INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA.

CAMPUS JOINVILLE
CURSO SUPERIOR DE TECNOLOGIA EM MECATRÔNICA INDUSTRIAL

ALEXSANDER MICHAEL CORREA
JOSÉ VANILDO MARTINS SANTOS

ROBÔ MÓVEL COM RODAS OMNIDIRECIONAIS

TRABALHO DE CONCLUSÃO DE CURSO

ALEXSANDER MICHAEL CORREA
JOSÉ VANILDO MARTINS SANTOS

ROBÔ MÓVEL COM RODAS OMNIDIRECIONAIS

JOINVILLE, 2013

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA.
CAMPUS JOINVILLE
CURSO MECATRÔNICA INDUSTRIAL

ALEXSANDER MICHAEL CORREA
JOSÉ VANILDO MARTINS SANTOS

ROBÔ MÓVEL COM RODAS OMNIDIRECIOANIS

Submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos de obtenção do título de Tecnólogo em Mecatrônica Industrial.

Orientador: Michael Klug, MSC.

JOINVILLE, 2013

Correa, Alexsander Michael; Santos, José Vanildo Martins.
Robô Móvel com Rodas Omnidirecionais / Correa, Alexsander Michael; Santos, José Vanildo Martins – Joinville: Instituto Federal de Santa Catarina, 2013. 94 f.

Trabalho de Conclusão de Curso - Instituto Federal de Santa Catarina, 2013. Graduação. Curso Superior de Tecnologia em Mecatrônica Industrial. Modalidade: Presencial.
Orientador: Michael Klug, Msc.

1. Omnidirecional 2. Robótica Móvel I. Título

ROBÔ MÓVEL COM RODAS OMNIDIRECIONAIS

ALEXSANDER MICHAEL CORREA
JOSÉ VANILDO MARTINS SANTOS

Este trabalho foi julgado adequado para obtenção do título de Tecnólogo em Mecatrônica Industrial e aprovado na sua forma final pela banca examinadora do Curso Mecatrônica Industrial do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Joinville, 23 de Julho de 2013.

Banca Examinadora:

_____
Prof. Michael Klug, Mestre
Orientador

_____
Prof. Janderson Duarte, Mestre
Avaliador

_____
Prof. Valter Vander de Oliveira, Mestre
Avaliador

AGRADECIMENTOS

Agradecemos em primeiro lugar a Deus por todos os acontecimentos em nossas vidas, pelas dificuldades que tivemos durante o desenvolvimento deste trabalho e que nos ensinaram que por mais difícil que seja uma situação, nunca se deve desistir, devemos sempre acreditar e ir atrás de nossos objetivos.

Aos nossos pais pelo amor, dedicação, educação e confiança, sentimentos que fizeram com que lutássemos com determinação.

Às nossas esposas e companheiras por todo o apoio e ajuda que obtivemos nos momentos de dificuldades.

Ao professor, orientador e amigo Michael Klug pelo apoio e compreensão pessoal neste trabalho.

A todos os professores, colegas e amigos que de alguma forma participaram de mais esta etapa que cumprimos de nossas vidas.

A todos fica aqui registrado o nosso muito OBRIGADO.

RESUMO

Este trabalho de conclusão de curso (TCC) consiste no desenvolvimento de um robô móvel com possibilidade de movimentação em todas as direções sem mudar a sua orientação, dando mais agilidade ao robô. Para garantir esta mobilidade, utilizou-se uma concepção de robô com três rodas omnidirecionais. Na parte de comando fez-se uso de um *joystick*, que em conjunto com um software, envia comandos via rede sem fio ao robô, tendo também a possibilidade de pequenos percursos automáticos. O robô foi construído em estrutura de alumínio e contém uma câmera, um módulo GPS, baterias recarregáveis e uma interface de comando simples. Para a programação utilizou-se do ambiente de desenvolvimento integrado Delphi juntamente com o Arduino. Pretende-se, com este projeto, a disseminação da robótica móvel e a aplicação dos conhecimentos obtidos durante o curso.

Palavras-chave: Omnidirecional. Robô Móvel. Rede sem Fio.

ABSTRACT

This work of course completion (WCC) is the development of a mobile robot with the possibility of moving in all directions without changing its orientation, giving more flexibility to the robot. To ensure this mobility it is used a robot design with three omnidirectional wheels. In the control use is made of a joystick, which, together with software sends commands through the wireless network to the robot, and also the possibility of automatic short trajectories. The robot was built aluminum frame and contains a camera, a GPS module, rechargeable batteries and a simple command interface. For programming we used the Delphi integrated development environment together with the Arduino. It is intended with this project, the spread of mobile robotics and application of knowledge obtained during the course.

Keywords: Omnidirectional. Mobile Robot. Wireless Network.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

Por conceito, robótica móvel é a área da robótica que trata de dispositivos sem base fixa, que possuem a capacidade de locomover-se em um ambiente limitado ou não, tendo autonomia para interagir de forma direta com o meio em que estão [SECCHI, 2008]. Dentro da robótica móvel encontra-se o robô omnidirecional, um dispositivo que tem como principal característica a facilidade na mudança de direção [NASCIMENTO, 2009].

Esta concepção de robô pode se deslocar em todas as direções no plano sem a necessidade de mudança de orientação, isto se dá em função das rodas omnidirecionais, também conhecidas como rodas suecas [SECCHI, 2008]. Este tipo de roda possui a propriedade de diminuir consideravelmente o atrito com o solo, já que possuem roldanas perpendiculares à linha de tração que permitem o deslizamento da roda [NASCIMENTO, 2009]. Esta característica atribui ao robô omnidirecional uma maior facilidade na mudança de direção, sendo muito utilizado no futebol de robôs e transporte de carga [SOBRINHO, 2011].

No projeto deste TCC desenvolveu-se um robô omnidirecional de três rodas controlado manualmente por *joystick,* e também por percursos simples feitos na interface de comando localizado no computador. Esses comandos passam primeiramente pelo Delphi para depois serem enviados, via rede sem fio, para o robô, ao qual uma placa Arduino [MCROBERTS, 2011] decodifica e distribui os sinais de controle entre os motores. Posteriormente, através das imagens obtidas pelo robô, o mesmo poderá ser utilizado para inspeção, reconhecimento de locais de difícil acesso e tarefas autônomas.

O projeto tem por objetivo interligar boa parte do que foi estudado durante o curso, sendo assim um desafio para os acadêmicos por em prática conhecimentos de várias áreas como eletrônica, mecânica, programação, robótica, projetos, entre outras.

O principal desafio na construção deste tipo de robô está na forma de sincronização das rodas. O primeiro passo para isso está no entendimento da cinemática do robô para, posteriormente, encontrar a equação que a represente.

1.1 Organização

Este trabalho está divido em quatro capítulos, além da introdução e conclusão. No Capítulo 2 está a revisão bibliográfica, na qual se descreve o funcionamento, as vantagens e as principais utilizações dos componentes utilizados na construção do robô.

O Capítulo 3 apresenta o projeto que é uma etapa fundamental no processo, como decisões a respeito de materiais, tipos de equipamentos, custos e prazos de execução. Também nesta etapa foram gerados os desenhos com auxílio de um software 3D e feita a análise da cinemática do equipamento

No Capítulo 4 mostra-se a etapa de fabricação, onde ocorreu a transformação daquilo que foi pensado e projetado em algo concreto. Esta transformação se deu através de máquinas-ferramentas, ferramentas de corte, instrumentos de medição e ferramentas utilizadas para confecção de circuitos eletrônicos.

No Capítulo 5 é abordada a parte de programação responsável pelo controle, incluindo programas do Delphi e do Arduino, e também os testes de funcionamento do robô.

Por último, no Capítulo 6 têm-se a conclusão juntamente com a sugestão de trabalhos futuros.

2 COMPONENTES UTILIZADOS

Neste tópico será apresentado os principais componentes utilizados na construção do robô omnidirecional.

2.1 Rodas Omnidirecionais

As rodas omnidirecionais, também conhecidas como rodas suecas, são aplicadas a dispositivos específicos que permitem a movimentação em todas as direções sem a necessidade da mudança de orientação do robô.

Este tipo de dispositivo possui pequenas roldanas ao longo da sua circunferência, conforme pode-se observar na Figura 1, posicionadas perpendicularmente ao eixo principal. Desta forma, em determinado movimento o trabalho realizado é de uma roda comum, mas quando há mudança na direção ocorre um deslizamento devido às pequenas roldanas [RIBEIRO et al, 2004 e NASCIMENTO, 2009].

Suas aplicações estão relacionadas principalmente ao transporte de materiais e futebol de robôs.

FIGURA 1 - Roda omnidirecional
Fonte: www.vexrobotics.com

2.2 Wi-Fi

Para tornar o robô omnidirecional independente de cabos de comando, utilizou-se dispositivos de rede local sem fio provenientes do padrão IEEE 802.11, popularmente conhecido como *Wi-Fi.* Esta abreviação (do inglês *Wireless Fidelity*), que significa fidelidade sem fio, refere-se a uma tecnologia que permite conectar dispositivos dentro de um raio pré-determinado sem a necessidade de cabos de transmissão, o que torna os dispositivos muito mais práticos e flexíveis, do ponto de vista da mobilidade.

O funcionamento das redes sem fio se dá por meio de ondas de rádio que são transmitidas por meio de um roteador. Este, por sua vez, recebe o sinal, decodifica e os emite a partir de uma antena para outros dispositivos que estejam dentro do raio de ação do roteador, chamado de *hotspot.*

Outras importantes vantagens do *Wi-Fi*, além da conexão sem fios, são a grande capacidade de transmissão de dados e quantidade de aparelhos comuns disponíveis. [HAYKIN, 2008].

2.3 Ethernet

Na comunicação interna do robô, as conexões entre roteador/Arduino e roteador/câmera, foram viabilizadas pela tecnologia de rede local (*LAN*, do inglês *Local Area Network*) mais utilizada atualmente, a *Ethernet*. Este sistema, que é baseado no envio de pacotes de dados, permite o envio e o recebimento de informações entre dispositivos por meio de cabos de transmissão. Ela foi padronizada pelo Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) como 802.3 e define cabeamentos, sinais elétricos, formato de pacotes e protocolos de envio [SPURGEON, 2000].

O protocolo utilizado na Ethernet *é* o *Carrier Sense Multiple Access With Collision Detection* (CSMA/CD), que significa acesso múltiplo com detecção de portadora e detecção de colisão. Neste protocolo, uma determinada estação precisa esperar para poder transmitir (*Carrier Sense*), e todas as estações têm a mesma prioridade de acesso para transmissão (*Multiple Access*). Assim, poderá ocorrer de várias estações mandarem seus dados simultaneamente gerando colisões de dados. Neste momento os dispositivos de sinalização conectados à rede, percebem a colisão e sinalizam as estações que parem de transmitir. Em seguida, cada uma das estações escolhe um tempo aleatório para retransmitir os dados.

2.4 Câmera

Para que o robô fornecesse imagens em tempo real utilizou-se uma câmera *IP* (do inglês*, Internet Protocol*). Este tipo de dispositivo permite ser acessado e controlado utilizando uma rede *LAN* e um navegador *web*, justamente por possuir um endereço de *IP*. Internamente possui um servidor e uma placa de processamento, não sendo necessária a utilização de softwares ou placas adicionais.

Para funcionar é preciso estar alimentada e conectada via *Wi-Fi* ou cabo de rede a um roteador. Na sequência conecta-se outro dispositivo neste roteador, e através de um navegador *web* tem-se acesso à câmera. Concluída a conexão, é possível ver as imagens geradas e controlá-las, pois geralmente possui visão noturna e recursos de movimentação denominados *PTZ* (do inglês*, pan, tilt e zoom*).

Sua aplicação se dá principalmente para vigilância tanto de locais internos quanto externos, e sua principal vantagem é a possibilidade de ser acessada pela Internet [PINHEIRO, 2006].

Para haver comunicação entre computador/câmera e obter as imagens foi preciso acessá-la via navegador web. A câmera foi conectada ao roteador por um cabo de rede e fornecia as

informações por meio desta conexão. Na sequência, é necessário conectar o computador ao roteador e então usufruir das imagens em tempo real. Na Figura 2 observa-se o modelo de câmera utilizado no robô.

FIGURA 2 - Câmera *IP*
Fonte: www.cidadesaopaulo.olx.com.br

A alimentação da câmera é efetuada através de um cabo conectado a bateria. Como a tensão suportada pela câmera é de apenas 5V, houve a necessidade de implementar um regulador de tensão na entrada, já que a bateria fornece 11,1V.

2.5 Motor

Os responsáveis pela movimentação do robô foram os motores de corrente contínua (CC). Este tipo de motor é composto por duas principais estruturas magnéticas que são o estator e o rotor.

O estator é composto de uma estrutura ferromagnética onde são enroladas as bobinas e formam o campo, enquanto que o rotor é um eletroímã formado por um núcleo de ferro tendo em sua superfície um enrolamento alimentado por um sistema mecânico de comutação.

Seu princípio de funcionamento baseia-se na atração e repulsão do eletroímã e seus polos. A aplicação deste tipo de motor é muito variada, sendo encontrada na indústria e nos dispositivos robóticos [HONDA, 2006]. Na Figura 3 é mostrado o motor CC.

FIGURA 3 - Motor CC
Fonte: www.neoyama.com.br

Como o robô não tem por requisito uma grande velocidade de deslocamento foram utilizados motores CC com redução. Estes motores já vêm com uma pequena caixa de redução acoplada ao seu corpo, fazendo-se desnecessário o uso de uma combinação de engrenagens para reduzir a rotação, facilitando bastante o controle de velocidade.

2.6 Joystick

O controle dos motores é feito em várias etapas, sendo que na primeira está o *joystick,* e que encontra-se conectado ao

computador via *USB* (do inglês*, Universal Serial Bus*) [OLIVEIRA, 2006]. Este tipo de conexão permite a alimentação e a transferência de dados simultaneamente.

Os *joysticks,* em geral, possuem manches analógicos, conforme Figura 4. Para o computador poder interpretar estes sinais analógicos é necessário transformá-los em sinais digitais. Para isso, tem-se um conversor analógico/digital na saída do *joystick* que permite esta conversão [ROCHA, 2012]. Concluída esta etapa, os sinais ou coordenadas são enviados para o Delphi, no caso do robô omnidirecional.

FIGURA 4 – *Joystick*
Fonte: www.todaoferta.uol.com.br

2.7 Delphi

Na interface de comando e no processamento fez-se uso do Embarcadero Delphi, justamente por se tratar de um software muito versátil e de fácil aprendizagem. Mais conhecido como Delphi, ele é um compilador, um ambiente de desenvolvimento integrado (*IDE*, do inglês *Integrated Development Environment)* e uma linguagem de programação muito utilizada na construção de aplicativos desktop e aplicações multicamadas cliente/servidor. Também pode ser empregado em diversos tipos de

desenvolvimento de projetos, incluindo serviços e aplicações web [CANTÙ, 2000].

O Delphi, por ser extensível, possui uma *IDE* muito maleável, podendo ser facilmente modificada conforme a necessidade. Utilizando-se do Object Pascal, uma linguagem de programação direcionada a objeto, é possível construir janelas de maneira visual, simplesmente arrastando e soltando os objetos integrantes da interface do usuário, sendo que o próprio Delphi escreve o código fonte, tornando-se ágil e fácil de utilizá-lo [CANTÙ, 2000].

O Delphi é um software extremamente versátil que cria, dentre várias coisas, aplicativos muito elaborados. No caso do robô omnidirecional, desenvolveu-se uma interface de comando composta por uma tela de visualização das imagens da câmera, visualizador de posição do manche, posição do robô (coordenadas) e nível de carga da bateria.

Além de interface de comando, o Delphi faz o processamento das coordenadas convertendo-as em comandos de acionamento dos motores. Esta conversão ocorre na decomposição das coordenadas ao passar pela equação da cinemática do robô, que está embutida dentro do Delphi. Em seguida, os comandos são enviados via rede sem fio para o Arduino, que os decodifica e os distribui entre os motores.

2.8 Arduino

2.8.1 Arduino Uno

Esta ferramenta é destinada ao controle de sistemas físicos a nível móvel, comercial e doméstico, sendo composta por um microcrontrolador, suporte de entradas e saídas e linguagem de programação padrão, podendo ser observado na Figura 5.

FIGURA 5 - Arduino Uno
Fonte: www.arduino.cc

Isto faz do Arduino uma ferramenta muito útil no desenvolvimento de protótipos eletrônicos. Ele pode ser usado em sistemas autônomos utilizando-se da rotina desenvolvida na sua *IDE*, ou então, pode comunicar-se com um software instalado no computador.

O Arduino possui um microcontrolador ATmega328 de 8 bits, da família AVR, fabricado pela Atmel e programável em linguagem C++. Possui 32 kB (kilobytes) de memória flash sendo capaz de executar várias funções num único ciclo, equilibrando consumo de energia e velocidade de processamento.

Os principais objetivos do Arduino são de diminuir a complexidade e o custo dos projetos eletrônicos. Possui seus códigos fonte abertos podendo ser facilmente modificados e ampliados, auxiliando no desenvolvimento de protótipos por parte dos usuários em geral [MCROBERTS, 2011].

2.8.2 Ethernet Shield

O Ethernet Shield, visto na Figura 6, é um circuito que permite a uma placa de Arduino conectar-se a uma rede Ethernet. Isto é possível devido a um chip de Ethernet W5100 Wiznet.

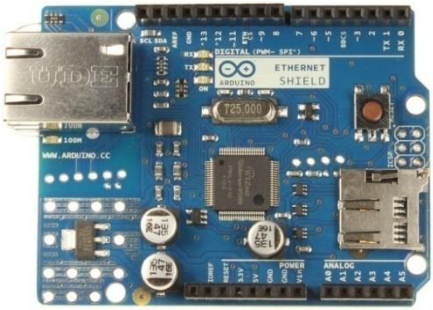


FIGURA 6 - *Ethernet Shield*
Fonte: www.arduino.cc

Este recurso possibilita o controle de um dispositivo através de comandos vindos de um site, ou de outro dispositivo conectado na mesma rede que o *Ethernet Shield* [MCROBERTS, 2011].

No caso do robô omnidirecional houve a necessidade de se aplicar este circuito, pois o controle dos movimentos é feito remotamente pelo software de controle instalado em um computador.

2.8.3 Ponte H

A ponte H é um circuito simplificado cuja finalidade é fazer o controle dos motores CC. O nome do mesmo se dá em função dos transistores que operam como chaves e que permitem a inversão do sentido de rotação [OLIVEIRA, 2006]

Esses circuitos podem ser construídos ou adquiridos comercialmente, como foi o nosso caso. Sua aplicação ocorre principalmente na robótica. A Figura 7 mostra um exemplo de Ponte H comercial.

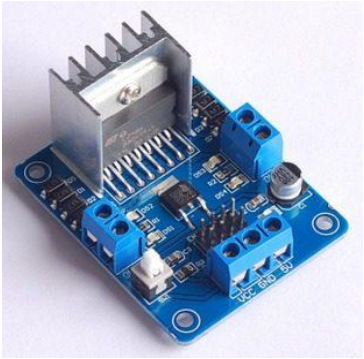


FIGURA 7 – Ponte H
Fonte: www.mercadolivre.com.br

2.9 Bateria Lipo

Umas das características principais do robô é a autonomia, e para alcançar esta característica utilizaram-se baterias Lipo. Este tipo de bateria é composta por polímeros de Lithium e possui células de 3,7 V de tensão. Sua corrente é definida em mA (miliamperes) e está diretamente ligada com a duração da bateria, conforme Figura 8.

FIGURA 8 - Bateria Lipo
Fonte: www.amigomodelista.com.br

A vantagem dessas baterias é o tamanho significativamente menor que outras de mesma capacidade de descarga de corrente, que normalmente vem especificada na embalagem do produto. A capacidade de descarga é definida em C (Coulomb).

A bateria de Lipo é largamente utilizada no modelismo, justamente por ter o tamanho reduzido e ter uma alta corrente de descarga [RITA, 2009].

3 PROJETO DO ROBÔ

O ponto de partida para a construção do robô omnidirecional foi a etapa do projeto. Nesta fase, utilizaram-se os conhecimentos adquiridos na unidade curricular de Projeto de Máquinas para tomar decisões importantes relativas ao robô.

A etapa de projeto, quando bem executada, reduz o risco do produto ter insucesso. Isto porque, evita-se o desperdício de material, tempo, e busca-se sempre a redução de custo, desde que não interfira no funcionamento e na durabilidade do produto. Além disso, busca-se também, a melhor solução para cada requisito solicitado pelo cliente [BLANCHARD e FABRYCKY, 1990].

É nesta etapa que as ideias são juntadas, organizadas e melhoradas. Além disso, são especificados materiais a serem utilizados, custos do produto, componentes, bem como os desenhos para a etapa de fabricação.

3.1 Requisitos

No caso do robô omnidirecional, primeiramente, listaram-se os requisitos do projeto, como vemos a seguir:

1. Movimentos em todas as direções com a mesma orientação;
2. Mobilidade (sem cabos de alimentação ou de comandos);
3. Autonomia;
4. Bom aspecto externo;
5. Formato triangular da carcaça;
6. Baixo peso e custo;
7. Controle manual comandado por Joystick;
8. Percursos automáticos simples;
9. Fácil interface de comando;

10. Fornecimento de imagens em tempo real independente da iluminação;

3.2 Especificações

Em seguida, especificou-se os requisitos. Nesta parte do projeto damos números aos requisitos, ou seja, delimitamos os limites mínimos ou máximos:

1. Velocidade mínima: 50 m/min;
2. Distanciamento de no mínimo 10 m da central de comando;
3. Duração de bateria: 2 h;
4. Componentes não devem estar visíveis;
5. Largura máxima: 350 mm, Altura: 125 mm, Altura do chão: 25 mm;
6. Carga máxima: 10 kg.

3.3 Soluções

Depois, pesquisaram-se várias soluções para as funções ou requisitos listados anteriormente e identificou-se as vantagens e desvantagens delas. Com isso, foi possível determinar qual a solução mais apropriada para cada requisito, como pode ser visto na Tabela Desenvolvimento de Soluções, localizada no Apêndice A.

3.4 Estrutura Funcional

Para o melhor entendimento do funcionamento do robô desenvolveu-se uma estrutura funcional, como observado na Figura 9.

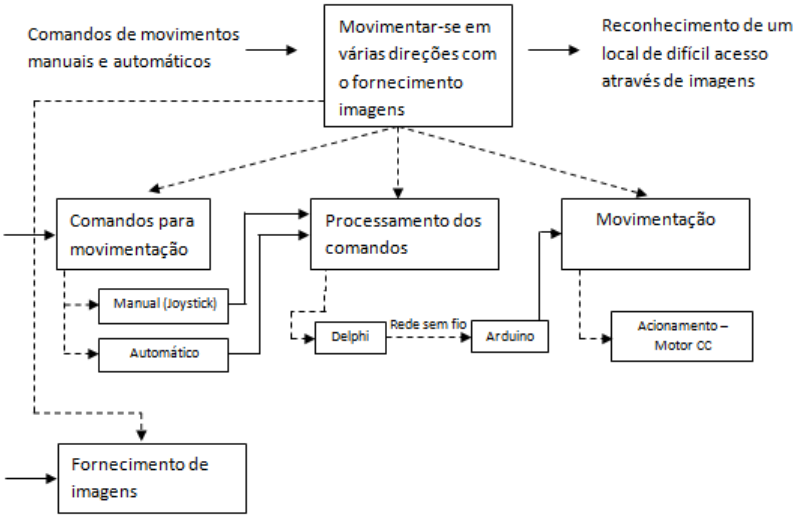


FIGURA 9 - Estrutura funcional

3.5 Modelamento

Após a escolha dos equipamentos e métodos a partir da planilha de soluções, deu-se início a modelagem do robô no software 3D SolidWorks.

As peças a serem usinadas foram modeladas no software, onde conseguiu-se ter uma primeira visualização. Por ser um software 3D a visualização é facilitada, bem como a montagem de todas as partes mecânicas, conforme Figura 10. Uma simulação de montagem é realizada virtualmente para verificarmos a funcionalidade de todas as partes e se alguma peça estiver errada, é feita uma correção, evitando desperdício de material, tempo e dinheiro.

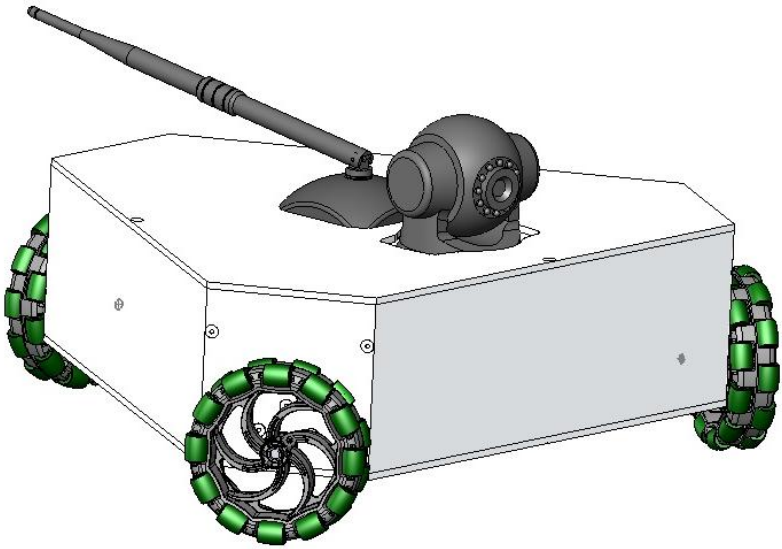


FIGURA 10 - Modelo geométrico do robô

Depois da verificação da montagem virtual dos componentes, utilizando-se do mesmo software, são gerados os desenhos em 2D contendo todas as informações necessárias para a fabricação das peças.

3.6 Placa de Circuito Impresso

No robô foi necessária a confecção de uma placa de circuito impresso para fazer a comunicação entre o microcontrolador e a ponte H, além de receber a tensão das baterias, possibilitando a leitura das mesmas.

O circuito eletrônico foi desenhado com auxílio do software 2D TraxMaker (ferramenta do CircuitMaker), que proporcionou

uma boa visualização da disposição dos componentes e das linhas do circuito [RIBEIRO, 2008], como mostrado na Figura 11.

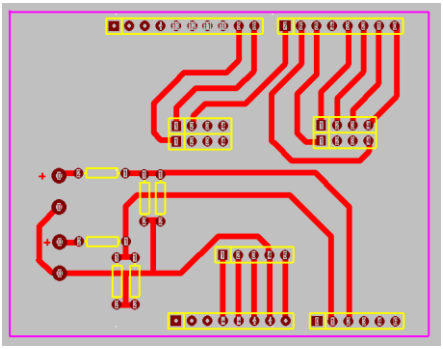


FIGURA 11 - Desenho da placa eletrônica para comunicação

3.7 Cinemática

A cinemática de um robô é um estudo muito importante para o projeto, pois é através dela que conseguimos encontrar quais as velocidades e movimentos que o robô terá. Neste estudo, em que as forças não são consideradas, encontram-se as equações que representam a cinemática do mecanismo e que propiciam o cálculo de velocidades lineares e angulares, assim como posição e orientação [SECCHI, 2008], [PINHEIRO et al, 2010]. É a partir da cinemática que se desenvolve a parte de controle para o dispositivo.

No caso do robô omnidirecional, criou-se primeiramente um esboço da sua estrutura para a melhor visualização, como na Figura 12. Nela é possível observar a disposição das rodas.

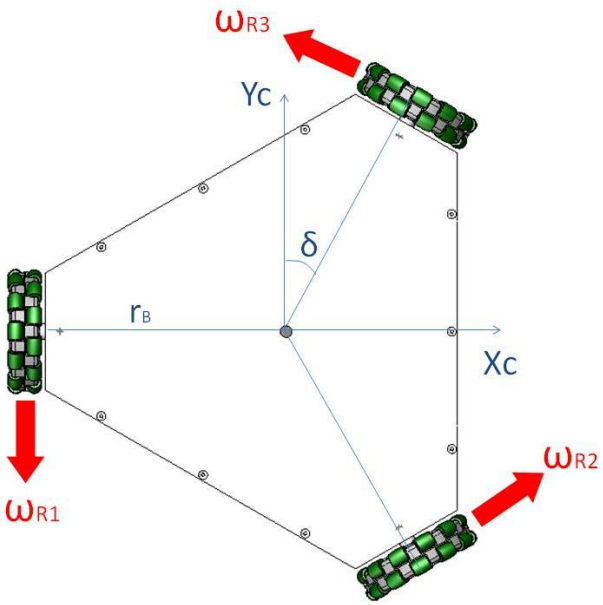


FIGURA 12 - Análise geométrica das velocidades angulares

Analisando a figura encontraram-se as seguintes equações:

$$\omega_{R1} = -V_{Cy} + \omega_c \times r_B \qquad (3.1)$$

$$\omega_{R2} = V_{Cx} \times cos(\delta) + V_{Cy} \times sen(\delta) + \omega_c \times r_B \ , \qquad (3.2)$$

$$\omega_{R3} = -V_{Cx} \times cos(\delta) + V_{Cy} \times sen(\delta) + \omega_c \times r_B \qquad (3.3)$$

sendo:

$\omega_R = velocidade\ angular\ da\ roda$;
$V_C = velocidade\ do\ centro\ de\ massa\ do\ robô$;
$\omega_C = velocidade\ angular\ do\ centro\ de\ massa\ do\ robô$;
$r_B = raio\ da\ base\ do\ robô$;
$\delta = ângulo\ das\ rodas\ 2\ e\ 3\ em\ relação\ ao\ eixo\ Y_C$.

Os sistemas de equações das velocidades angulares das rodas também podem ser escritos na forma matricial

$$\begin{bmatrix}\omega_{R1}\\ \omega_{R2}\\ \omega_{R3}\end{bmatrix} = \begin{bmatrix}0 & -1 & r_B\\ cos(\delta) & sen(\delta) & r_B\\ -cos(\delta) & sen(\delta) & r_B\end{bmatrix} \times \begin{bmatrix}V_{Cx}\\ V_{Cy}\\ \omega_C\end{bmatrix} , \quad (3.4)$$

ou ainda na forma resumida

$$\vec{\omega} = B^T \times V_C \ . \quad (3.5)$$

Assim, pode-se obter a relação entre velocidade angular do motor e a velocidade angular da roda

$$\omega_{Ri} = \frac{r_\omega}{\eta.N} \times \omega_{mi} \ . \quad (3.6)$$

Na qual:

$\omega_{mi} = velocidade\ angular\ do\ motor$;
$N = fator\ de\ acoplamento$;
$\eta = eficiência\ do\ acoplamento\ da\ roda$;
$r_\omega = raio\ da\ roda$;
$\omega_{Ri} = velocidade\ angular\ da\ roda$.

Substituindo na equação (3.4), tem-se:

$$\frac{r_\omega}{\eta \times N}\begin{bmatrix}\omega_{m1}\\ \omega_{m2}\\ \omega_{m3}\end{bmatrix} = \begin{bmatrix}0 & -1 & r_B\\ cos(\delta) & sen(\delta) & r_B\\ -cos(\delta) & sen(\delta) & r_B\end{bmatrix} \text{ x } \begin{bmatrix}V_{Cx}\\ V_{Cy}\\ \omega_C\end{bmatrix} \qquad (3.7)$$

Para obter a velocidade do robô, isola-se o termo Vc:

$$\begin{bmatrix}V_{Cx}\\ V_{Cy}\\ \omega_c\end{bmatrix} = (B^T)^{-1} \times \frac{r_\omega}{\eta \times N}\begin{bmatrix}\omega_{m1}\\ \omega_{m2}\\ \omega_{m3}\end{bmatrix} . \qquad (3.8)$$

Por fim, substituindo os termos encontra-se a equação final da cinemática direta

$$\begin{bmatrix}V_{Ix}\\ V_{Iy}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos(\theta) & -sen(\theta) & 0\\ sen(\theta) & cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix} \times (B^T)^{-1} \times \frac{r_\omega}{\eta \times N}\begin{bmatrix}\omega_{m1}\\ \omega_{m2}\\ \omega_{m3}\end{bmatrix} . \qquad (3.9)$$

3.8 Cinemática Inversa

Para este caso tem-se a velocidade final do robô e precisa-se descobrir a velocidade angular necessária para inserir nos motores. Para isso, aproveitamos a equação da cinemática direta e apenas isolamos o ω_m para, assim, obter as velocidades angulares de cada motor

$$\begin{bmatrix}\omega_{m1}\\ \omega_{m2}\\ \omega_{m3}\end{bmatrix} = \frac{r_\omega}{\eta \times N} \times B^T \begin{bmatrix}cos(\theta) & sen(\theta) & 0\\ -sen(\theta) & cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix} \times \begin{bmatrix}V_{Ix}\\ V_{Iy}\\ \dot{\theta}\end{bmatrix} . \qquad (3.11)$$

4 FABRICAÇÃO E MONTAGEM

A partir dos desenhos em 2D contendo as informações das peças a serem fabricadas, partiu-se para a usinagem. Foram utilizadas máquinas ferramentas e diversas ferramentas de corte no processo de fabricação. Os desenhos das peças usinadas com os detalhes podem se vistos no Apêndice B.

A carcaça do robô omnidirecional foi produzida pelo processo de usinagem. Sua função é de abrigar e ocultar os componentes do robô. Dentre os requisitos principais está o baixo peso, a fim de evitar motores com torque elevado. Em função disso, avaliaram-se vários tipos de materiais e devido à boa usinabilidade, ao peso específico e alta disponibilidade optou-se pelo alumínio [RIBEIRO e CUNHA, 2004].

Antes de adquirir os materiais e usiná-los surgiu a idéia de confeccionar a carcaça em chapas de alumínio, com corte a laser, modelamento através de dobras e fixação com solda TIG. Apesar da vantagem de um menor tempo de construção e um melhor aspecto não seria viável, pois aumentaria em muito o custo do robô, o que não atenderia um dos principais requisitos, o de custo baixo.

4.1 Torno Convencional

Esta máquina é utilizada para a usinagem de peças com formato cilíndrico. No caso do torno a peça gira em torno do seu próprio eixo e a ferramenta faz um movimento linear, onde a combinação dos dois movimentos gera a remoção do cavaco [SOUZA, 2011].

Foram usinados nesta máquina o acoplamento do eixo do motor e o adaptador de fixação do motor, como pode ser visto na Figura 13.

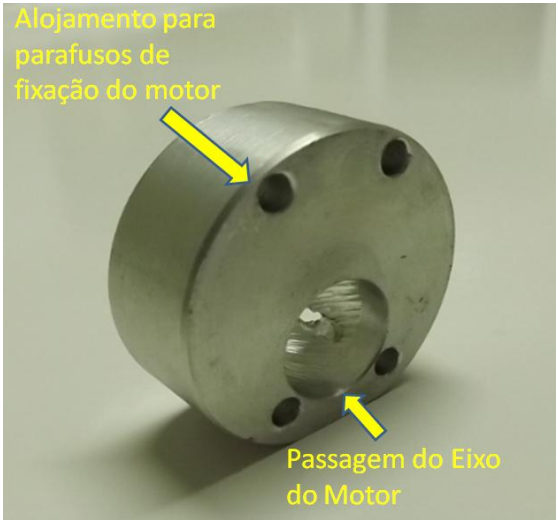


FIGURA 13 - Adaptador de fixação do motor

4.2 Fresadora Convencional

As peças com formatos planos de menor complexidade foram feitas na fresadora convencional. Este tipo de máquina é indicada para usinar superfície planas, canais e furações [ROSA e SIQUEIRA, 2007].

Na fresadora convencional foram feitas as laterais do robô, esquadrejamento, encaixe das laterais, e a furação por onde passa o eixo do motor tanto das laterais quanto dos adaptadores, como mostrado na Figura 14.

FIGURA 14 - Lateral com passagem para parafusos e eixo do motor

Também na fresadora, foram feitas as furações para fixação do motor no adaptador e a fixação deste na lateral da carcaça, como observado na Figura 15. Para finalizar, o furo nas rodas para acoplá-las no eixo do motor também foram usinadas nesta máquina.

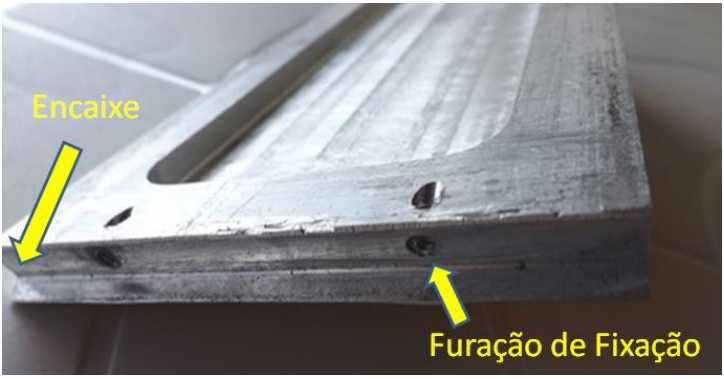


FIGURA 15 - Lateral com encaixes e furação de fixação

4.3 Fresadora CNC

Por possuir geometria mais complexa que as outras peças, a tampa e base do robô foram usinadas na fresadora CNC (Comando Numérico Computadorizado). Neste tipo de máquina a usinagem é feita de modo automático sendo programada em linguagem ISO (do inglês, *International Organization for Standardization*), norma que padroniza os códigos utilizados na programação CNC. Este tipo de máquina torna a usinagem mais ágil, confiável e flexível [HARBS, 2012].

As operações feitas pela fresadora CNC na base foram o contorno e o posicionamento das furações de passagem de parafusos. Na tampa, foram feitos o contorno, os posicionamentos e o alojamento para a câmera *IP*, como pode ser observado na Figura 16. O programa para usinar o contorno e o posicionamento da furação foi feito manualmente, no caso do alojamento da câmera, o programa foi gerado pelo *software* Edgecam.

FIGURA 16 - Tampa com alojamento da câmera *IP*

O Edgecam é um *software* CAM (do inglês, *Computer Aided Manufacturing*) que auxilia na programação de máquinas CNC [FETT, 2010]. CAM significa manufatura assistida por computador e é utilizado em casos onde a geometria da peça é muito complexa. No caso da câmera, tivemos que modelá-la no SolidWorks e exportar para o Edgecam. Em seguida, indicamos ao software a região na qual deveria calcular o percurso de usinagem da ferramenta. Na sequência gerou-se o programa em linguagem ISO e transmitiu-se o programa para o centro de usinagem CNC.

4.4 Furadeira de Bancada

Neste equipamento foram terminados os furos de passagem de parafuso iniciados na fresadora CNC, e também os escareamentos destes para o alojamento da cabeça dos parafusos, dando um melhor aspecto de acabamento na montagem. Para essa operação utilizou-se brocas de aço rápido.

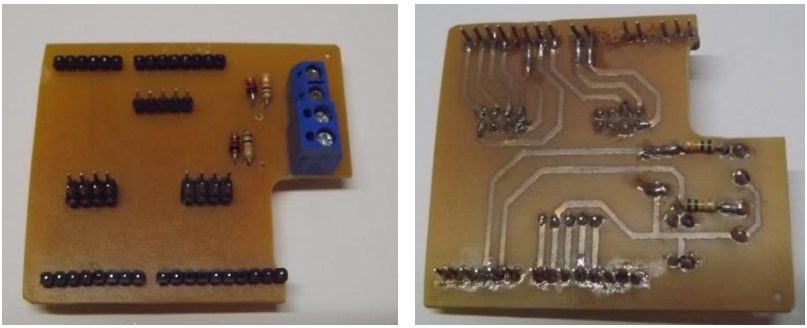
4.5 Medições

Todo o processo de medição foi realizado utilizando paquímetros quadricionais e micrômetros. A centralização das peças nas máquinas durante a preparação para a usinagem foi feito através de centralizadores de arestas

4.6 Placa Eletrônica

Na fabricação da placa eletrônica, como pode ser observado na Figura 17(a) e 17(b), foram utilizadas placas de fenolite como base para impressão do circuito, percloreto de ferro

para corroer a placa de cobre, a furadeira de bancada juntamente com uma broca de aço rápido para fazer a furação para a inserção dos componentes.



(b) - Frente **(a) - Verso**

FIGURA 17 - Placa eletrônica para comunicação

4.7 Montagem

Estando todos os componentes confeccionados iniciou-se o processo de montagem. Para montar a carcaça e fazer a fixação utilizaram-se parafusos M4 de cabeça chata com sextavado interno. A escolha deste tipo de parafuso se deu em função do pouco espaço para o alojamento do parafuso. O objetivo foi deixar a face do parafuso rente à face das laterais, melhorando o aspecto externo.

A disposição das peças usinadas, rodas e motores foram predefinidos no projeto. Os circuitos, câmera, roteador e baterias foram alocados conforme espaço interno da carcaça do robô, de forma a aproveitar ao máximo todos os espaços.

O restante das peças como motores, rodas omnidirecionais, circuitos Arduino, câmera, roteador e baterias foram unidos através de parafusos, fios e cabos às peças usinadas. A Figura 19 observa-se o parafuso rente à face.

FIGURA 18 - Parafuso de fixação rente à face

A seguir, a Figura 19 mostra o robô com a montagem concluída.

FIGURA 19 - Montagem Final do Robô

5 PROGRAMAÇÃO

5.1 Delphi

Na primeira parte do programa está a identificação e o mapeamento dos botões do *joystick.* Ao conectar o dispositivo de controle ao computador, no caso o *joystick*, obtém-se na pasta de Gerenciador de Dispositivo um número referente ao fabricante e ao modelo do *joystick*. Estes números foram inseridos no programa e relacionados ao mesmo. Deste modo, ao conectar outro dispositivo teríamos que realizar o mesmo procedimento, pois cada dispositivo de controle possui um número diferente.

O *joystick*, quando é utilizado, emite coordenadas nos eixos X e Y. Essas coordenadas são enviadas ao Delphi na forma de um conjunto de *Bytes* que devem ser mapeados. No mapeamento é criada uma variável para receber estes *Bytes* e poder utilizá-las posteriormente.

Depois de mapear os botões deu-se início à conversão dos dados recebidos do *joystick*. Isto foi necessário porque quando o *joystick* não está deflexionado, ou seja, sem ação, recebe-se na variável do programa o valor de 128. Se não houvesse esta conversão de dados, mesmo quando o joystick estivesse parado o robô estaria em movimento. No código 01 a seguir está a conversão.

```
// ADEQUAÇÃO VALOR X1
  if valx1=128 then
    begin
    xv:=1;
    end;
  if valx1=0 then
    begin
    xv:=-127;
    end;
  if valx1<128 then
    if valx1>0 then
```

```
      begin
      xv:=-(128-valx1);
      end;
  if valx1>128 then
    if valx1<=255 then
      begin
      xv:=valx1-128;
      end;
```

CÓDIGO 1 - Adequação dos valores de x1

Ainda nas adequações de coordenadas, citamos a transformação dos limites quadrados para limites circulares. Esta transformação se deu em função da diferença de velocidades em que o robô apresentaria ao deflexionar o joystick no limite máximo. A resultante das coordenadas nos limites quadrados não é o mesmo dos limites circulares, porque num círculo todas as retas que partem do centro até o limite são iguais, no caso o raio, como mostra a Figura 20. Ao aplicarmos esta transformação, conforme Código 02, padronizamos a velocidade nos limites máximos independente do ângulo.

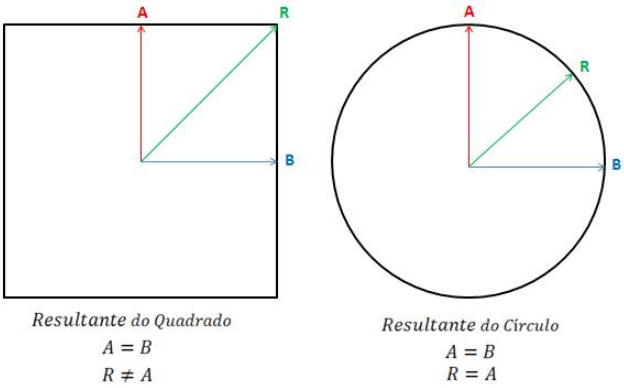


FIGURA 20 - Comparação de limites entre quadrados e círculos

```
// Mapeamento Quadrado para Circulo
  xvl:=yv*sqrt(1-((xv/127)*(xv/127))/2);
  yvl:=xv*sqrt(1-((yv/127)*(yv/127))/2);
```

CÓDIGO 2 - Mapeamento quadrado para círculo

O controle automático do robô permite apenas trajetos simples como quadrados e retângulos. Primeiramente determinou-se o tamanho do lado do quadrado ou então os lados do retângulo e na sequência clica-se em “Iniciar Trajetória”, como pode ser visto no item 13 da Figura 21. Os comandos para esta função encontram-se nos Códigos 03 e 04. Na interface também há a possibilidade de escolher trajetos como círculo e elipse, porém eles ainda não foram configurados.

```
2:  // Quadrado
    begin
       contatempo:=contatempo+1;
       if (contatempo>=0) and
       (contatempo<=2*StrToInt(lado.Text)) then //PASSO1
       begin
          xvl:=126; yvl:=0;
       end;
       if (contatempo>2*StrToInt(lado.Text)) and
       (contatempo<=4*StrToInt(lado.Text)) then //PASSO2
       begin
          xvl:=0; yvl:=126;
       end;
       if (contatempo>4*StrToInt(lado.Text)) and
       (contatempo<=6*StrToInt(lado.Text)) then //PASSO3
       begin
          xvl:=-126; yvl:=0;
       end;
       if (contatempo>6*StrToInt(lado.Text)) and
       (contatempo<=8*StrToInt(lado.Text)) then //PASSO4
       begin
          xvl:=0; yvl:=-126;
       end;
```

```
      if (contatempo>8*StrToInt(lado.Text)) then // FIM
      begin
         xvl:=0; yvl:=0;
         contatempo:=0; flagbotao:=0;
         initraje.Caption:='Iniciar Trajetória';
      end;
   end;
```

CÓDIGO 3 - Comando para quadrado

```
3:  // Retangulo
    begin
       contatempo:=contatempo+1;
       // PASSO 1
       if (contatempo>=0) and
       (contatempo<=2*StrToInt(lado1.Text)) then
       begin
          xvl:=126; yvl:=0;
       end;
       // PASSO 2
       if (contatempo>2*StrToInt(lado1.Text)) and
       contatempo<=(2*StrToInt(lado1.Text)+
       2*StrToInt(lado2.Text))) then
       begin
          xvl:=0; yvl:=126;
       end;
       // PASSO 3
       if (contatempo>(2*StrToInt(lado1.Text)+
       2*StrToInt(lado2.Text))) and
       (contatempo<=(4*StrToInt(lado1.Text)+
       2*StrToInt(lado2.Text))) then
       begin
          xvl:=-126; yvl:=0;
       end;
       // PASSO 4
       if (contatempo>(4*StrToInt(lado1.Text)+
       2*StrToInt(lado2.Text))) and
       (contatempo<=(4*StrToInt(lado1.Text)+
       4*StrToInt(lado2.Text))) then
       begin
          xvl:=0; yvl:=-126;
       end;
```

```
    // FIM
    if (contatempo>(4*StrToInt(lado1.Text)+
    4*StrToInt(lado2.Text))) then
    begin
       xvl:=0; yvl:=0;
       contatempo:=0; flagbotao:=0;
       initraje.Caption:='Iniciar Trajetória';
    end;
end;
```

CÓDIGO 4 - Comando para retângulo

Em seguida, introduziram-se as equações da cinemática que, juntamente com as coordenadas emitidas pelo *joystick*, permitem determinar o *PWM* (do inglês, *Pulse Width Modulate*) de cada motor. O PWM tem a função de controlar a velocidade dos motores de corrente contínua através da modulação da largura do pulso [OLIVEIRA, 2006]. As equações encontram-se no Código 05.

```
//Equações da Cinemática
motor0:=trunc(-Sin(PI/3)*xvl+0.5*yvl);
motor1:=trunc(-yvl);
motor2:=trunc(Sin(PI/3)*xvl+0.5*yvl);
```

CÓDIGO 5 - Equações da cinemática

Para conseguir a rotação foi preciso implementar um comando nos botões 5 e 6. Quando pressionado o botão 5 os motores giram todos no mesmo sentido, fazendo com que o robô gire em torno do seu próprio eixo no sentido anti-horário. E quando pressionado botão 6 ocorre o processo inverso. O Código 06 mostra este comando.

```
// Rotação_Anti-Horário
if bt5 then
begin
  motor0:=50;
  motor1:=50;
  motor2:=50;
end;
// Rotação_Horário
if bt6 then
begin
  motor0:=-50;
  motor1:=-50;
  motor2:=-50;
end;
```

CÓDIGO 6 – Comando de rotação

Na sequência identificaram-se as direções dos motores. Nesta etapa, para cada combinação de sentido de rotação dos motores foi atribuído uma letra e armazenado numa variável, que, posteriormente, é enviada ao Arduino. Conforme Código 07, observam-se as especificações das combinações dos sentidos de rotação dos motores.

```
// Direção dos Motores na posição zero do buffer buf[0]
// 000
if (motor0>=0) and (motor1>=0) and (motor2>=0) then
begin
  buf[0]:='A';
end;
// 001
if (motor0>=0) and (motor1>=0) and (motor2<0) then
begin
  buf[0]:='B';
end;
// 010
if (motor0>=0) and (motor1<0) and (motor2>=0) then
begin
  buf[0]:='C';
end;
```

```
// 011
if (motor0>=0) and (motor1<0) and (motor2<0) then
begin
  buf[0]:='D';
end;
// 100
if (motor0<0) and (motor1>=0) and (motor2>=0) then
begin
  buf[0]:='E';
end;
// 101
if (motor0<0) and (motor1>=0) and (motor2<0) then
begin
  buf[0]:='F';
end;
// 110
if (motor0<0) and (motor1<0) and (motor2>=0) then
begin
  buf[0]:='G';
end;
// 111
if (motor0<0) and (motor1<0) and (motor2<0) then
begin
  buf[0]:='H';
end;
```

CÓDIGO 7 - Combinações com os sentidos de rotação dos motores

Nas últimas linhas do programa configurou-se a parte responsável pelo envio de dados ao Arduino. Os valores das variáveis contendo o PWM e a direção dos motores foram armazenados nos *buffers* de saída. No total foram utilizados quatro *buffers*, sendo um para a direção e o restante para o PWM dos motores, como mostrado no Código 08.

```
buf[1]:=char(abs(motor0));
buf[2]:=char(abs(motor1));
buf[3]:=char(abs(motor2));
```

CÓDIGO 8 - *Buffers* com o PWM dos motores

A interface de comando também foi feita no Delphi. Nela é possível visualizar as imagens da câmera, o nível de carga das baterias, determinar a trajetória automática, verificar o sentido de rotação do robô e observar quais botões estão sendo acionados no momento. Na sequencia, a Figura 21 especifica cada item da interface de comando.

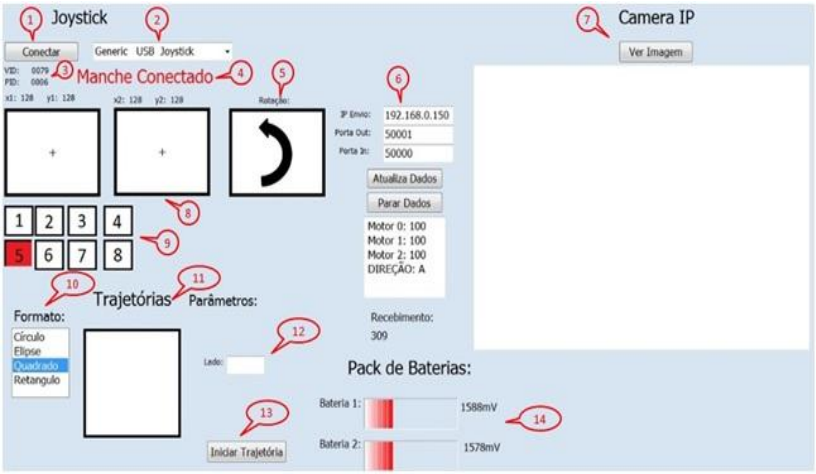


FIGURA 21 - Interface de comando

Legenda:

1. Botão para conectar *joystick;*
2. Nome do *joystick;*
3. Números referentes ao fabricante e ao modelo do *joystick*;
4. Sinaliza o manche conectado;
5. Visualização do sentido de rotação;
6. Números de IP e portas utilizadas do roteador;
7. Botão para inicializar a visualização das imagens da câmera;

8. Visualização da direção e da intensidade da deflexão do manche;
9. Botões auxiliares;
10. Trajetos automáticos possíveis;
11. Figura dos trajetos automáticos;
12. Escolha do tamanho do trajeto;
13. Botão para inicializar trajetória;
14. Visualização da carga das baterias.

O programa completo pode ser visualizado no Apêndice C.

5.2 Arduino

No início do programa os pinos da placa de comando dos motores são mapeados e declarados, conforme Código 09.

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h
#include <stdio.h>
#include <stdlib.h>

int M0 = 9;
int M1 = 5;
int M2 = 6;

int dir01 = 7;
int dir02 = 8;
int dir11 = 2;
int dir12 = 3;
int dir21 = 0;
int dir22 = 1;
```

CÓDIGO 9 - Mapeamento dos pinos da placa de comando

Na sequência, foi definida a direção do robô por meio dos caracteres recebidos no buffer [0], atribuindo diferentes sentidos de rotações aos motores, conforme observado no Código 10.

```
int packetSize = Udp.parsePacket();
  if(packetSize)
  {
    contador2=0;
    Udp.read(packetBuffer,UDP_TX_PACKET_MAX_SIZE);
          switch (packetBuffer[0]) {
    case 'A':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
    break;
    case 'B':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
    break;
    case 'C':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
    break;
    case 'D':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
          digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
    break;
    case 'E':
          digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
    break;
    case 'F':
          digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
    break;
    case 'G':
          digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
          digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
    break;
```

```
case 'H':
      digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
      digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
      digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
break;
}
```

CÓDIGO 10 - Definição dos sentidos de rotação dos motores

Após a direção, verificou-se o PWM dos motores através dos buffers [1], [2] e [3], mostrado no Código 11.

```
analogWrite(M0,2*packetBuffer[1]);
analogWrite(M1,2*packetBuffer[2]);
analogWrite(M2,2*packetBuffer[3]);
```

CÓDIGO 11 - PWM dos motores

No final do programa implementou-se uma maneira para evitar colisões e desperdício de energia. Nesta parte há um contador que tem a função de monitorar o recebimento de dados, se o Arduino ficar algum tempo sem receber dados enviados pelo computador (este tempo é programado), ele envia “0” para o PWM fazendo os motores pararem. Este comando é mostrado no Código 12.

```
{
    contador2=contador2+1;
  if (contador2==100)
  {
    analogWrite(M0,0);
    analogWrite(M1,0);
    analogWrite(M2,0);
  }
}
```

CÓDIGO 12 - Comando de segurança

O programa do Arduino inteiro encontra-se no Apêndice D.

5.3 Testes de Funcionamento

Os testes iniciais de funcionamento foram importantes para verificar os pontos que deveriam ser corrigidos e melhorados no robô. Nesses primeiros testes os objetivos iniciais foram conferir a parte elétrica para depois analisar a comunicação, dando mais atenção aos sinais do roteador. Em seguida analisaram-se os movimentos do robô e o sincronismo com os comandos do *joystick.* Por fim, verificou-se o funcionamento da câmera.

O primeiro ponto a ser melhorado foi referente ao sinal do roteador. Como o roteador estava situado dentro da carcaça de alumínio havia interferência devido a estrutura ser completamente fechada, fazendo com que o sinal atingisse apenas 5 m de distância, não alcançando o objetivo inicial determinado em projeto de no mínimo 10 m. Para resolver isso utilizamos uma antena maior e a deslocamos para a parte externa do robô, melhorando o sinal.

Outro ponto observado também é relacionado ao sinal do roteador, pois quando o robô saía fora do *hotspot* perdia-se o controle sobre ele e o último comando vindo do *joystick* permanecia no Arduino, gerando colisões e desperdício de carga das baterias. Neste caso, a situação foi resolvida na programação do Arduino, implementando um comando de segurança, como mostrado no capítulo de programação do Arduino.

No teste de movimento verificou-se a direção do robô e o sincronismo com os comandos do *joystick*, para isso tomou-se um lado da estrutura como referência e executou-se os movimentos em terrenos planos e sem obstáculos. Nesta etapa foi necessário inverter os fios do motor 1 para o robô deslocar-se corretamente.

Ainda na questão de movimentos havia a preocupação do torque do motor ser suficiente para movimentar o robô. No projeto tinha-se o requisito de peso máximo total de 10 kg, contudo, devido ao planejamento conseguiu-se um peso de 6,5 kg, evitando problemas de torque insuficiente.

O último componente a ser verificado foi a câmera IP, sendo testada em situações com boa iluminação e sem iluminação. Em ambos os casos obteve-se bons resultados sem maiores dificuldades.

5.4 Resultados Obtidos

Os resultados obtidos com a elaboração e execução do projeto do robô omnidirecional foram muitos satisfatórios. Isto porque, apesar do tempo ser relativamente curto, foi possível projetar, construir e testar os componentes.

No cronograma do TCC primeiramente tem-se o projeto com os devidos requisitos e especificações. Em seguida, adquiriu-se os componentes e materiais necessários para iniciar a fabricação do robô, passando pela usinagem e depois pela montagem dos componentes mecânicos e eletrônicos. Nesta etapa ainda ocorreu a programação da interface e dos comandos dos motores. Por fim, deu-se início aos testes de funcionamento. Paralelo a tudo isso tem-se a elaboração do TCC escrito.

O cronograma foi um pouco prejudicado devido ao atraso na entrega das rodas omnidirecionais, pois foram adquiridas no mercado internacional. Isto também gerou atrasos na fabricação o que acabou encurtando o tempo destinado aos testes do robô. Por isso, a determinação de limites como autonomia, tipos de terrenos aplicáveis e alcance de sinal de roteador não foram realizados.

6 CONSIDERAÇÕES FINAIS

6.1 Conclusão

O tema apresentado neste TCC descreveu o projeto e a construção de um robô móvel com rodas omnidirecionais, controlado por joystick e que fornece imagens em tempo real, contando ainda, com a possibilidade de movimentos automáticos simples.

Atualmente, a aplicação de robôs omnidirecionais está relacionada principalmente ao futebol de robôs e ao transporte de cargas, mas futuramente, em conjunto com o fornecimento de imagens, ele pode ser implementado em várias em tarefas de inspeção e reconhecimento de locais de difícil acesso, devido à grande mobilidade que possui.

O principal objetivo deste trabalho foi a fazer a integração dos assuntos abordados durante todo o curso de Mecatrônica Industrial. Na execução deste trabalho tivemos a oportunidade de estudar e praticar mais detalhadamente a parte de programação, assim como a usinagem e o modelamento em software 3D.

A programação em Pascal foi utilizada no software Delphi e baseada na linguagem “C”, no microcontrolador Arduino.

Na usinagem, as fresadoras convencionais e CNC foram as máquinas mais requisitadas, e na modelação citamos os sistemas CAD e CAM.

O maior desafio na elaboração do robô foi fazer o controle dos movimentos das rodas. Isto porque, deve haver muito sincronismo no acionamento delas, fazendo-se necessário um estudo aprofundado da cinemática e da sua implementação na programação.

Comparando os resultados obtidos com os objetivos do projeto chegamos a conclusão que os resultados foram expressivos e satisfatórios. As principais funções do robô como movimentos manuais, automáticos e fornecimento de imagens

foram concluídas com sucesso. Entretanto, devido ao tempo, não foi possível realizar testes mais aprofundados para determinar limites de funcionamento, pois ainda temos apenas estimativas destes limites. Desta maneira, estes testes ficam como um possível complemento para estudos posteriores.

6.2 Trabalhos Futuros Sugeridos

Para o futuro, fica a possibilidade de se aprimorar o modo de trajetória automática do robô omnidirecional. Isto pode ser feito com a implantação de módulo GPS e sensores ultrassônicos.

Visando uma navegação autônoma, poderá ser implementado um módulo com Sistema de Posicionamento Global, conhecido por GPS. Isto auxiliaria no aperfeiçoamento do trajeto a ser percorrido pelo robô no modo automático. Este módulo já está instalado, porém não está configurado.

No intuito de evitar colisões, poderá ser adicionado sensores capazes de detectar obstáculos durante o percurso. Além dos componentes, a programação deverá ser modificada para suportar estas novas funções.

Ainda como sugestões de trabalhos futuros ficam os testes para determinar os limites de funcionamento do robô como: tempo real de duração das baterias, alcance de sinal do roteador e peso máximo suportado pelos motores.

Todos estes recursos mencionados ajudarão a aperfeiçoar o controle do robô omnidirecional, aumentando o grau de tecnologia aplicado.

REFERÊNCIAS

BLANCHARD, B. S.; FABRYCKY, W. J. **Systems Engineering and Analysis** / B.S. Blanchard; W. J. Fabrycky Prentice - Hall, 1990.

CANTÙ, Marco. **Dominando o Delphi 5 – "A Bíblia"**/ Marco Cantù, Tradução João E. N. Tortello; revisão técnica Álvaro Rodrigues Antunes e Marcos Jorge. São Paulo: Makron Books, 2000.

FETT, Marcos. **Análise dos sistemas CAD/CAM e suas aplicações na indústria náutica de embarcações de receio** / Marcos Fett. Trabalho de Conclusão (Design Industrial). Universidade do Estado de Santa Catarina, 2010.

HAYKIN, Simon. **Sistemas modernos de comunicações wireless** / Simon Haykin, Michael Moher. Tradução de Glayson Eduardo de Figueiredo e José Lucimar do Nascimento. Porto Alegre: Bookman, 2008.

HARBS, Eduardo. **CNC-C²: Um Controlador Aderente às Normas ISO 14649 e IEC 61499** / Eduardo Harbs. Dissertação (Mestrado). Universidade do Estado de Santa Catarina, 2012.

HONDA, Flavio. **Motores de Corrente Contínua – Guia Rápido para uma Especificação Rápida**/ Flavio Honda - Ed. 01, 2006.

MCROBERTS, Michael. **Arduino Básico**/ Michael McRoberts, Tradução de Rafael Zanolli. São Paulo: Novatec Editora, 2011.

NASCIMENTO, Tiago Pereira do. **Controle de Trajetória de Robôs Móveis Omni-direcionais: Uma abordagem**

multivariável / Tiago Pereira do Nascimento. Dissertação (Mestrado). Universidade Federal da Bahia, 2009.

OLIVEIRA, André Schneider de. **Sistemas Embarcados: Hardware e Firmware na Prática** / André Schneider de Oliveira e Fernando de Souza de Andrade. Ed. 01.São Paulo: Érica, 2006.

PINHEIRO, Hayanne Soares; NASCIMENTO, José F.L.; QUEIROS-NETO, José P. **Simulador de cinemática direta de um robô didático (robix) em ambiente Labview**. Instituto Federal de Educação, Ciência, e Tecnologia do Amazonas. Amazonas, 2010.

PINHEIRO, Daniel. **Câmeras IP permitem vigiar a casa pela Internet; veja como funciona**. Disponível em <http://tecnologia.uol.com.br/ultnot/2006/12/13/ult2870u215.jhtm>. Acesso em 10 jun. 2013.

RIBEIRO, David Edson. **Adaptação de um sistema microcontrolado de detecção e monitoramento de corrente de fuga para inserir comunicação sem fio** / David Edson Ribeiro. Trabalho de Conclusão de Curso (Engenharia da Computação). Universidade de Pernambuco, 2008.

RIBEIRO, F.; MOUTINHO, I.; SILVA, P.; FRAGA, C.; PEREIRA, N.; **Three Omnidirectional Wheels Control on a Mobile Robot** / F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, N. Pereira. Universidade do Minho, 2004.

RIBEIRO, Marcos Valério; CUNHA, Elias Alves da. **Usinagem da Liga de Alumínio ASTM AA7050 por Torneamento** / Marcos Valério Ribeiro; Elias Alves da Cunha. Departamento de Materiais e Tecnologia, UNESP. São Paulo, 2004.

RITA, Tiago Daniel Teixeira, **Controle de Helicópteros de Aeromodelismo** / Tiago Daniel Teixeira. Dissertação (Mestrado). Universidade Técnica de Lisboa, 2009

ROCHA, Juliana Almeida. **Estudo de Operação de Movimentos do Veículo Submersível VSI-02 Via Joystick** /Juliana Almeida Rocha. Trabalho de Conclusão de Curso (Engenharia de Controle e Automoção). Universidade Federal de Ouro Preto, 2012.

ROSA, Luis Carlos; SIQUEIRA, Cláudio. **Oficina Mecânica para Automação – Prática 02: Fresadora e o Processo de Fresamento** / Luis Carlos Rosa; Cláudio Siqueira. Universidade Estadual Paulista, 2007.

SECHHI, Humberto. **Uma Introdução aos Robôs Móveis**/ Humberto Sechhi. Tradução de Cynthia Netto de Almeida e Felipe Nascimento Martins. Monografia. Universidade Nacional de San Juan – UNSJ: Argentina, 2008.

SOBRINHO, Júlio César Lins Barreto. **Controle Preditivo de um Robô Omnidirecional com Compensação de Atrito** / Júlio César Lins Barreto Sobrinho. Dissertação (Mestrado). Universidade Federal da Bahia, 2011.

SOUZA, André João de. **Apostila: Processo de Fabricação por Usinagem - Parte 2** / André João de Souza. Universidade Federal do Rio Grande do Sul, 2011.

SPURGEON, Charles E. **Ethernet: o guia definitivo** / Charles E. Spurgeon. Tradução de Daniel Vieira. Rio de Janeiro: Elsevier, 2000 – 2ª impressão.

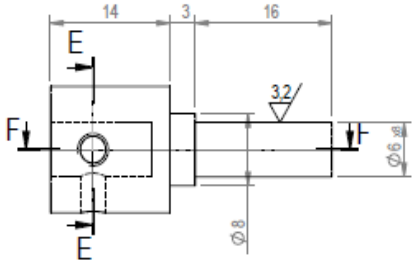
APÊNDICE A – Tabela Desenvolvimento de Soluções

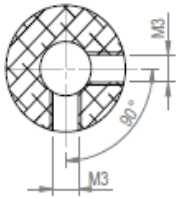
DESENVOLVIMENTO DE SOLUÇÕES PARA AS FUNÇÕES				
Requisitos:		**Soluções:**	**Vantagens:**	**Desvantagens:**
Movimentação	Rodas	Roda Omnidirecional	custo baixo, fácil instalação	dificil de encontrar
		Roda Mecanum	fácil instalação	dificil de encontrar, custo elevado
		Roda esférica	boa mobilidade	dificil instalação, difícil controle
	Motores	Motor CC	custo baixo, disponibilidade, baixa complexidade de controle	baixa precisão
		Motor CA		baixa precisão
		Servo	alta precisão	custo elevado
Mobilidade (Comunicação)		Bluetooth	custo baixo, baixo consumo de energia	baixo alcance, baixa capacidade de dados
		Wi-Fi	grande alcance, grande quantidade de dados	custo elevado, mais peças empregadas, alto consumo de energia
		Ronja	longo alcance, grande capacidade de dados	não pode haver qualquer interferência ou obstáculo entre emissor e receptor , custo elevado

	Infrared Data Association (IrDA) - Infravermelho	custo baixo	baixa capacidade de dados
Autonomia	Baterias Lipo	tamanho pequeno, duração média, leve	custo elevado
	Bateria selada	preço baixo, alta duração	pesado, tamanho grande
	Bateria alcalina (pilhas)	tamanho pequeno	baixa duração
Baixo Peso (Estrutura)	Estrutura de alumínio	leve, custo médio, alta disponibilidade, boa usinabilidade	
	Estrutura de policarbonato	leve, disponibilidade, boa usinabilidade	custo elevado, transparência
	Estrutura de titânio	leve	custo muito elevado, difícil usinagem
	Estrutura de madeira	leve, disponibilidade, boa usinabilidade	aspecto incompatível
Interface de comando	Delphi	programação simples, grande versatiidade	custo elevado
	Elipse Scada	programação simples	pouca versatilidade
Processamento	Arduino	programação simples	pouca versatilidade

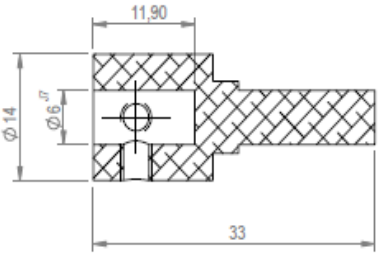
	CLP	programação simples	alto consumo de energia, custo muito elevado, tamanho grande
	Microprocessador (PIC)	programação media	necessário componentes adicionais
	Triângulo	fácil fabricação	espaços não utilizados sem
Formato da carcaça	Quadrado	fácil fabricação	mais motores empregados, necessária implentação de suspensão
	Círculo	compactação	difícil fabricação

APÊNDICE B – Desenhos Técnicos



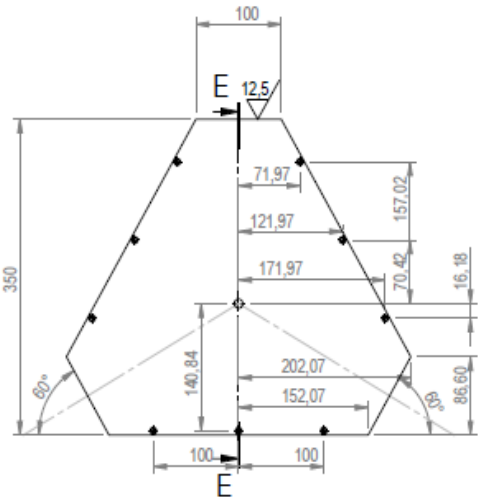


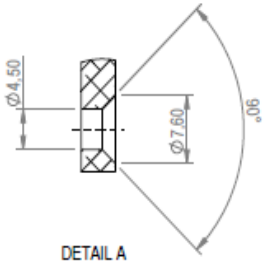
SEÇÃO E-E



SEÇÃO F-F

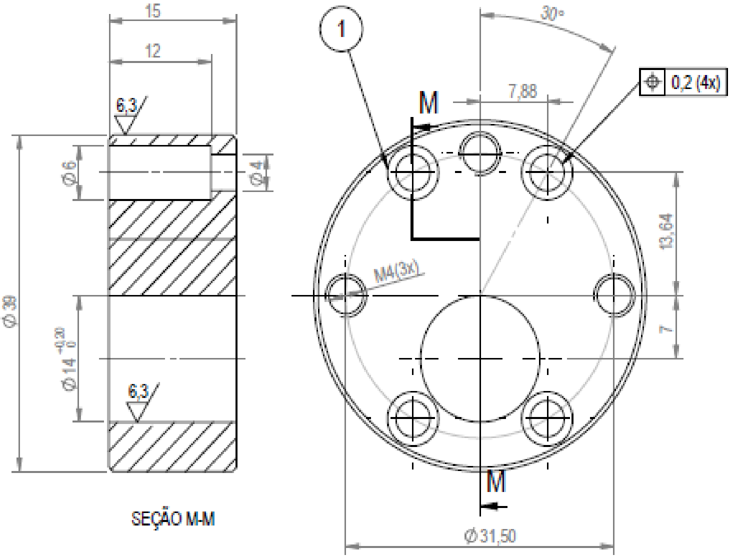
UDESC JOINVILLE	Fase: Projeto	Área:	Tolerância Geral: ±0,05	
Tel.47 3461 9200	Tratamento Térmico:	Dimensões: 14x33 mm	Obs.: 03	Projeto: Robô Omnidireicional
Data: 15/06/2013	Material: Calibração do Robô	Torno		Folha: 01
Esc.: 2:1	Denominação: Acoplamento da Roda		Código: RB-01	





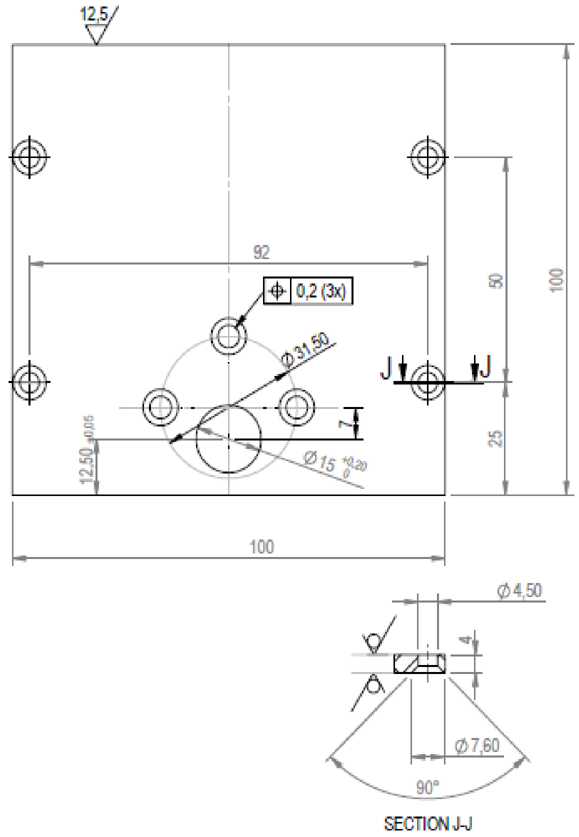
DETAIL A
SCALE 2 : 1

ISC JOINVILLE Tel.:47 3461 8600	Tratamento Geral: Pintura	Rugosidade Geral: R4 12,5	Área: 96843,46mm²	Tolerância Geral: ±0,1
Data: 30/08/2013	Tratamento: Confecção do Robô	Tret. Térmico:	Dimensões: 350x404x4 mm	Qtd.: 01
Esc.: 1:5	Denominação: Base			Código: RB-02

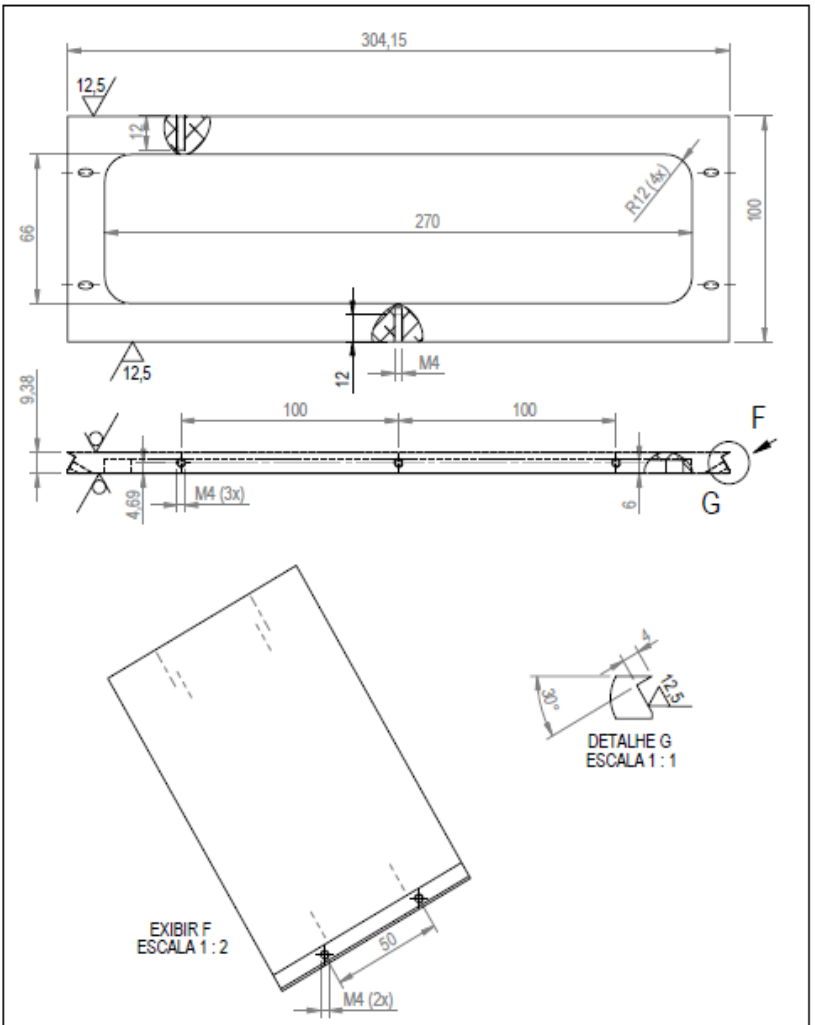


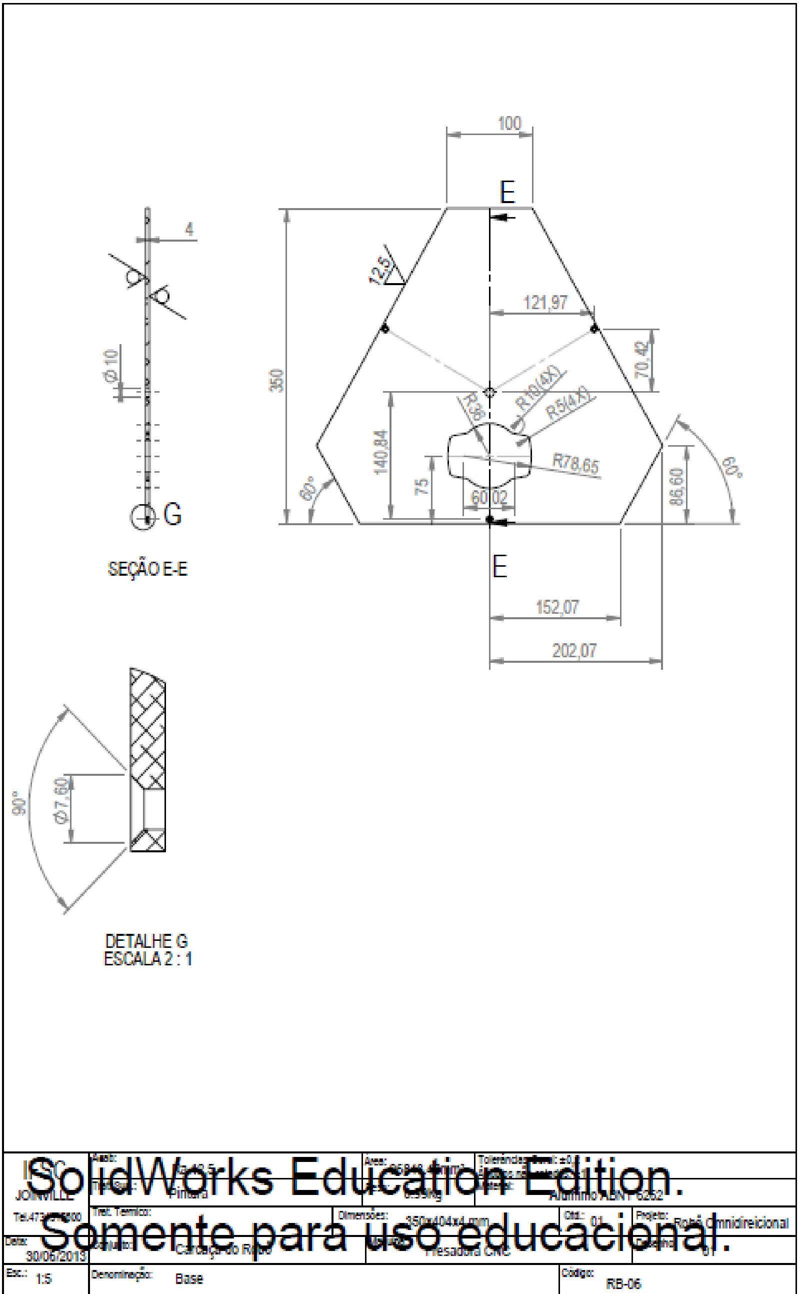
1 Alojamento p/ parafuso M3 (4x).

JOINVILLE	Acab.: Ra 6,3	Área:	Tolerância Geral: ±0,2
Tel.:	Trat. Sup.:		Material: Alumínio ABNT 6262
	Trat. Termico:	Dimensões: 39x15 mm	Qtd.: 03 / Projeto: Robô Omnidireicional
Data: 15/06/2013	Conjunto: Carcaça do Robô	Usinagem: Torno / Fresadora	Desenho: 01
Esc.: 2:1	Denominação: Espaçador		Código: RB-03



JOINVILLE	Acab.: Ra 12,5	Área: 10000 mm²	Tolerância Geral: ±0,1
Tel.: 47	Trat. Sup.: Pintura	0,16kg	Material: Alumínio ABNT 6262
	Trat. Térmico:	Dimensões: 100x100x4 mm	Qtd.: 03 — Projeto: Robô Omnidireicional
Data: 15/06/2013	Conjunto: Carcaça do Robô		
Esc.: 1:1	Denominação: Lateral Quadrada		Código: RB-04





APÊNDICE C – Programa da Interface em Delphi

```
unit usb_test;

interface

uses
  Windows, Messages, SysUtils,
Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, JvComponentBase,
JvHidControllerClass, StdCtrls,
ExtCtrls, jpeg,
  JvExExtCtrls, JvExtComponent,
JvItemsPanel, JvExControls,
JvPageList,
  JvNavigationPane, OleCtrls,
SHDocVw, Menus, IdUDPServer,
IdBaseComponent,
  IdComponent, IdUDPBase,
IdUDPClient, IdSocketHandle,
JvExComCtrls,
  JvProgressBar, ComCtrls;

type

  TReport = packed record
  ReportID: byte;
  Data: array [0..64] of byte;
end;

  TForm1 = class(TForm)
    HidCtl: TJvHidDeviceController;
    labelvid: TLabel;
    labelpid: TLabel;
    vidnumber: TLabel;
    pidnumber: TLabel;
    Label2: TLabel;
    escrita: TTimer;
    seljoystick: TComboBox;
    conectar: TButton;
    fundo1: TImage;
    cursor1: TImage;
    fundo2: TImage;
    cursor2: TImage;
    bt1on: TImage;
    bt1off: TImage;
    bt2on: TImage;
    bt2off: TImage;
    bt3on: TImage;
    bt3off: TImage;
    bt4on: TImage;
    bt4off: TImage;
    Label1: TLabel;
    Label3: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    valuex1: TLabel;
    valuey1: TLabel;
    valuex2: TLabel;
    valuey2: TLabel;
    conectjoy: TLabel;
    navegador: TWebBrowser;
    Label4: TLabel;
    Button1: TButton;
    MainMenu1: TMainMenu;
    Label5: TLabel;
    Label6: TLabel;
    Arquivo1: TMenuItem;
    Editar1: TMenuItem;
    Help1: TMenuItem;
    Fechar1: TMenuItem;
    About1: TMenuItem;
    bt5on: TImage;
    bt5off: TImage;
    bt6on: TImage;
    bt6off: TImage;
    bt7on: TImage;
    bt7off: TImage;
    bt8on: TImage;
    bt8off: TImage;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    ipnumber: TEdit;
    portout: TEdit;
    portin: TEdit;
    atualiza1: TButton;
```

```
    enviar: TMemo;
    Cliente: TIdUDPClient;
    Servidor: TIdUDPServer;
    enviar_dados: TButton;
    shape: TShape;
    formato: TListBox;
    Label10: TLabel;
    raio: TEdit;
    lado: TEdit;
    eixomenor: TEdit;
    eixomaior: TEdit;
    lado2: TEdit;
    lado1: TEdit;
    lbraio: TLabel;
    lblado: TLabel;
    lbmaior: TLabel;
    lbmenor: TLabel;
    lblado2: TLabel;
    lblado1: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    lblcount: TLabel;
    battery_server: TIdUDPServer;
    tenbateria1: TLabel;
    battery_server2: TIdUDPServer;
    tenbateria2: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    tensaobar1:
TJvGradientProgressBar;
    tensaobar2:
TJvGradientProgressBar;
    initraje: TButton;
    rot_0: TImage;
    rot_hor: TImage;
    rot_ahor: TImage;
    Rotação: TLabel;
    procedure FormCreate(Sender:
TObject);
    procedure FormDestroy(Sender:
TObject);
    procedure
HidCtlDeviceChange(Sender:
TObject);
    function HidCtlEnumerate(HidDev:
TJvHidDevice; const Idx: Integer):
Boolean;
    procedure btnlerClick(Sender:
TObject);
    procedure conectarClick(Sender:
TObject);
    procedure Button1Click(Sender:
TObject);
    procedure Fechar1Click(Sender:
TObject);
    procedure atualiza1Click(Sender:
TObject);
    procedure envia_dados(Sender:
TObject);
    procedure leitura(AThread:
TIdUDPListenerThread; AData:
TBytes;
      ABinding: TIdSocketHandle);
    procedure escritaTimer(Sender:
TObject);
    procedure formatoClick(Sender:
TObject);
    procedure leiturabaterias(AThread:
TIdUDPListenerThread; AData:
TBytes;
      ABinding: TIdSocketHandle);
    procedure
leiturabaterias2(AThread:
TIdUDPListenerThread; AData:
TBytes;
      ABinding: TIdSocketHandle);
    procedure initrajeClick(Sender:
TObject);
  private
    { Private declarations }
    DevList: TList;
  public
    { Public declarations }
  end;

  TJoyThread = class(TThread)
  private
  public
    procedure Execute; override;
    procedure HandleJoyData;
  end;

var
  Form1: TForm1;
  TheDev: TJvHidDevice;
  HidData : TReport;
```

```
  written : DWORD;
  flagdados: Integer=0;
  flagbotao: Integer=0;
  formatraje: Integer=0;
  passo: Integer=0;
  contatempo: Integer=0;
  count: Integer=0;
  indice : array[0..10] of Integer;
  joystick: String;
  valx1: Integer =128; valy1:
Integer=128; valx2: Integer=128;
valy2: Integer=128;
  contadorrec: Integer =0;
  bt5: boolean; bt6: boolean;

implementation
{$R *.dfm}

Const
  VIDmanche = $0079;   // Put in your
matching VendorID
  PIDmanche = $0006;  // Put in your
matching ProductID

  VIDgamepad = $0E8F;   // Put in
your matching VendorID
  PIDgamepad = $1140;  // Put in your
matching ProductID


procedure
TJoyThread.HandleJoyData;
var calx1: double; caly1: double;
calx2: double; caly2: double;
bt1: boolean; bt2: boolean; bt3:
boolean; bt4: boolean; bt7: boolean;
bt8: boolean;
begin

if joystick='manche' then
begin
  valx1:=HidData.Data[0];
valy1:=HidData.Data[1];
if  ((HidData.Data[5]=$0F) OR
(HidData.Data[5]=$1F) OR
(HidData.Data[5]=$2F) OR
(HidData.Data[5]=$3F) OR
(HidData.Data[5]=$4F) OR
(HidData.Data[5]=$5F) OR
(HidData.Data[5]=$6F) OR
(HidData.Data[5]=$7F) OR
(HidData.Data[5]=$8F) OR
(HidData.Data[5]=$9F) OR
(HidData.Data[5]=$AF) OR
(HidData.Data[5]=$BF) OR
(HidData.Data[5]=$CF) OR
(HidData.Data[5]=$DF) OR
(HidData.Data[5]=$EF) OR
(HidData.Data[5]=$FF) ) then
//Sem Botao
  begin
    valx2:=128; valy2:=128;
  end
else
  begin
    if  (HidData.Data[5] AND $04 =
$04) then     //baixo
      begin
        valx2:=128; valy2:=255;
      end;
    if  (HidData.Data[5] AND $02 =
$02) then     //direita
      begin
        valx2:=255; valy2:=128;
      end;
    if  (HidData.Data[5] AND $06 =
$06) then     //esquerda
      begin
        valx2:=0; valy2:=128;
      end;
    if  ((HidData.Data[5]=$00) OR
(HidData.Data[5]=$10) OR
(HidData.Data[5]=$20) OR
(HidData.Data[5]=$30) OR
(HidData.Data[5]=$40) OR
(HidData.Data[5]=$50) OR
(HidData.Data[5]=$60) OR
(HidData.Data[5]=$70) OR
(HidData.Data[5]=$80) OR
(HidData.Data[5]=$90) OR
(HidData.Data[5]=$A0) OR
(HidData.Data[5]=$B0) OR
(HidData.Data[5]=$C0) OR
(HidData.Data[5]=$D0) OR
(HidData.Data[5]=$E0) OR
(HidData.Data[5]=$F0) ) then
//cima
      begin
```

```
      valx2:=128; valy2:=0;
    end;
  if  (HidData.Data[5] AND $01 =
$01) then     //nordeste
    begin
      valx2:=255; valy2:=0;
    end;
  if  (HidData.Data[5] AND $05 =
$05) then     //sudoeste
    begin
      valx2:=0; valy2:=255;
    end;
  if  (HidData.Data[5] AND $03 =
$03) then     //sudeste
    begin
      valx2:=255; valy2:=255;
    end;
  if  (HidData.Data[5] AND $07 =
$07) then     //sudeste
    begin
      valx2:=0; valy2:=0;
    end;

end;

if  (HidData.Data[5] AND $40 = $40)
then     //BOTAO 1
bt1:=true
else
bt1:=false;

if  (HidData.Data[5] AND $80 = $80)
then     //BOTAO 2
bt2:=true
else
bt2:=false;

if  (HidData.Data[6] AND $01 = $01)
then     //BOTAO 3
bt3:=true
else
bt3:=false;

if  (HidData.Data[6] AND $02 = $02)
then     //BOTAO 4
bt4:=true
else
bt4:=false;

if  (HidData.Data[5] AND $20 = $20)
then     //BOTAO 5
bt5:=true
else
bt5:=false;

if  (HidData.Data[5] AND $10 = $10)
then     //BOTAO 6
bt6:=true
else
bt6:=false;

bt7:=false;
bt8:=false;
end;

if joystick='gamepad' then
begin
  valx1:=HidData.Data[0];
valy1:=HidData.Data[3];
valx2:=HidData.Data[6];
valy2:=HidData.Data[5];

if  (HidData.Data[1] AND $01 = $01)
then     //BOTAO 1
bt1:=true
else
bt1:=false;

if  (HidData.Data[1] AND $02 = $02)
then     //BOTAO 2
bt2:=true
else
bt2:=false;

if  (HidData.Data[1] AND $04 = $04)
then     //BOTAO 3
bt3:=true
else
bt3:=false;

if  (HidData.Data[1] AND $08 = $08)
then     //BOTAO 4
bt4:=true
else
bt4:=false;

if  (HidData.Data[1] AND $10 = $10)
then     //BOTAO 5
```

```
bt5:=true
else
bt5:=false;

if  (HidData.Data[1] AND $20 = $20)
then     //BOTAO 6
bt6:=true
else
bt6:=false;

if  (HidData.Data[1] AND $40 = $40)
then     //BOTAO 7
bt7:=true
else
bt7:=false;

if  (HidData.Data[1] AND $80 = $80)
then     //BOTAO 8
bt8:=true
else
bt8:=false;


end;

if  bt1 then     //BOTAO 1
  begin
    form1.bt1on.Visible:=true;
    form1.bt1off.Visible:=false;
  end
else
  begin
    form1.bt1on.Visible:=false;
    form1.bt1off.Visible:=true;
end;

if  bt2 then     //BOTAO 2
  begin
    form1.bt2on.Visible:=true;
    form1.bt2off.Visible:=false;
  end
else
  begin
    form1.bt2on.Visible:=false;
    form1.bt2off.Visible:=true;
end;

if  bt3 then     //BOTAO 3
  begin
    form1.bt3on.Visible:=true;
    form1.bt3off.Visible:=false;
  end
else
  begin
    form1.bt3on.Visible:=false;
    form1.bt3off.Visible:=true;
end;

if  bt4 then     //BOTAO 4
  begin
    form1.bt4on.Visible:=true;
    form1.bt4off.Visible:=false;
  end
else
  begin
    form1.bt4on.Visible:=false;
    form1.bt4off.Visible:=true;
end;

if  bt5 then     //BOTAO 5
  begin
    form1.bt5on.Visible:=true;
    form1.bt5off.Visible:=false;
    form1.rot_ahor.Visible:=true;
  end
else
  begin
    form1.bt5on.Visible:=false;
    form1.bt5off.Visible:=true;
    form1.rot_ahor.Visible:=false;
end;

if  bt6 then     //BOTAO 6
  begin
    form1.bt6on.Visible:=true;
    form1.bt6off.Visible:=false;
    form1.rot_hor.Visible:=true;
  end
else
  begin
    form1.bt6on.Visible:=false;
    form1.bt6off.Visible:=true;
    form1.rot_hor.Visible:=false;
end;


if  bt7 then     //BOTAO 7
  begin
```

```
    form1.bt7on.Visible:=true;
    form1.bt7off.Visible:=false;
  end
else
  begin
    form1.bt7on.Visible:=false;
    form1.bt7off.Visible:=true;
end;

if  bt8 then     //BOTAO 8
  begin
    form1.bt8on.Visible:=true;
    form1.bt8off.Visible:=false;
  end
else
  begin
    form1.bt8on.Visible:=false;
    form1.bt8off.Visible:=true;
end;


// muda cursor x1
calx1:=(136/255)*valx1+form1.fundo1
.Left+6;
form1.cursor1.Left:=Round(calx1);

// muda cursor y1
caly1:=(108/255)*valy1+form1.fundo1
.Top+6;
form1.cursor1.Top:=Round(caly1);

// muda cursor x2
calx2:=(136/255)*valx2+form1.fundo2
.Left+6;
form1.cursor2.Left:=Round(calx2);

// muda cursor y1
caly2:=(108/255)*valy2+form1.fundo2
.Top+6;
form1.cursor2.Top:=Round(caly2);


form1.valuex1.Caption:=InttoStr(valx1
);

form1.valuey1.Caption:=InttoStr(valy1
);

form1.valuex2.Caption:=InttoStr(valx2
);

form1.valuey2.Caption:=InttoStr(valy2
);

end;


procedure TJoyThread.Execute;
begin
while not Terminated do
begin
  HidData.ReportID:=0;
  TheDev.ReadFile(HidData,
TheDev.Caps.InputReportByteLength
, Written);
  Synchronize(HandleJoyData);
end;
end;


procedure
TForm1.atualiza1Click(Sender:
TObject);
begin
 Cliente.Host := IPnumber.Text;
 Cliente.Port :=
StrToIntDef(portout.Text, 0 );
 Servidor.DefaultPort :=
StrToIntDef(portin.Text, 0 );
end;

procedure
TForm1.btnlerClick(Sender: TObject);
var ThreadJoy: TJoyThread;
begin
 ThreadJoy:=
TJoyThread.Create(True);
 ThreadJoy.FreeOnTerminate:=True;
 ThreadJoy.Resume;
end;


procedure
TForm1.Fechar1Click(Sender:
TObject);
```

```
begin
Close;
end;

procedure
TForm1.formatoClick(Sender:
TObject);
begin
formatraje:=formato.ItemIndex;
if formato.ItemIndex=0 then
begin
  shape.Shape:=stCircle;
  lbraio.Visible:=true;
raio.Visible:=true;
  lbmaior.Visible:=false;
eixomaior.Visible:=false;
  lbmenor.Visible:=false;
eixomenor.Visible:=false;
  lblado.Visible:=false;
lado.Visible:=false;
  lblado1.Visible:=false;
lado1.Visible:=false;
  lblado2.Visible:=false;
lado2.Visible:=false;
end;

if formato.ItemIndex=1 then
begin
  shape.Shape:=stEllipse;
  lbraio.Visible:=false;
raio.Visible:=false;
  lbmaior.Visible:=true;
eixomaior.Visible:=true;
  lbmenor.Visible:=true;
eixomenor.Visible:=true;
  lblado.Visible:=false;
lado.Visible:=false;
  lblado1.Visible:=false;
lado1.Visible:=false;
  lblado2.Visible:=false;
lado2.Visible:=false;
end;

if formato.ItemIndex=2 then
begin
  shape.Shape:=stSquare;
  lbraio.Visible:=false;
raio.Visible:=false;
  lbmaior.Visible:=false;
eixomaior.Visible:=false;
  lbmenor.Visible:=false;
eixomenor.Visible:=false;
  lblado.Visible:=true;
lado.Visible:=true;
  lblado1.Visible:=false;
lado1.Visible:=false;
  lblado2.Visible:=false;
lado2.Visible:=false;
end;


if formato.ItemIndex=3 then
begin
  shape.Shape:=stRectangle;
  lbraio.Visible:=false;
raio.Visible:=false;
  lbmaior.Visible:=false;
eixomaior.Visible:=false;
  lbmenor.Visible:=false;
eixomenor.Visible:=false;
  lblado.Visible:=false;
lado.Visible:=false;
  lblado1.Visible:=true;
lado1.Visible:=true;
  lblado2.Visible:=true;
lado2.Visible:=true;
end;


end;


procedure
TForm1.FormCreate(Sender:
TObject);
begin
  initraje.Enabled:=false;
  lbraio.Visible:=true;
raio.Visible:=true;
  lbmaior.Visible:=false;
eixomaior.Visible:=false;
  lbmenor.Visible:=false;
eixomenor.Visible:=false;
  lblado.Visible:=false;
lado.Visible:=false;
  lblado1.Visible:=false;
lado1.Visible:=false;
```

```
  lblado2.Visible:=false;
lado2.Visible:=false;

  formato.ItemIndex:=0 ;
  shape.Shape:=stCircle;

  Cliente.Active:=True;
  Servidor.Active:=True;
  ipnumber.Text:='192.168.0.150';
  portout.Text:='50001';
  portin.Text:='50000';
  enviar.Lines.Clear;
  valuex1.Caption:='';
valuey1.Caption:='';
valuex2.Caption:='';
valuey2.Caption:='';
  bt1on.Enabled:=False;
bt2on.Enabled:=False;
bt3on.Enabled:=False;
bt4on.Enabled:=False;
  bt1off.Enabled:=True;
bt2off.Enabled:=True;
bt3off.Enabled:=True;
bt4off.Enabled:=True;
  bt5on.Enabled:=False;
bt6on.Enabled:=False;
bt7on.Enabled:=False;
bt8on.Enabled:=False;
  bt5off.Enabled:=True;
bt6off.Enabled:=True;
bt7off.Enabled:=True;
bt8off.Enabled:=True;
  cursor1.Left:=fundo1.Left+74;
cursor1.Top:=fundo1.Top+60;
  cursor2.Left:=fundo2.Left+74;
cursor2.Top:=fundo2.Top+60;
  escrita.Enabled:=False;
  HidCtl.Enumerate;
  seljoystick.Text:='Selecionar
Joystick';
end;

procedure
TForm1.FormDestroy(Sender:
TObject);
var
  I: Integer;
begin
  Cliente.Active:=False;
  Servidor.Active:=False;
  for I := 0 to DevList.Count-1 do
  begin
    TheDev := DevList.Items[I];
    HidCtl.CheckIn(TheDev);
  end;
  DevList.Free;
end;

procedure
TForm1.HidCtlDeviceChange(Sender
: TObject);
var
  I: Integer;
begin
  if DevList <> nil then
  begin
    for I := 0 to DevList.Count-1 do
    begin
      TheDev := DevList.Items[I];
      TheDev.Free;
    end;
    DevList.Clear;
  end
  else
  DevList := TList.Create;
  seljoystick.Clear;
  conectjoy.Caption:='';
  count:=0;
  seljoystick.Text:='Selecionar
Joystick';
  HidCtl.Enumerate;
end;

function
TForm1.HidCtlEnumerate(HidDev:
TJvHidDevice;
  const Idx: Integer): Boolean;
var
  Dev: TJvHidDevice;
begin
   if ((HidDev.Attributes.VendorID =
VIDmanche ) and
(HidDev.Attributes.ProductID =
PIDmanche))or
((HidDev.Attributes.VendorID =
VIDgamepad ) and
(HidDev.Attributes.ProductID =
PIDgamepad)) then
```

```
  begin

seljoystick.Items.Add(HidDev.Product
Name);
     indice[count]:=Idx;
     count:=count+1;
  end;
  HidCtl.CheckOutByIndex(Dev,Idx);
  DevList.Add(Dev);
  Result := True;
end;



procedure
TForm1.initrajeClick(Sender:
TObject);
begin
case flagbotao of
  0:
    begin
    initraje.Caption:='Parar Trajetória';
    flagbotao:=1;
    end;
  1:
    begin
    initraje.Caption:='Iniciar Trajetória';
    flagbotao:=0;
    end;
end;
end;


procedure TForm1.leitura(AThread:
TIdUDPListenerThread; AData:
TBytes;
  ABinding: TIdSocketHandle);
begin
 //    receber.Lines.Add( 'De: ' +
ABinding.PeerIP );
 //    receber.Lines.Add( 'Mensagem: '
+  StringOf( AData ) );

 //  receber.Lines.Add( 'Mensagem: ' +
Char(AData[0])+' '+Char(AData[1])+'
'+Char(AData[2])+' '+Char(AData[3])
);

 contadorrec:=contadorrec+1;
lblcount.Caption:=inttostr(contadorrec
);

 end;

procedure
TForm1.leiturabaterias(AThread:
TIdUDPListenerThread; AData:
TBytes;
  ABinding: TIdSocketHandle);
var
  tensao1: Integer;
begin
tensao1:=StrToInt(StringOf( AData ));
tensao1:=trunc(5000*tensao1/1023);
tenbateria1.Caption:=IntToStr(tensao
1)+'mV';
tensaobar1.Position:=tensao1;
// COLORAÇÂO DA PROGRESS
BAR DE ACORDO COM A TENSÂO
NA BATERIA
if (tensao1>0) and (tensao1<=1667)
then
begin
  tensaobar1.BarColorTo:=clRed;
end;
if (tensao1>1667) and
(tensao1<=3333) then
begin
  tensaobar1.BarColorTo:=clYellow;
end;
if (tensao1>3333) and
(tensao1<=5000) then
begin
  tensaobar1.BarColorTo:=clGreen;
end;

end;

procedure
TForm1.leiturabaterias2(AThread:
TIdUDPListenerThread; AData:
TBytes;
  ABinding: TIdSocketHandle);
var
  tensao2: Integer;
begin
tensao2:=StrToInt(StringOf( AData ));
```

```
tensao2:=trunc(5000*tensao2/1023);
tenbateria2.Caption:=IntToStr(tensao
2)+'mV';
tensaobar2.Position:=tensao2;
// COLORAÇÂO DA PROGRESS
BAR DE ACORDO COM A TENSÂO
NA BATERIA
if (tensao2>0) and (tensao2<=1667)
then
begin
  tensaobar2.BarColorTo:=clRed;
end;
if (tensao2>1667) and
(tensao2<=3333) then
begin
  tensaobar2.BarColorTo:=clYellow;
end;
if (tensao2>3333) and
(tensao2<=5000) then
begin
  tensaobar2.BarColorTo:=clGreen;
end;

end;

procedure
TForm1.Button1Click(Sender:
TObject);
begin
navegador.Navigate('192.168.0.160');
end;
```

```
procedure
TForm1.conectarClick(Sender:
TObject);
var ThreadJoy: TJoyThread;
begin
  if (seljoystick.Items.Count > 0) and
(seljoystick.ItemIndex >= 0) then
  begin
    TheDev :=
DevList.Items[indice[seljoystick.ItemIn
dex]];
    vidnumber.Caption :=
Format('%.4x',
[TheDev.Attributes.VendorID]);
    pidnumber.Caption :=
Format('%.4x',
[TheDev.Attributes.ProductID]);

  if ((TheDev.Attributes.VendorID =
VIDmanche ) and
(TheDev.Attributes.ProductID =
PIDmanche)) then
  begin
     joystick:='manche';
     conectjoy.Caption:='Manche
Conectado';
  end;
  if ((TheDev.Attributes.VendorID =
VIDgamepad ) and
(TheDev.Attributes.ProductID =
PIDgamepad)) then
  begin
     joystick:='gamepad';
     conectjoy.Caption:='Gamepad
Conectado';
  end;

    ThreadJoy:=
TJoyThread.Create(True);

ThreadJoy.FreeOnTerminate:=True;
    ThreadJoy.Resume;
  end;
end;
```

```
procedure
TForm1.envia_dados(Sender:
TObject);
begin

case flagdados of
  0:
    begin
    enviar_dados.Caption:='Parar
Dados';
    flagdados:=1;
    escrita.Enabled:=true;
    initraje.Enabled:=true;
```

```
    end;
  1:
    begin
    enviar_dados.Caption:='Enviar
Dados';
    flagdados:=0;
    escrita.Enabled:=false;
    initraje.Enabled:=false;
    initraje.Caption:='Iniciar Trajetória';
    flagbotao:=0; contatempo:=0;
    enviar.Lines.Clear;
    end;
end;

end;


procedure
TForm1.escritaTimer(Sender:
TObject);
var buf:array[0..3] of ansichar;
xv: integer; yv: integer; xvl: double;
yvl: double; motor0: integer; motor1:
integer; motor2: integer;
begin

// CONTROLE PELO JOYSTICK
if (flagbotao=0) then
begin
// ADEQUAÇÃO VALOR X1
  if valx1=128 then
    begin
    xv:=1;
    end;

  if valx1=0 then
    begin
    xv:=-127;
    end;

  if valx1<128 then
    if valx1>0 then
      begin
      xv:=-(128-valx1);
      end;

  if valx1>128 then
    if valx1<=255 then
      begin
      xv:=valx1-128;
      end;

// ADEQUAÇÃO VALOR Y1
  if valy1=128 then
    begin
    yv:=1;
    end;

  if valy1=0 then
    begin
    yv:=127;
    end;

  if valy1<128 then
    if valy1>0 then
      begin
      yv:=128-valy1;
      end;

  if valy1>128 then
    if valy1<=255 then
      begin
      yv:=-(valy1-128);
      end;

// Mapeamento Quadrado para
Circulo
  xvl:=yv*sqrt(1-((xv/127)*(xv/127))/2);
  yvl:=xv*sqrt(1-((yv/127)*(yv/127))/2);

end;

// CONTROLE PELA TRAJETÓRIA
AUTOMATICA
if (flagbotao=1) then
begin
case formatraje of
  0:  // Circulo
    begin
    end;
  1:  // Elipse
    begin
    end;
  2:  // Quadrado
    begin
      contatempo:=contatempo+1;
      if (contatempo>=0) and
```

```
(contatempo<=2*StrToInt(lado.Text))
then // PASSO 1
      begin
        xvl:=126; yvl:=0;
      end;
      if
(contatempo>2*StrToInt(lado.Text))
and

(contatempo<=4*StrToInt(lado.Text))
then // PASSO 2
      begin
        xvl:=0; yvl:=126;
      end;
      if
(contatempo>4*StrToInt(lado.Text))
and

(contatempo<=6*StrToInt(lado.Text))
then // PASSO 3
      begin
        xvl:=-126; yvl:=0;
      end;
      if
(contatempo>6*StrToInt(lado.Text))
and

(contatempo<=8*StrToInt(lado.Text))
then // PASSO 4
      begin
        xvl:=0; yvl:=-126;
      end;
      if
(contatempo>8*StrToInt(lado.Text))
then // FIM
      begin
        xvl:=0; yvl:=0;
        contatempo:=0; flagbotao:=0;
        initraje.Caption:='Iniciar
Trajetória';
      end;
   end;
 3:  // Retangulo
   begin
      contatempo:=contatempo+1;
      // PASSO 1
      if (contatempo>=0) and

(contatempo<=2*StrToInt(lado1.Text)
) then
      begin
        xvl:=126; yvl:=0;
      end;
      // PASSO 2
      if
(contatempo>2*StrToInt(lado1.Text))
and

(contatempo<=(2*StrToInt(lado1.Text
)+
      2*StrToInt(lado2.Text))) then
      begin
        xvl:=0; yvl:=126;
      end;
      // PASSO 3
      if
(contatempo>(2*StrToInt(lado1.Text)
+
      2*StrToInt(lado2.Text))) and

(contatempo<=(4*StrToInt(lado1.Text
)+
      2*StrToInt(lado2.Text))) then
      begin
        xvl:=-126; yvl:=0;
      end;
      // PASSO 4
      if
(contatempo>(4*StrToInt(lado1.Text)
+
      2*StrToInt(lado2.Text))) and

(contatempo<=(4*StrToInt(lado1.Text
)+
      4*StrToInt(lado2.Text))) then
      begin
        xvl:=0; yvl:=-126;
      end;
      // FIM
      if
(contatempo>(4*StrToInt(lado1.Text)
+
      4*StrToInt(lado2.Text))) then
      begin
        xvl:=0; yvl:=0;
        contatempo:=0; flagbotao:=0;
```

```
        initraje.Caption:='Iniciar
Trajetória';
      end;
    end;
end;
end;


//Equações da Cinemática
motor0:=trunc(-Sin(PI/3)*xvl+0.5*yvl);
motor1:=trunc(-yvl);
motor2:=trunc(Sin(PI/3)*xvl+0.5*yvl);


// Rotação_Anti-Horário
if bt5 then
begin
  motor0:=50;
  motor1:=50;
  motor2:=50;

end;
// Rotação_Horário
if bt6 then
begin
  motor0:=-50;
  motor1:=-50;
  motor2:=-50;
end;

// Direção dos Motores na posição
zero do buffer buf[0]
// 000
if (motor0>=0) and (motor1>=0) and
(motor2>=0) then
begin
  buf[0]:='A';
end;

// 001
if (motor0>=0) and (motor1>=0) and
(motor2<0) then
begin
  buf[0]:='B';
end;

// 010
if (motor0>=0) and (motor1<0) and
(motor2>=0) then
begin
  buf[0]:='C';
end;

// 011
if (motor0>=0) and (motor1<0) and
(motor2<0) then
begin
  buf[0]:='D';
end;

// 100
if (motor0<0) and (motor1>=0) and
(motor2>=0) then
begin
  buf[0]:='E';
end;

// 101
if (motor0<0) and (motor1>=0) and
(motor2<0) then
begin
  buf[0]:='F';
end;

// 110
if (motor0<0) and (motor1<0) and
(motor2>=0) then
begin
  buf[0]:='G';
end;

// 111
if (motor0<0) and (motor1<0) and
(motor2<0) then
begin
  buf[0]:='H';
end;

buf[1]:=char(abs(motor0));
buf[2]:=char(abs(motor1));
buf[3]:=char(abs(motor2));

enviar.Lines.Clear;
enviar.Lines.Add('Motor 0: ' +
inttostr(2*motor0));
enviar.Lines.Add('Motor 1: ' +
inttostr(2*motor1));
```

```
enviar.Lines.Add('Motor 2: ' +
inttostr(2*motor2));
enviar.Lines.Add('DIREÇÃO: ' +
buf[0]);                                    end.
Cliente.Send(buf);
end;
```

APÊNDICE D – Programa do Arduino

```
#include <SPI.h> // needed for Arduino versions later than 0018
#include <Ethernet.h>
#include <EthernetUdp.h> // UDP library from:
bjoern@cs.stanford.edu 12/30/2008
#include <stdio.h>
#include <stdlib.h>

int M0 = 9;
int M1 = 5;
int M2 = 6;

int dir01 = 7;
int dir02 = 8;
int dir11 = 2;
int dir12 = 3;
int dir21 = 0;
int dir22 = 1;

int contador = 0;

int bateria1 = 1;
int bateria2 = 2;
char tensao1[12];
char tensao2[12];

// Enter a MAC address and IP address for your controller
below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 0, 150);

unsigned int localPort = 50001;      // local port to listen on

// buffers for receiving and sending data
//char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; //buffer to hold
incoming packet,
 char packetBuffer[4]; //buffer to hold incoming packet,
// An EthernetUDP instance to let us send and receive packets
over UDP
EthernetUDP Udp;

void setup() {
```

```
  pinMode(M0, OUTPUT); pinMode(M1, OUTPUT); pinMode(M2,
OUTPUT);
  pinMode(dir01, OUTPUT); pinMode(dir02, OUTPUT);
pinMode(dir11, OUTPUT); pinMode(dir12, OUTPUT); pinMode(dir21,
OUTPUT); pinMode(dir22, OUTPUT);
  digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
  analogWrite(M0,0); analogWrite(M1,0); analogWrite(M2,0);

  // start the Ethernet and UDP:
  Ethernet.begin(mac,ip);
  Udp.begin(localPort);
}

void loop() {
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if(packetSize)
  {
    Udp.read(packetBuffer,UDP_TX_PACKET_MAX_SIZE);

    switch (packetBuffer[0]) {
    case 'A':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
    break;
    case 'B':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
    break;
    case 'C':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
    break;
    case 'D':
          digitalWrite(dir01,LOW); digitalWrite(dir02,HIGH);
          digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
          digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
    break;
    case 'E':
          digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
```

```
    break;
    case 'F':
          digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
          digitalWrite(dir11,LOW); digitalWrite(dir12,HIGH);
          digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
    break;
    case 'G':
          digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
          digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
          digitalWrite(dir21,LOW); digitalWrite(dir22,HIGH);
    break;
    case 'H':
          digitalWrite(dir01,HIGH); digitalWrite(dir02,LOW);
          digitalWrite(dir11,HIGH); digitalWrite(dir12,LOW);
          digitalWrite(dir21,HIGH); digitalWrite(dir22,LOW);
    break;
    }
  analogWrite(M0,2*packetBuffer[1]);
  analogWrite(M1,2*packetBuffer[2]);
  analogWrite(M2,2*packetBuffer[3]);

 // ENVIA UM OK DE RECEBIMENTO NA PORTA 50000
    Udp.beginPacket(Udp.remoteIP(), 50000);
    Udp.write("ok");
    Udp.endPacket();

 // APÓS 20 RECEBIMENTOS ENVIA TENSÃO DAS BATERIAS NAS PORTAS
49999 E 49998
    contador=contador+1;
    if (contador==20)
    {
      // Envia tensao bateria 1
      Udp.beginPacket(Udp.remoteIP(), 49999);
      Udp.write(itoa(analogRead(bateria1), tensao1, 10));
      Udp.endPacket();
      // Envia tensao bateria 2
      Udp.beginPacket(Udp.remoteIP(), 49998);
      Udp.write(itoa(analogRead(bateria2), tensao2, 10));
      Udp.endPacket();
      contador=0;
    }
  }
//  delay(10);
}
```