Amperes), possui 5 saídas PWM, programáveis em 16 níveis. A figura 17 ilustra a estrutura física do PIC 18F26K22. As principais características são:

- Processador de alto desempenho;
- Estrutura do oscilador flexível;
- Conversor analógico-digital;
- Conversor digital-analógico;
- Módulo comparador analógico;
- Módulo de unidade de medição de tempo de carga;
- Gestão de baixa potência extrema em NanoWatt;
- Até 35 pinos de I/O mais um pino de entrada;
- Várias opções de entrada Set/Reset;
- Dois módulos captura / compare / PWM (CCP);
- Três módulos CCP avançados (ECCP);
- Dois transmissores síncronos e receptores assíncronos universal melhorados;
- Entre outras características especiais do microcontrolador.

Figura 17: Forma física PIC18F26K22.



Fonte :Microchip - http://www.microchip.com/wwwproducts/en/PIC18F26K22.

Na tabela 2 serão apresentados os pinos utilizados e que funções foram atribuídas aos mesmos.

Tabela 2: Configuração dos pinos utilizados

Pinos	Funções dos Pinos
RA4	Geração do PWM4 para uma direção de um motor
RB0	Geração do PWM3 para uma direção de um motor
RB4	Leitura de sinal digital do sensor Infravermelho1
RB5	Leitura de sinal digital do sensor Infravermelho2
RB6	Leitura de sinal digital do sensor Infravermelho3
RB7	Leitura de sinal digital do sensor Infravermelho4
RC0	Envio de sinal pulso para o pino *Trigger* do sensor HC-SR04
RC1	Geração do PWM1 para uma direção de um motor
RC2	Leitura do sinal de pulso enviado pelo *Echo* do sensor HC-SR04
RC6	Geração do PWM2 para uma direção de um motor

Fonte: Próprio Autor

5.2 Sensoriamento

O sensor usado para o reconhecimento do robô adversário é o HC-SR04. Ele é utilizado para reconhecimento do ambiente hostil, tanto para encontro de um objeto, quanto para evitar obstáculos, facilitando o traçado de rota do robô a utilizá-lo.

Podendo medir distâncias de 0,02 a 4 metros, com precisão de 0,03 metros, tendo um ângulo de detecção de 15 graus, e utilizando sinais ultrassônicos de 40 KHz. Com alimentação CC de 5 Volts e corrente máxima de 15 mA, o que facilita para projeto de pouco consumo de energia.

Ele é composto por 4 pinos, incluindo o de alimentação (Vcc) e o GND, tendo ainda o *Trigger* e o *Echo*. O funcionamento foi apresentado na secção 2.3.1.

Trigger receberá um sinal com uma duração de 10 us (microssegundo) iniciando a medição, fazendo com que o transmissor automaticamente envie 8 pulsos de 40 Khz. O receptor fará a leitura de sinal de retorno, após encontrado o objeto e éter criado um tempo de

atraso do sinal emitido para o recebido, esse sinal de retorno é enviado pelo pino *Echo*, para o componente responsável pelo cálculo de leitura. A figura 18 representa a imagem do sensor.

Figura 18: Sensor HC-SR04.

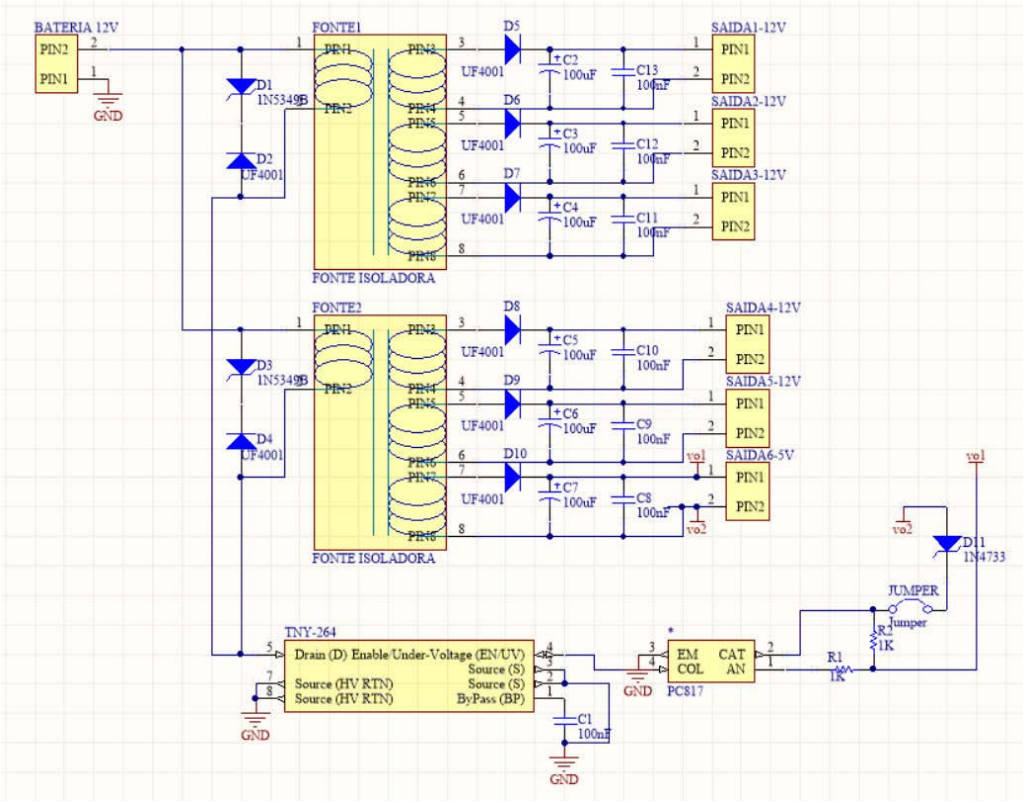
Fonte: http://buildbot.com.br/blog/como-utilizar-o-sensor-ultrasonico-hc-sr04/.

5.3 Placa da fonte

Nesta placa foi desenvolvida uma forma para que alimentação fornecida por uma bateria de 12V fosse distribuída para as outras placas, de forma independente, exceto da bateria, e com a intenção maior que o circuito da placa principal fosse protegido das outras placas, em caso de um curto, em outra placa e não afetasse a da principal.

Esta placa é constituída por duas fontes em toroide (com função de fonte isoladora, cada uma entrada alimentada com 12V) e 3 saídas na mesma tensão (cada saída com diodos com capacitores de filtro para regularizar a saída, sendo que uma das saídas da segunda fonte é abaixadora para 5V, que irá alimentar o microcontrolador). Essa placa é também composta pelo TNY-264 antecipado pelo optoacoplador PC817 para receber a saída de 5V. O TNY-264 terá função de controlar a alimentação de entrada recebendo sinal pela tensão de saída, tentando trazer a melhor eficiência energética para essa transformação. Veremos na figura 19 o esquemático da placa da fonte.

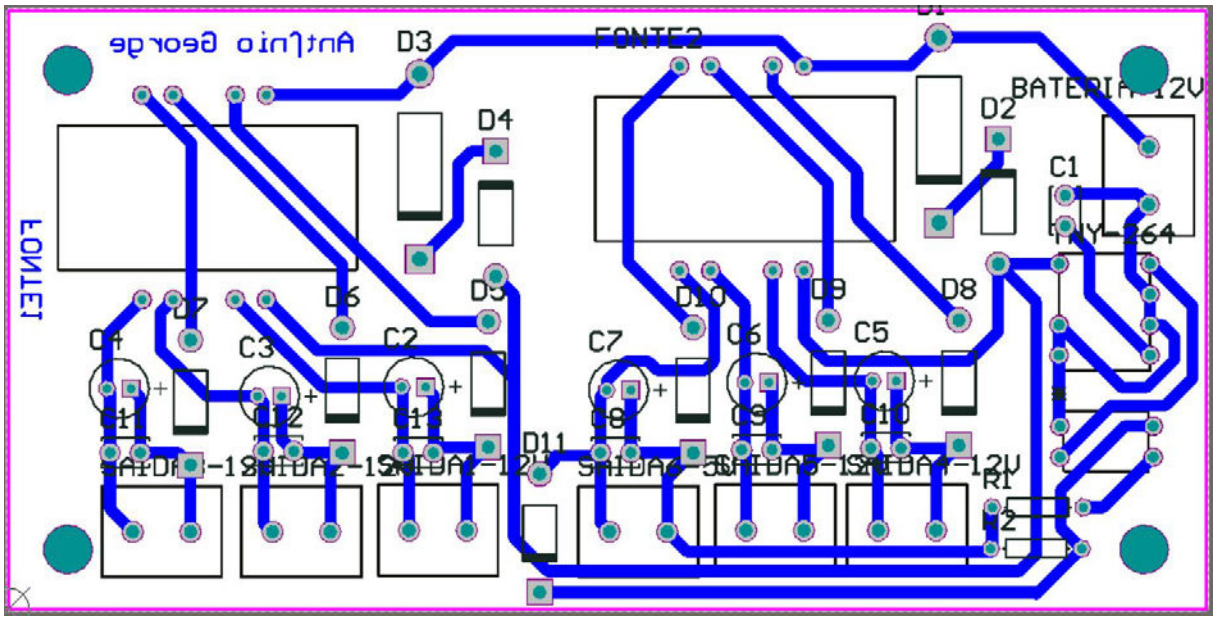
Figura 19: Esquema da placa da fonte.



Fonte: Próprio Autor.

Na figura 20 a seguir, a imagem do PCB da Placa de Fonte.

Figura 20: PCB da placa da fonte.



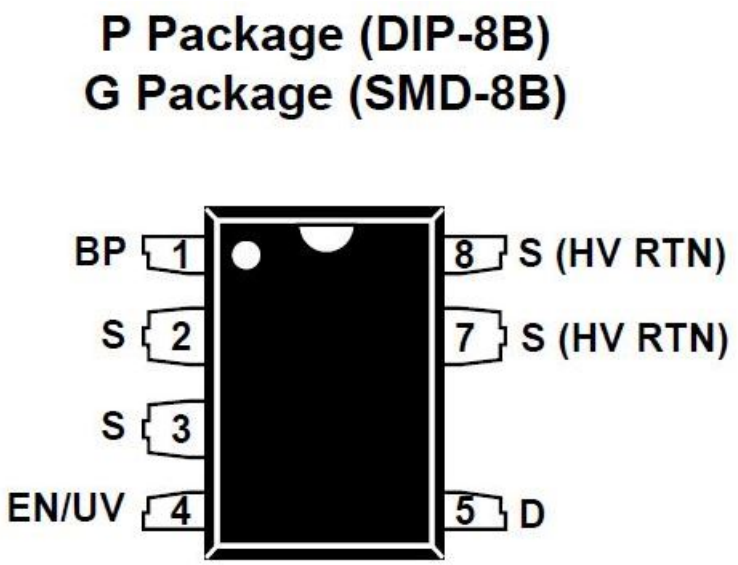
Fonte: Próprio Autor.

5.3.1 Características do TNY-264

O componente tem como função de controlar a saída da fonte a partir da alimentação do mesmo, fazendo uma leitura da saída de 5 volts. Ideal para carregadores de telemóveis, por operar com 132 kHz reduzindo o tamanho do transformador, sem consumo de carga menor que 50 mW com enrolamento de polarização e menor que 250 mW sem enrolamento de polarização com entrada de 265 V_{CA}, elimina praticamente o ruído audível do trasformador normal. Tendo tolerâncias muito apertadas e variações de temperatura insignificantes, auto-reinício totalmente integrado para curto-circuito e Loop para proteção de falhas

A Figura 21 a seguir esboça como é o componete TNY-264 físicamente.

Figura 21: TNY-264 Fisicamente.



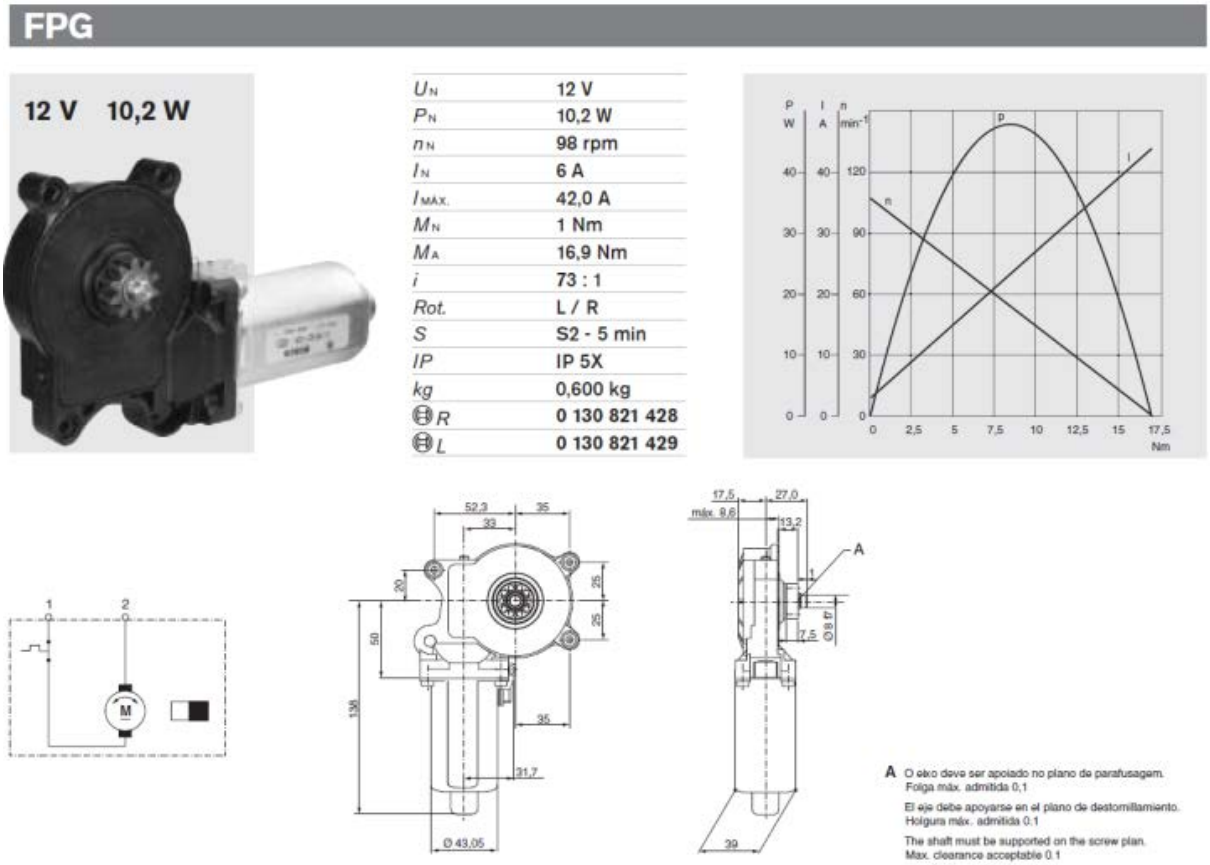
Fonte*: Alldatasheet* - www.alldatasheet.com/Tny264.

5.4 Placa da ponte H

Vimos no capítulo 3 a lógica do funcionamento de uma ponte H, e também o funcionamento do PWM. Nesta placa serão usadas as duas funções, a ponte H para controlar o sentido de rotação dos motores e o PWM para controlar a velocidade do mesmo atuador.

O motor escolhido para utilizarmos foi BOSCH FPG 12V, com corrente nominal de 6A que pode atingir até 42A, como o peso de 600g. Segue a figura 22, ilustrando o motor e seus dados.

Figura 22: Motor BOSCH FPG.

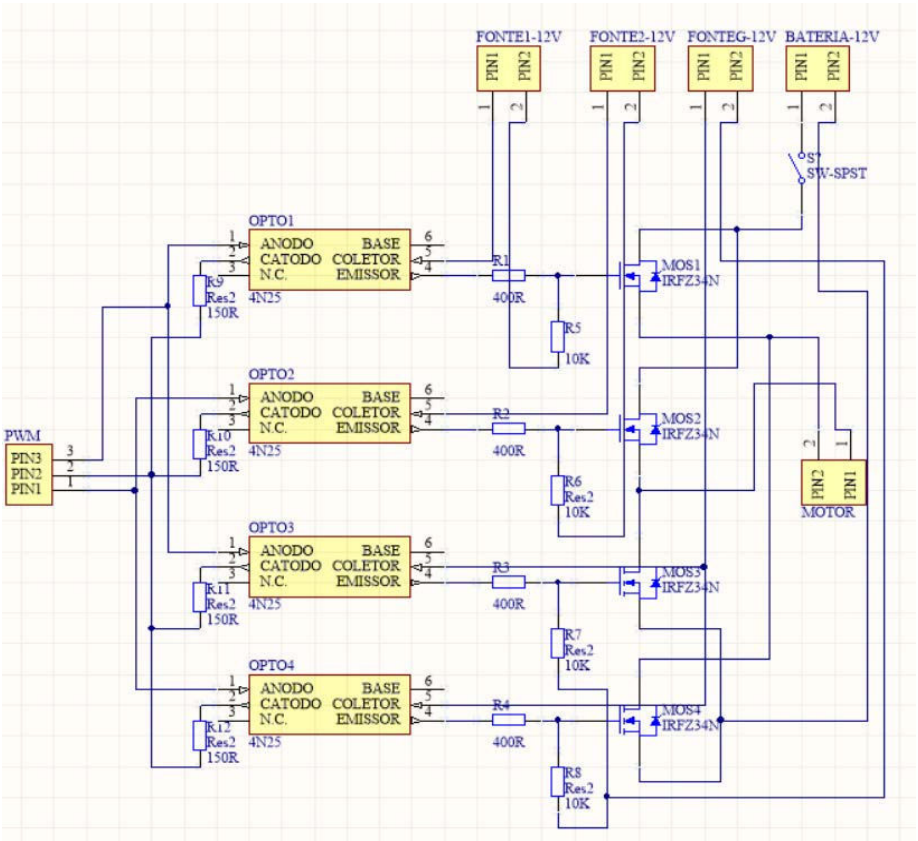


Fonte: BOSCH - http://www.bosch-ibusiness.com/boaaelmoocs-la/category/FPG%20EVO/361.

Para poder suprir comutar na ponte H o acionamento do motor citado, usamos um MOSFET, Transistor de Efeito de Campo de Óxido de Metal Semicondutor. O MOSFET e uma chave ativa com camadas semicondutoras N e P, cujo controle de condução é feito por um terminal isolado chamado de gate. É um semicondutor totalmente controlado, através de uma tensão aplicada entre o gate e o source (V_{GS}). Quando uma tensão V_{GS} adequada é aplicada, o MOSFET entra em condução e conduz correntes positivas. O MOSFET escolhido foi o IRFZ34, sua corrente máxima de trabalho é de 29A para uma tensão V_{GS} de 20V e tendo somente como conduzir corrente a partir de 4V no V_{GS}.

Então a tensão no V_{GS} deverá operar entre 4V a 20V, sendo que se deseja que chegue até 12V, porém o micro só fornece em seus pinos 5V. Para isso, entre o MOSFET e o Microcontrolador foi adicionado para cada MOSFET um Optoacoplador 4N25, onde o pino 1 receberá sinal do PIC (ânodo), o pino 5 (Coletor) receberá 12V, e pelo pino 4 (Emissor) enviará a tensão recebida pelo pino 5 para o Gate do MOSFET correspondente. A Figura 23 esboça o esquema da ponte H.

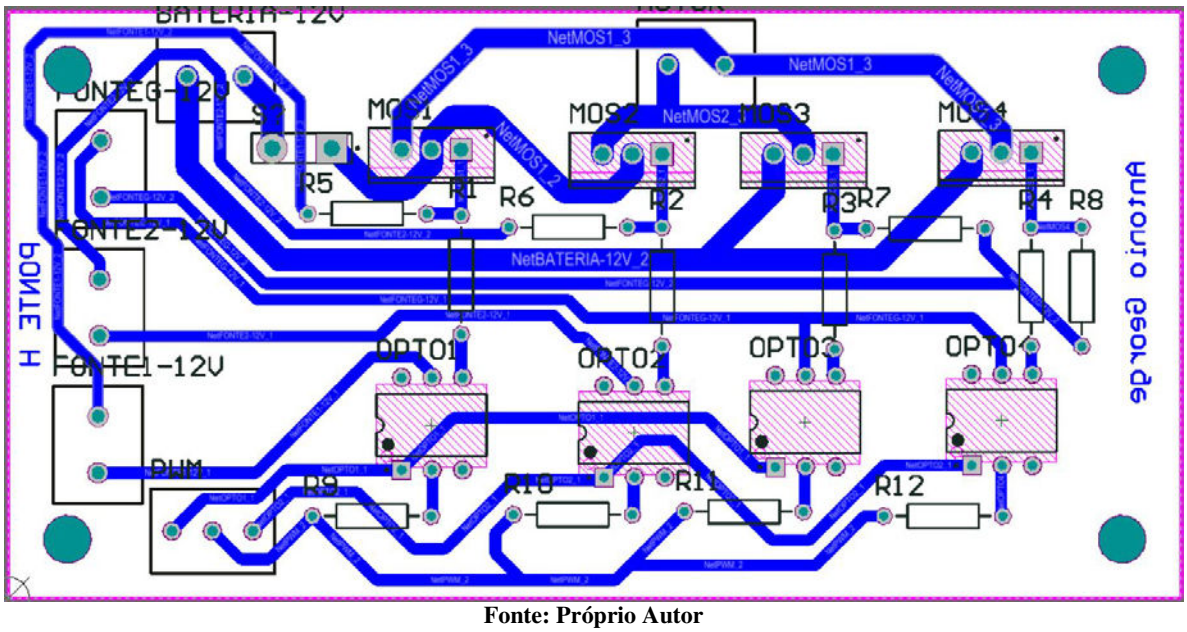
Figura 23: Esquema da placa da ponte-H



Fonte: Próprio Autor

Na figura 24 ilustra o PCB da placa da ponte H.

Figura 24: PCB da placa da ponte-H

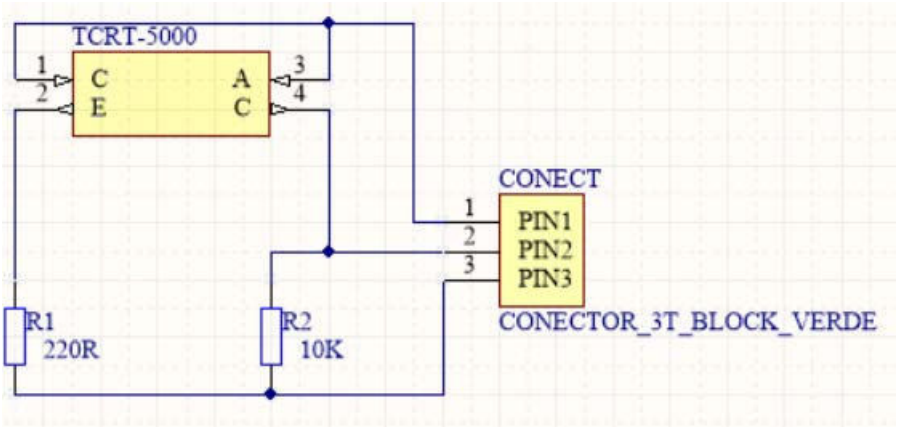


Fonte: Próprio Autor

5.5 Placa do sensor infravermelho

A placa do sensor infravermelho é a placa mais simples, composta somente por resistores para limitar as correntes do circuito para ser usado o sensor TCRT5000 (Figura 25), onde o mesmo é composto por um LED azul e, quando passa por uma faixa branca, o receptor recebe o reflexo da luz emitida pelo LED, emitindo o sinal para saída (KIMENET). A figura 25 representa o esquema do circuito.

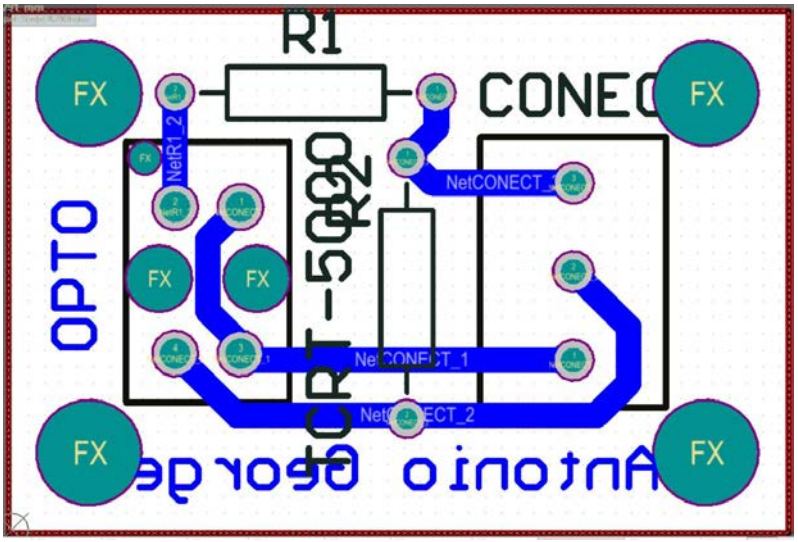
Figura 25: Esquema da placa do sensor infravermelho.



Fonte: Próprio Autor

A figura 26 representa o PCB da placa do circuito infravermelho.

Figura 26: PCB da placa do sensor infravermelho.



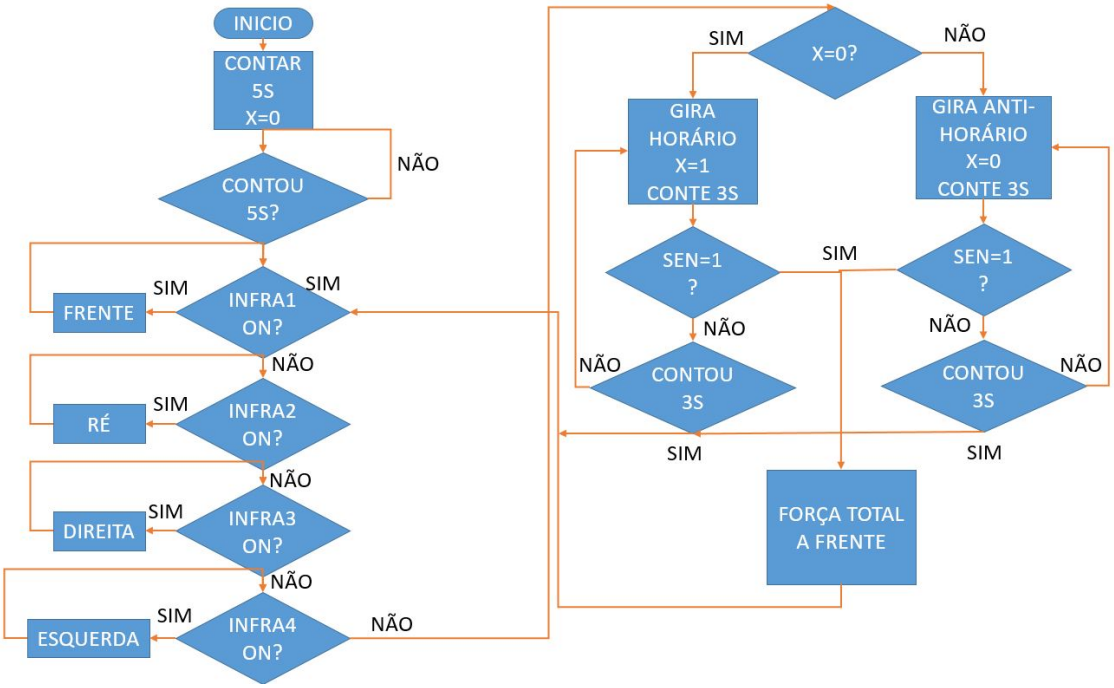
Fonte: Próprio Autor.

6 FIRMWARE

O robô será composto por dois motores, um motor para cada lado, e os mesmos serão controlados por uma ponte H e 2 dois PWM para cada. Assim, a mobilidade do robô se fará por esteiras, tendo também sensores infravermelhos para cada lado do seu formato, como será um paralelograma, terá 4 sensores para detectar as extremidades do tatame, que se dispõe de faixas brancas. Terá também um sensor ultrassônico para localizar o adversário, onde a distância será menor que 140cm, pois o diâmetro do tatame tem 160cm, menos a menor dimensão dos robôs, 10cm para cada competidor, resultando 140cm.

Como para iniciar a partida teremos uma contagem de 5s, só após isso o robô começará sua ação. Primeiramente, ele se certificar de que sua localização não está nas extremidades, para isso deve-se testar primeiros os sensores infravermelhos, se um dos sensores estiverem acionados, a lógica deve fazer com que o robô ande em sentido oposto para correr o risco de sair voluntariamente, ou de facilitar sua saída forçada. Após ter feito os testes nos sensores infravermelhos, a programação fará com que o robô gire até 3 segundos em um certo sentido, se esse tempo for estourado e o sensor ultrassônico não localizar o adversário em 140 cm de diâmetro, ele retornará ao estado de testes dos sensores infravermelhos e começará a girar no sentido oposto ao anterior, isso fará com que ele fique no loop de testes e mudanças de sentido de giro até encontrar o adversário. Encontrando o adversário o robô seguirá em frente com toda velocidade possível até o encontro do mesmo, mas sempre tomando cuidado para não sair do tatame. Vejamos o diagrama de blocos do programa na figura 27.

Figura 27: Diagrama de blocos



Fonte: Próprio Autor

7 RESULTADOS

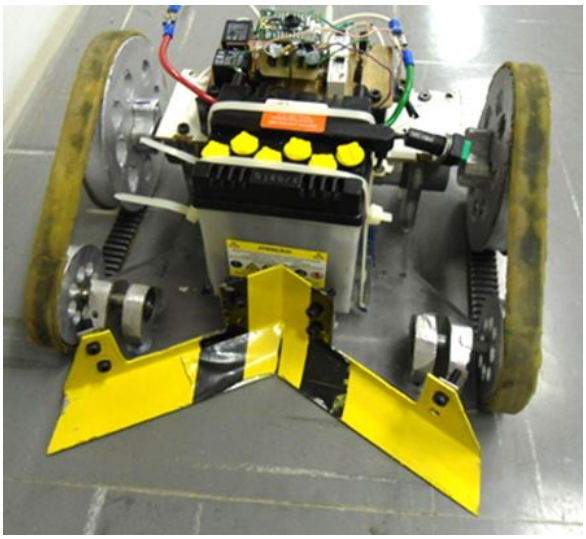
Era esperado para esse tópico resultados 100% positivo, porém, como se trata de um estudo e pesquisa de um protótipo, ele está exposto a falhas, e muitas falhas.

O software teve alguns problemas na configuração dos controles do PWM, mas no fim se obteve sucesso nessa área.

A fonte foi a maior dificuldade, pois o indutor embobinado foi confeccionado manualmente. O TNY-264 não estava disponível a venda no Brasil, tendo que ser comprado por sites do exterior. Na montagem, temos algumas soldas mal feitas e em algumas saídas resultados diferentes do esperado, como não sair nenhuma tensão ou sair tensão maior que o esperado.

A estrutura física foi outra complicação, pois a ideia era um robô que locomove–se com esteira, aumentando o atrito com a superfície. Porém, não foi possível adquirir esteiras para o tamanho desejado. Sendo assim, seria necessário comprar dois pares de esteiras de brinquedo da lego, como a esteira pequena, de uma pá de esteira foi feita só uma. Outro problema foi o dimensionamento do comprimento do robô, pois depende do tamanho do motor, com o tamanho da esteira e o da bateria, não podendo ultrapassar 20x20 cm. A figura 28 é a imagem de um robô que serviu como modelo para estrutura mecânica do protótipo que se pretendia construir.

Figura 28: Modelo para construção do protótipo



Fonte: https://radioitapps.wordpress.com/2013/02/19/projeto-de-robo-vencedor-do-campeonato-unip-jundiai/

8 CONCLUSÃO

A robótica e a automação são áreas novas em forte crescimento, mostrando uma intensidade de desenvolvimento nesses últimos anos, tendo muito a ser descoberto e pesquisado.

A partir dessa ideia, foi idealizado o trabalho nesse ramo, querendo adentrar e aproximar dos conhecimentos que poderiam ser adquiridos ao longo da pesquisa.

O robô de sumô é uma competição que agrega muito conhecimento a prática, fazendo com que tenha uma importância no desenvolvimento dos estudos da área de automação e dando um grande passo à frente na vida acadêmica. Como se trata de um robô autônomo, também acaba aglomerando a área da computação, trabalhando com a programação em microcontrolador e o manipulamento com o mesmo. Ainda fazendo com que o estudante procure sempre melhorias e implementações que permitam novas ideias, para aplicar em sua pesquisa, podendo até revolucionar o mercado da automação

Esse trabalho, além de abrir novos horizontes de projetos para pesquisa acadêmica na Universidade Federal Rural do Semi-Árido, teve como objetivo aproximar mais o aluno da vida profissional, fazendo com que o estudante aplicasse em prática o conhecimento já adquirido.

REFERÊNCIAS

CAMPOS, Camila Ribeiro; QUINTELLA, Camila Mendes; BATISTA ,Mariana Desireé Reale. **Projeto eletrônico para construção de Robô autônomo de sumô.** Disponível em: < http://www.revistas.unifacs.br/index.php/sepa/article/viewFile/305/253>. Acesso em: 08 ago. 2016.

CARVALHO, Juliana Andrade; RICCIO, Juliana Gomes; GAMA, Larissa Baptista. **Projetos mecânico e lógico para a Construção de robô de sumô autônomo.** Disponível em: < www.revistas.unifacs.br/index.php/sepa/article/download/318/267>. Acesso em: 08 ago. 2016

FONSECA, Fabrício Ramos da. **Sensores**. 2006. Disponível em: <http://www.adororobotica.com/Sensores.pdf>. Acesso em: 10 ago. 2016.

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>. Acesso em: 08 ago. 2016.

JESUS, Marlons Alonso Araújo de; MARTINS, Paulo Victor Ribeiro; CAL, Vinícius de Carvalho. **Robótica autônoma – sumô de robôs.** Disponível em: < www.revistas.unifacs.br/index.php/sepa/article/download/307/255>. Acesso em: 15 ago. 2016.

KLOTZ, Darlan; DECKERT, Marcus Eduardo; PAGLIOSA Mauro André. **Desenvolvimento de um robô autônomo para participação em competição de sumô.** Disponível em: < http://eventos.ifc.edu.br/wp-content/uploads/sites/5/2014/09/ENG-81.pdf>. Acesso em: 10 ago. 2016.

NICOLOSI, Denys E. C. **Microcontrolador 8051**: Detalhado. 8.ed. São Paulo: Érica, 2007.

OLIVEIRA, Gustavo C.. **Acionamento de Motores: PWM e Ponte H.** 2014. Disponível em: <https://www.assembla.com/spaces/warthogdc/documents/dclHqyPner46_cacwqEsg8/download/dclHqyPner46_cacwqEsg8>. Acesso em: 20 ago. 2016.

PAREDE, Ismael Moura; GOMES, Luiz Eduardo Lemes. Eletrônica: **Automação Industrial.** 2011. Disponível em: <http://eletro.g12.br/arquivos/materiais/eletronica6.pdf>. Acesso em: 10 ago. 2016.

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>. Acesso em: 06 ago. 2016.

PEREIRA, Fabio. **Microcontroladores PIC Programação em C.** 7.ed. São Paulo: Erica: 2009.

ROBOCORE; **Regras de Sumô.** 2016. Disponível em: < https://www.robocore.net/upload/attachments/robocore__regras_sumo_170.pdf>. Acesso em: 05 ago. 2016.

WENDLING, Marcelo. **Sensores.** 2010. Disponível em: <http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf>. Acesso em: 20 ago. 2016.

APÊNDICE 1

Programação completa em Linguagem C para o Robô de Sumô.

```
*   DESCRICAO: Este programa tem a finalidade de fazer o um robô de sumô. Este robô terá
*   1 sensores ultrassónicos, 2 motores 12Vcc, 4 sensor infra-vermelho.
*     PINOS REFERENTE AO SENSOR ULTRASSONICO
*     RC0 - TRIG DO SENSOR ULTRASSONICO
*     RC2 - ECHO DO SENSOR ULTRASSONICO
*
*     PINOS REFERENETES AOS SENSORES INFRAVERMELHOS
*     RB7 - SENSOR INFRAVERMELHO DA FRENTE
*     RB6 - SENSOR INFRAVERMELHO DE TRÁS
*     RB5 - SENSOR INFRAVERMELHO DO LADO DIREITO
*     RB4 - SENSOR INFRAVERMELHO DO LADO ESQUERDO
*
*     PINOS REFERENTES AO PWM
*       RC1 - PWM DO MOTOR DO LADO DIREITO PARA LOCOMOVER O ROBO PARA FRENTE
*       RA4 - PWM DO MOTOR DO LADO DIREITO PARA LOCOMOVER O ROBO PARA TRÁS
*      RC6 - PWM DO MOTOR DO LADO ESQUERDO PARA LOCOMOVER O ROBO PARA FRENTE
*       RB0 - PWM DO MOTOR DO LADO ESQUERDO PARA LOCOMOVER O ROBO PARA TRÁS
**********************************************************************************/
//**********************************INCLUDES*********************************
#include "Config.h"
#include "Funcoes.h"
//**********************************DEFINICOES*******************************
#define OLD 0
#define NEW 1
//*****************Conexão com os pinos do modulo LCD ************************
```

```
/*
sbit LCD_RS at LATC3_bit;
sbit LCD_EN at LATC1_bit;
sbit LCD_D4 at LATC7_bit;
sbit LCD_D5 at LATC6_bit;
sbit LCD_D6 at LATC5_bit;
sbit LCD_D7 at LATC4_bit;

sbit LCD_RS_Direction at TRISC3_bit;
sbit LCD_EN_Direction at TRISC1_bit;
sbit LCD_D4_Direction at TRISC7_bit;
sbit LCD_D5_Direction at TRISC6_bit;
sbit LCD_D6_Direction at TRISC5_bit;
sbit LCD_D7_Direction at TRISC4_bit;
*/
/*
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB0_bit;
sbit LCD_D5 at LATB1_bit;
sbit LCD_D6 at LATB2_bit;
sbit LCD_D7 at LATB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
*/
//**********************DECLARACAO DE VARIAVEIS ************************
//variaveis referente ao modulo ccp1
```

```
union Data1{
  unsigned char timer1_8bits[2];
  unsigned int timer1_16bits;
};

volatile saindo_do_circulo = 0;
volatile union Data1 Valor1;
volatile unsigned int VarOld1;
volatile unsigned int VarNew1;
volatile unsigned char _status1 = OLD;

volatile unsigned char temp;

extern unsigned char robo_andando_frente = 0;
extern unsigned char robo_andando_esquerda = 0;
extern unsigned char robo_andando_direita = 0;
extern unsigned char robo_andando_tras = 0;

//*****************************FUNCAO PRINCIPAL*************************
void main(){
//variaveis da funcao principal---------------------------------
//CCP1
    unsigned int largura1;
    unsigned int distancia1_cm;
    char txtData1[6];
//-------------------------------------------------------------
    ConfigEntradas();
    ConfigSaidas();          //configura saidas
    ConfigInterrupcao();
    Config_CCP1();      //configura modulo CCP1 para Capture
    ConfigChangeRB();
    ConfigMCU();
```

```
ConfigTimer1();     //configura o timer 1 para funcionar com o modulo CCP1

//CCP2 timer 2
PWMInit_RC1_DIR_FRE(20000);         // Initialize PWM module at 20KHz no pino RC1
PWMduty_RC1_DIR_FRE(50);
//PWMStart_RC1_DIR_FRE();                   // start PWM no pino RC1
//CCP3 timer 4
PWMInit_RC6_ESQ_FRE(20000);         // Initialize PWM module at 20KHz no pino RC6
PWMduty_RC6_ESQ_FRE(50);
//PWMStart_RC6_ESQ_FRE();                   // start PWM no pino RC6
//CCP4  timer 6
PWMInit_RB0_ESQ_TRAS(20000);       // Initialize PWM module at 20KHz no pino RB0
PWMduty_RB0_ESQ_TRAS(50);
//PWMStart_RB0_ESQ_TRAS();                  // start PWM no pino RB0
//CCP5  timer 6
PWMInit_RA4_DIR_TRAS(20000);       // Initialize PWM module at 20KHz no pino RA4
PWMduty_RA4_DIR_TRAS(50);
//PWMStart_RA4_DIR_TRAS();                  // start PWM no pino RA4

/*
//Inicializacao do display LCD
Lcd_Init();                     // Initialize LCD
Lcd_Cmd(_LCD_CLEAR);              // Clear display
Lcd_Cmd(_LCD_CURSOR_OFF);         // Cursor off
Lcd_Out(1,1, "   Aguarde!!!   ");
Delay_ms(2000);
Lcd_Out(1,1, "                ");
Lcd_Out(2,1, "                ");
*/
 //delay_ms(5000);
//loop infinito
while(1){
```

```
//CCP1
if (PIE1.CCP1IE == 0){
  //asm CLRWDT;     //zera flag watch dog
  largura1 = VarNew1 - VarOld1;
  //WordToStr(largura1,txtData1);
  //Lcd_Out(2,7, txtData1);
  distancia1_cm = largura1/58;
  //WordToStr(distancia1_cm,txtData1);
  //Lcd_Out(1,1, txtData1);

  PIE1.CCP1IE = 1;  //Liga novamente o modulo CCP1 para fazer o cálculo de uma nova largura de pulso
  trigger1();
}
//Encontrando o robo adversario, mover robo para frente
if (distancia1_cm <= 140 && distancia1_cm >= 0 && robo_andando_frente == 0){
   Parar();
   delay_ms(500);
   Frente();
}
//ENQUATO NÃO FOR ENCONTRADO O ROBO ADVERSÁRIO, GIRAR ROBO PARA ESQUERDA
else if (distancia1_cm > 140 && robo_andando_esquerda == 0){
   Parar();
   delay_ms(500);
   Esquerda();
}
//Quando algum sensor infravermelho está saindo do círculo, executa comando abaixo
if (saindo_do_circulo == 1){
   Parar();
   Delay_ms(500);
   saindo_do_circulo = 0;
```

```
        if (RB7_bit == 1){
            while (RB7_bit == 1){
                TRAS();
            }
        }
        if (RB6_bit == 1){
            while (RB6_bit == 1){
                Frente();
            }
        }
        if (RB5_bit == 1){
            while (RB5_bit == 1){
                Esquerda();
            }
            Parar();
            Delay_ms(500);
            Frente();
            Delay_ms(1000);
        }
        if (RB4_bit == 1){
            while (RB4_bit == 1){
                Direita();
            }
            Parar();
            Delay_ms(500);
            Frente();
            Delay_ms(1000);
        }
    }

}
```

```
}
// ************************FIM DA FUNCAO PRINCIPAL**********************
//***********ENDERECO DE INTERRUPCAO DE ALTA PRIORIDADE************
void INTERRUPCAO_HIGH() iv 0x0008 ics ICS_AUTO{
    //Interrupção por mudança de estado nos pinos RB7 a RB4
    if (INTCON.RBIE == 1 && INTCON.RBIF == 1){
        LED ^= 1;       //inverte o estado lógico do led
        saindo_do_circulo = 1;
        //técnica usada para corrigir o problema com o projeto do circuito RB
        //ler referencias 1 e 2
        temp = PORTB;   //lê o valor do portb para resgatar o valor do estado real do portb
        PORTB = temp; //escreve o valor novamente para carrega-lo no LATB, pois é pelo
valor
            //do trinco garantido pelo LAT que é feito internamente a operação XOR pelo MCU.
        INTCON.RBIF = 0;
    }
}
//************ENDERECO DE INTERRUPCAO DE BAIXA PRIORIDADE***********
void INTERRUPCAO_LOW() iv 0x0018 ics ICS_AUTO {
//************************Sensor ultrassônico frontal*****************************
   if (PIR1.CCP1IF == 1 && PIE1.CCP1IE == 1){
 //tempo T1
      if (_status1 == OLD){
                          //CCPR1L = TMR1L; E CCPR1H = TMR1H;
        Valor1.timer1_8bits[0] = CCPR1L;     //Salva os 8 bits menos significativos
        Valor1.timer1_8bits[1] = CCPR1H;     //Salva os 8 bits mais significativos
        VarOld1 = Valor1.timer1_16bits;      //Salva o Valor do tempo T1

        CCP1CON = 0;   //Recomendação do fabricante
        CCP1CON = 0b00000100;   //Configura captura na borda de descida
        _status1 = NEW;
      }
```

```
      //tempo T2
      else if(_status1 == NEW){

        Valor1.timer1_8bits[0] = CCPR1L;     //Salva os 8 bits menos significativos
        Valor1.timer1_8bits[1] = CCPR1H;     //Salva os 8 bits mais significativos
        VarNew1 = Valor1.timer1_16bits;      //Salva o Valor do tempo T1

        CCP1CON = 0;   //Recomendação do fabricante
        CCP1CON = 0b00000101;   //Configura captura na borda de subida
        _status1 = OLD;
        PIE1.CCP1IE = 0;       //Desliga o modulo CCP1 para fazer o cálculo da largura do pulso
      }
      PIR1.CCP1IF = 0;    //zera flag da interrupção do modulo ccp1
    }
}
```

CONFIGURAÇÃO DA ENTRADA E SAÍDAS E DO PWM

```
#include "Config.h"

//**********************CONFIGURACAO DO MCU*****************************
void ConfigMCU(){
    ANSELA = 0;
    ANSELC = 0;      //PORTC configurado como digitais
    ANSELB = 0;
}
//**************CONFIGURA AS CHAVES DE INTERRUPCAO*******************
void ConfigInterrupcao(){
    INTCON.GIEH = 1;      //habilita todas as interrupções
    INTCON.GIEL = 1;      //habilita todas as interrupções
```

```
    RCON.IPEN = 1;        //Habilita nível de prioridade de interrupção
}
//*****************************ENTRADAS*********************************
void ConfigEntradas(){
    eco1_direction = 1;
    infravermelho1_direction = 1;
    infravermelho2_direction = 1;
    infravermelho3_direction = 1;
    infravermelho4_direction = 1;
}
//*****************************SAIDAS***********************************
void ConfigSaidas(){
    trig1_direction = 0;
    LED_direction = 0;
    LED = 0;
    LATC1_bit = 0;
    pwm1_direction = 0;
    LATC6_bit = 0;
    pwm2_direction = 0;
    LATB0_bit = 0;
    pwm3_direction = 0;
    LATA4_bit = 0;
    pwm4_direction = 0;
}
//************************Configura modulo CCP1*************************
void Config_CCP1(){
    PIE1.CCP1IE = 0;     //Desabilita a interrupção do modulo CCP1
    IPR1.CCP1IP = 0;     //Define a interrupção do modulo CCP1 como baixa prioridade
    PIR1.CCP1IF = 0;
    CCPTMRS0 = 0b10001000;   //Configura os timers para operar em cada modulo CCP
                        //timer 1 -> CCP1  bits 1-0 do registrador CCPTMRS0
                        //timer 3 -> CCP2  bits 4-3 do registrador CCPTMRS0
```

```
                    //timer 5 -> CCP3  bits 7-6 do registrador CCPTMRS0
                    // CCPxCON
    CCP1CON = 0;  //recomendação do fabricante, sempre que for alterar esse registrador, primeiramente
                    //zera ele e depois altera para evitar falsas interrupções
    CCP1CON = 0b00000101;  //Configura captura na borda de subida
}
//************************Configura timer 1**********************************
//ciclo de maquina
//Fosc = 16MHz  --> 16/4 = 4MHz
//Ciclo de maquina = 0.25us
//Prescale Timer1 1:4
//logo -> 0.25us * 4 = 1us
void ConfigTimer1(){

    T1CON = 0b00100010;  //configurando timer de 16 bit, prescale 1:4, Fosc/4,
    //Zera o timer
    TMR1H = 0;
    TMR1L = 0;
    T1CON.TMR1ON = 1;    //Liga o timer 1
}
//************************Change PORTB<7:4>*********************************
//Configura a interrupção por mudança de estado
void ConfigChangeRB(){

    INTCON.RBIE = 1;  //Habilita a interrupção por mudança de estado
    INTCON.RBIF  = 0;  //Reseta flag de interrupção
    INTCON2.RBIP = 1;  //Configura interrupção com alta prioridade
    INTCON2.RBPU = 0;  //Habilita chave geral dos resistores de pull-up
    WPUB.WPUB4 = 1;   //Habilita resistor de pull-up no RB4
    WPUB.WPUB5 = 1;   //Habilita resistor de pull-up no RB5
    WPUB.WPUB6 = 1;   //Habilita resistor de pull-up no RB6
```

```
    WPUB.WPUB7 = 1;    //Habilita resistor de pull-up no RB7
    //Configuração individual da interrupção por mudança de estado no RB
    IOCB.IOCB4 = 1;    //chave individual de controle da interrupção no pino RB4 apenas
    IOCB.IOCB5 = 1;    //chave individual de controle da interrupção no pino RB5 apenas
    IOCB.IOCB6 = 1;    //chave individual de controle da interrupção no pino RB6 apenas
    IOCB.IOCB7 = 1;    //chave individual de controle da interrupção no pino RB7 apenas
}

//********************CONFIGURAÇÃO DO PWM CCP2**********************
//Esta função calcula o Período do PWM com base na Freq enviado como parâmetro
// Cristal usado: Fosc = 16MHz
//1° Etapa: Cálculo do Período
//Fórmula:
//PR2 = ( ((Fosc)/(4 * FPWM * PRESCALER)) -1)
//PR2 = (16.000.000Hz / (4 * 20.000Hz * 4) ) - 1
//PR2 = 49
void PWMInit_RC1_DIR_FRE(unsigned int Freq){
  float Aux, Tosc;
  Tosc = (1./(Clock_kHz()*1000));       //Equivale a: 1./16000Hz*1000
                                        //Clock_KHz é uma função inline do compilador mikroC
                                        //no qual informa o valor do oscilador Fosc configurado no painel
                                        //edit Project

  Aux = 1./Freq;          //Encontra o período com base na frequência carregada no parâmetro
  PR2 = ceil((Aux/(4 * Tosc * 4)) - 1); //Encontra o valor de PR2.
                                        //A função ceil arredonda o resultado. Habilitar biblioteca C_Math
                                        //tomar cuidado que o resultado deverá estar entre 0 a 255
  CCPTMRS0.C2TSEL1 = 0;
  CCPTMRS0.C2TSEL0 = 0;           //Configurando o timer 2 para o pwm
  T2CON = 0b00000001;             //Desliga TIMER2 e carrega prescaler 1:4
  CCP2CON = 0b00001100;       //Configura CCP no modo PWM. A partir desse momento o
PWM já está ligado.
```

```
}

void PWMduty_RC1_DIR_FRE(unsigned char Duty){

  int VarAux = 0;
  VarAux = 1 * 4 * (PR2+1);      //Com base no PR2 já carregado, encontra valor máximo de contagem
                                 //de TMR2, o que afeta diretamente na contagem do duty cycle
  VarAux =  (VarAux * (Duty/100.));        //Regra de 3x simples, no qual converte o valor máximo encontrado
                                 //em 0 a 100%
                                 //Ex: Se o parâmetro da função Duty = 92
                                 //VarAux =  (VarAux * (92/100.));
                                 //VarAux passa a ser igual a 92% de seu valor original, ou seja
                                 //Duty agora é de 92% do seu máximo valor
  //Carrega os dois LSB encontrado em VarAux nos bits CCP2CON<5:4>
  CCP2CON.B4 = ((VarAux & 0x01) == 0x01); //Mascara para setar/Resetar somente CCP2CON.B4
  CCP2CON.B5 = ((VarAux & 0x02) == 0x02); //idem para B5

  VarAux = VarAux >>2;  //Desloca para a direita o valor do Duty para carregar em CCPR2L
                 //pois já carregamos os LSBs nas duas linhas acima.
  CCPR2L = VarAux;      //Salva Duty em CCPR2L
  //VarAux = 0b0000111011
  //CCP1CON<5:4> = 11
  //VarAux = 0b0000111011 >> 2  =0b0000001110
  //CCPR1L = 0B00001110;
}

//Start na geração do sinal PWM em CCP2
void PWMStart_RC1_DIR_FRE(){
 T2CON.TMR2ON = 1;  //Liga TIMER e inicia a PWM
}
```

```
//Stop na geração do sinal PWM em CCP2
void PWMStop_RC1_DIR_FRE(){
  T2CON.TMR2ON = 0;  //desliga TIMER e inicia a P
}
//********************CONFIGURAÇÃO DO PWM CCP3************************
//Esta função calcula o Período do PWM com base na Freq enviado como parâmetro
// Cristal usado: Fosc = 16MHz
//
//1° Etapa: Cálculo do Período
//Fórmula:
//PR2 = ( ((Fosc)/(4 * FPWM * PRESCALER)) -1)
//PR2 = (16.000.000Hz / (4 * 20.000Hz * 4) ) - 1
//PR2 = 49
void PWMInit_RC6_ESQ_FRE(unsigned int Freq1){

  float Aux1, Tosc1;
  Tosc1 = (1./(Clock_kHz()*1000));       //Equivale a: 1./16000Hz*1000
                          //Clock_KHz é uma função inline do compilador mikroC
                          //no qual informa o valor do oscilador Fosc configurado no painel
                          //edit Project

  Aux1 = 1./Freq1;                    //Encontra o período com base na frequência carregada no
parâmentro
  PR4 = ceil((Aux1/(4 * Tosc1 * 4)) - 1); //Encontra o valor de PR2.
                          //A função ceil arredonda o resultado. Habilitar biblioteca C_Math
                          //tomar cuidado que o resultado deverá estar entre 0 a 255
  CCPTMRS0.C3TSEL1 = 0;
  CCPTMRS0.C3TSEL0 = 1;          //Configurando o timer 4 para o pwm no pino RC6

  T4CON = 0b00000001;               //Desliga TIMER2 e carrega prescaler 1:4
  CCP3CON = 0b00001100;                  //Configura CCP no modo PWM. A partir desse
momento o PWM já está ligado.
```

```
}

void PWMduty_RC6_ESQ_FRE(unsigned char Duty1){

  int VarAux1 = 0;
  VarAux1 = 1 * 4 * (PR4+1);                  //Com base no PR2 já carregado, encontra valor máximo de contagem
                                              //de TMR2, o que afeta diretamente na contagem do duty cycle
  VarAux1 =  (VarAux1 * (Duty1/100.));        //Regra de 3x simples, no qual converte o valor máximo encontrado
                                              //em 0 a 100%
                                              //Ex: Se o parâmetro da função Duty = 92
                                              //VarAux =  (VarAux * (92/100.));
                                              //VarAux passa a ser igual a 92% de seu valor original, ou seja
                                              //Duty agora é de 92% do seu máximo valor
  //Carrega os dois LSB encontrado em VarAux nos bits CCP2CON<5:4>
  CCP3CON.B4 = ((VarAux1 & 0x01) == 0x01); //Mascara para setar/Resetar somente CCP2CON.B4
  CCP3CON.B5 = ((VarAux1 & 0x02) == 0x02); //idem para B5

  VarAux1 = VarAux1 >>2;  //Desloca para a direita o valor do Duty para carregar em CCPR2L
                          //pois já carregamos os LSBs nas duas linhas acima.
  CCPR3L = VarAux1;      //Salva Duty em CCPR2L
  //VarAux = 0b0000111011
  //CCP1CON<5:4> = 11
  //VarAux = 0b0000111011 >> 2  =0b0000001110
  //CCPR1L = 0B00001110;
}

//Start na geração do sinal PWM em CCP2
void PWMStart_RC6_ESQ_FRE(){
  T4CON.TMR4ON = 1;  //Liga TIMER e inicia a PWM
```

```
}

//Stop na geração do sinal PWM em CCP2
void PWMStop_RC6_ESQ_FRE(){
  T4CON.TMR4ON = 0;  //desliga TIMER e inicia a P
}
//**********************CONFIGURAÇÃO DO PWM CCP4**********************
//Esta função calcula o Período do PWM com base na Freq enviado como parâmetro
// Cristal usado: Fosc = 16MHz
//
//1° Etapa: Cálculo do Período
//Fórmula:
//PR2 = ( ((Fosc)/(4 * FPWM * PRESCALER)) -1)
//PR2 = (16.000.000Hz / (4 * 20.000Hz * 4) ) - 1
//PR2 = 49
void PWMInit_RB0_ESQ_TRAS(unsigned int Freq2){

  float Aux2, Tosc2;
  Tosc2 = (1./(Clock_kHz()*1000));       //Equivale a: 1./16000Hz*1000
                                 //Clock_KHz é uma função inline do compilador mikroC
                                 //no qual informa o valor do oscilador Fosc configurado no painel
                                 //edit Project

  Aux2 = 1./Freq2;                       //Encontra o período com base na frequência carregada no
parâmentro
  PR6 = ceil((Aux2/(4 * Tosc2 * 4)) - 1); //Encontra o valor de PR2.
                                 //A função ceil arredonda o resultado. Habilitar biblioteca C_Math
                                 //tomar cuidado que o resultado deverá estar entre 0 a 255
  CCPTMRS1.C4TSEL1 = 1;
  CCPTMRS1.C4TSEL0 = 0;           //Configurando o timer 6 para o pwm no pino RC6

  T6CON = 0b00000001;                //Desliga TIMER6 e carrega prescaler 1:4
```

```
 CCP4CON = 0b00001100;                  //Configura CCP no modo PWM. Apartir desse momento o PWM já está ligado.
}

void PWMduty_RB0_ESQ_TRAS(unsigned char Duty2){

  int VarAux2 = 0;
  VarAux2 = 1 * 4 * (PR6+1);             //Com base no PR2 já carregado, encontra valor máximo de contagem
                                         //de TMR2, o que afeta diretamente na contagem do duty cycle
  VarAux2 =  (VarAux2 * (Duty2/100.));   //Regra de 3x simples, no qual converte o valor máximo encontrado
                                         //em 0 a 100%
                                         //Ex: Se o parâmetro da função Duty = 92
                                         //VarAux =  (VarAux * (92/100.));
                                         //VarAux passa a ser igual a 92% de seu valor original, ou seja
                                         //Duty agora é de 92% do seu máximo valor
  //Carrega os dois LSB encontrado em VarAux nos bits CCP2CON<5:4>
  CCP4CON.B4 = ((VarAux2 & 0x01) == 0x01); //Mascara para setar/Resetar somente CCP2CON.B4
  CCP4CON.B5 = ((VarAux2 & 0x02) == 0x02); //idem para B5

  VarAux2 = VarAux2 >>2;  //Desloca para a direita o valor do Duty para carregar em CCPR2L
                          //pois já carregamos os LSBs nas duas linhas acima.
  CCPR4L = VarAux2;      //Salva Duty em CCPR2L
  //VarAux = 0b0000111011
  //CCP1CON<5:4> = 11
  //VarAux = 0b0000111011 >> 2  =0b0000001110
  //CCPR1L = 0B00001110;
}

//Start na geração do sinal PWM em CCP2
void PWMStart_RB0_ESQ_TRAS(){
```

```
  T6CON.TMR6ON = 1;  //Liga TIMER e inicia a PWM
}

//Stop na geração do sinal PWM em CCP2
void PWMStop_RB0_ESQ_TRAS(){
  T6CON.TMR6ON = 0;  //desliga TIMER e inicia a P
}

//**********************CONFIGURAÇÃO DO PWM CCP5**********************
//Esta função calcula o Período do PWM com base na Freq enviado como parâmetro
// Cristal usado: Fosc = 16MHz
//
//1° Etapa: Cálculo do Período
//Fórmula:
//PR2 = ( ((Fosc)/(4 * FPWM * PRESCALER)) -1)
//PR2 = (16.000.000Hz / (4 * 20.000Hz * 4) ) - 1
//PR2 = 49
void PWMInit_RA4_DIR_TRAS(unsigned int Freq3){

  float Aux3, Tosc3;
  Tosc3 = (1./(Clock_kHz()*1000));       //Equivale a: 1./16000Hz*1000
                                 //Clock_KHz é uma função inline do compilador mikroC
                                 //no qual informa o valor do oscilador Fosc configurado no painel
                                 //edit Project

  Aux3 = 1./Freq3;                       //Encontra o período com base na frequência carregada no
parâmentro
  PR6 = ceil((Aux3/(4 * Tosc3 * 4)) - 1); //Encontra o valor de PR2.
                                 //A função ceil arredonda o resultado. Habilitar biblioteca C_Math
                                 //tomar cuidado que o resultado deverá estar entre 0 a 255
  CCPTMRS1.C5TSEL1 = 1;
  CCPTMRS1.C5TSEL0 = 0;          //Configurando o timer 6 para o pwm no pino RA4
```

```
  T6CON = 0b00000001;                //Desliga TIMER6 e carrega prescaler 1:4
  CCP5CON = 0b00001100;                    //Configura CCP no modo PWM. A partir desse momento o PWM já está ligado.
}

void PWMduty_RA4_DIR_TRAS(unsigned char Duty3){

   int VarAux3 = 0;
   VarAux3 = 1 * 4 * (PR6+1);                //Com base no PR2 já carregado, encontra valor máximo de contagem
                              //de TMR2, o que afeta diretamente na contagem do duty cycle
   VarAux3 =  (VarAux3 * (Duty3/100.));       //Regra de 3x simples, no qual converte o valor máximo encontrado
                              //em 0 a 100%
                              //Ex: Se o parâmetro da função Duty = 92
                              //VarAux =  (VarAux * (92/100.));
                              //VarAux passa a ser igual a 92% de seu valor original, ou seja
                              //Duty agora é de 92% do seu máximo valor
   //Carrega os dois LSB encontrado em VarAux nos bits CCP2CON<5:4>
   CCP5CON.B4 = ((VarAux3 & 0x01) == 0x01); //Mascara para setar/Resetar somente CCP2CON.B4
   CCP5CON.B5 = ((VarAux3 & 0x02) == 0x02); //idem para B5

   VarAux3 = VarAux3 >>2;  //Desloca para a direita o valor do Duty para carregar em CCPR2L
                        //pois já carregamos os LSBs nas duas linhas acima.
   CCPR5L = VarAux3;      //Salva Duty em CCPR2L
   //VarAux = 0b0000111011
   //CCP1CON<5:4> = 11
   //VarAux = 0b0000111011 >> 2  =0b0000001110
   //CCPR1L = 0B00001110;
}
```

```
//Start na geração do sinal PWM em CCP2
void PWMStart_RA4_DIR_TRAS(){
  T6CON.TMR6ON = 1;  //Liga TIMER e inicia a PWM
}

//Stop na geração do sinal PWM em CCP2
void PWMStop_RA4_DIR_TRAS(){
  T6CON.TMR6ON = 0;  //desliga TIMER e inicia a P
}
```

CONFIGURAÇÃO DE VARIAVEIS PARA CADA PINO USADO

```
#ifndef __Config__
#define __Config__

sbit eco1_direction at TRISC2_bit;
sbit trig1_direction at TRISC0_bit;

sbit infravermelho1_direction at TRISB4_bit;
sbit infravermelho2_direction at TRISB5_bit;
sbit infravermelho3_direction at TRISB6_bit;
sbit infravermelho4_direction at TRISB7_bit;

sbit pwm1_direction at TRISC1_bit;
sbit pwm2_direction at TRISC6_bit;
sbit pwm3_direction at TRISB0_bit;
sbit pwm4_direction at TRISA4_bit;

sbit LED_direction at TRISA5_bit;
sbit LED at LATA5_bit;
```

```
void ConfigMCU();
void ConfigInterrupcao();
void ConfigEntradas();
void ConfigSaidas();
void Config_CCP1();
void ConfigTimer1();
void ConfigChangeRB();
//CCP2
void PWMInit_RC1_DIR_FRE(unsigned int Freq);
void PWMduty_RC1_DIR_FRE(unsigned char Duty);
void PWMStart_RC1_DIR_FRE();
void PWMStop_RC1_DIR_FRE();
//CCP3
void PWMInit_RC6_ESQ_FRE(unsigned int Freq1);
void PWMduty_RC6_ESQ_FRE(unsigned char Duty1);
void PWMStart_RC6_ESQ_FRE();
void PWMStop_RC6_ESQ_FRE();
//CCP4
void PWMInit_RB0_ESQ_TRAS(unsigned int Freq1);
void PWMduty_RB0_ESQ_TRAS(unsigned char Duty1);
void PWMStart_RB0_ESQ_TRAS();
void PWMStop_RB0_ESQ_TRAS();
//CCP5
void PWMInit_RA4_DIR_TRAS(unsigned int Freq1);
void PWMduty_RA4_DIR_TRAS(unsigned char Duty1);
void PWMStart_RA4_DIR_TRAS();
void PWMStop_RA4_DIR_TRAS();


#endif
```

CONFIGURÇÃO DO ACIONAMENTO DOS MOTORES VIA PWM

```
#include "Funcoes.h"
#include "Config.h"

unsigned char robo_andando_frente = 0;
unsigned char robo_andando_esquerda = 0;
unsigned char robo_andando_direita = 0;
unsigned char robo_andando_tras = 0;



//*****************************Função Trigger1()*****************************
//envia pulso de 10us para o sensor ultrassônico 1(CCP1)
void trigger1(){
   trig1 = 1;
   Delay_us(10);
   trig1 = 0;
}
//*****************************Função ir para frente*****************************
//Faz com que os dois motores girem para levar o robo para frente
void Frente(){
   PWMStart_RC1_DIR_FRE();      //Timer 4
   PWMStart_RC6_ESQ_FRE();      //Timer 2

   robo_andando_frente = 1;
   robo_andando_tras = 0;
   robo_andando_esquerda = 0;
   robo_andando_direita = 0;
}
//*****************************Função ir para trás*****************************
//Faz com que os dois motores girem para levar o robo para trás
void Tras(){
```

```
   PWMStart_RB0_ESQ_TRAS();     //Timer 6
   PWMStart_RA4_DIR_TRAS();     //Timer 6

   robo_andando_frente = 0;
   robo_andando_tras = 1;
   robo_andando_esquerda = 0;
   robo_andando_direita = 0;
}
//*******************************Função parar*****************************
//Faz com que os dois motores parem de girar
void Parar(){
   PWMStop_RB0_ESQ_TRAS();
   PWMStop_RA4_DIR_TRAS();
   PWMStop_RC6_ESQ_FRE();
   PWMStop_RC1_DIR_FRE();

   CCPTMRS1.C5TSEL1 = 1;
   CCPTMRS1.C5TSEL0 = 0;            //Configurando o timer 6 para o pwm

   CCPTMRS1.C4TSEL1 = 1;
   CCPTMRS1.C4TSEL0 = 0;            //Configurando o timer 6 para o pwm
}
//************************Função girar para a direita*************************
//Faz com que o robô gire para a direita
void Direita(){

   CCPTMRS1.C4TSEL1 = 0;
   CCPTMRS1.C4TSEL0 = 0;            //Configurando o timer 2 para o pwm

   PWMStart_RC6_ESQ_FRE();          //Timer 4
   PWMStart_RA4_DIR_TRAS();         //Timer 6
```

```
   robo_andando_frente = 0;
   robo_andando_tras = 0;
   robo_andando_esquerda = 0;
   robo_andando_direita = 1;
}
//**************************Função girar para a esquerda************************
//Faz com que o robô gire para a esquerda
void Esquerda(){

   CCPTMRS1.C5TSEL1 = 0;
   CCPTMRS1.C5TSEL0 = 1;            //Configurando o timer 4 para o pwm

   PWMStart_RB0_ESQ_TRAS();         //Tiemr 6
   PWMStart_RC1_DIR_FRE();          //Timer 2

   robo_andando_frente = 0;
   robo_andando_tras = 0;
   robo_andando_esquerda = 1;
   robo_andando_direita = 0;
}
```

DECLARÇÃO DAS FUNÇÕES PARA AS AÇÕES DOS MOTORES

```
#ifndef __Funcoes__
#define __Funcoes__

sbit trig1 at LATC0_bit;

//protótipo de funções
void trigger1();
void Frente();
void Tras();
```

```
void Parar();
void Direita();
void Esquerda();

#endif
```