ROBÔ DELTA: *HARDWARE*, *SOFTWARE* E SIMULAÇÃO VIRTUAL

Fabiano Sandrini Moraes *, Cláudio Luís D'Elia Machado*, Mauro André Barbosa Cunha*

* *Grupo de Pesquisa em Automação e Controle - Coordenadoria de Engenharia Elétrica - Câmpus Pelotas - Instituto Federal Sul-rio-grandense*
Praça Vinte de Setembro, 455
Pelotas, RS, Brasil

Emails: fabianomoraes@pelotas.ifsul.edu.br, claudiomachado@pelotas.ifsul.edu.br, maurocunha@ifsul.edu.br

Abstract— This work presents the development of the hardware and software for the control system of a Delta robot to be used in teaching and research activities. Delta robot is a parallel robot and it presents high speed and accelearation that are desirable for many applications. The Delta robot is modeled and this model is used for controlling and for the virtual simulation. The hardware and software are described as well as the virtual simulator. The experimental results hold with the results obtained by using the developed virtual simulator.

Keywords— delta robot, parallel robot, control system, virtual simulation.

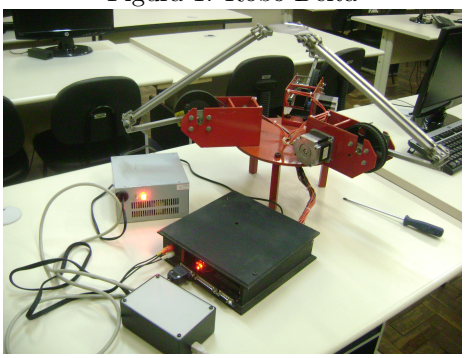
Resumo— O presente trabalho apresenta o desenvolvimento do *hardware* e do *software* para o sistema de controle de um robô delta visando seu uso no ensino e na pesquisa. O robô delta é um robô paralelo que apresenta alta velocidade e aceleração que pode ser usado em muitas aplicações. O robô delta é modulado e esse modelo é usado para o controle e para simulação virtual. O *hardware* e o *software*, bem como o simulador virtual são descritos. Os resultados experimentais estão de acordo com aqueles obtidos no simulador virtual.

Palavras-chave— Robô delta, robô paralelo, sistema de controle, simulação virtual.

1 Introdução

No ano de 2016, segundo a Federação Internacional de Robótica (IFR), as vendas de robôs industriais atingiram um novo recorde pelo quarto ano seguido crescendo 16% em todo o mundo com um total de 294.312 unidades vendidas e um volume de negócios de 40 bilhões de dólares. Em 2020, o IFR estima que serão vendidas 540 mil unidades com um crescimento médio de 15% ao ano (Robotics Industries Association - RIA, n.d.).

Figura 1: Robô Delta



O robô de arquitetura paralela tipo delta foi inventado em 1988 por um time de pesquisadores liderados pelo professor Reymond Clavel da École Polytechnique Fédérale de Lausanne (EPFL, Suíça), Clavel (1988). Este robô foi construído com a função de manipular luzes e pequenos objetos com uma velocidade alta, uma necessidade da indústria naquela época. O robô delta tem sido alvo de pesquisas recentes, como por exemplo em Kuo e Huang (2017), Jian e Lou (2017), dentre outras. Entre as características de um robô de arquitetura paralela tipo delta estão grandes velocidades e acelerações, grande capacidade de carga e precisão em seus movimentos. Em razão dessas características, os robôs paralelos passaram a ser uma ótima alternativa em operações do tipo "pegar e posicionar", necessárias em indústrias de embalagens, componentes eletrônicos e mecânicos, farmacêuticas e na medicina para a realização de cirurgias.

Este trabalho apresenta o desenvolvimento de *hardware* e *software* para acionamento e simulação virtual de um robô tipo delta, figura 1, que foi construído visando o seu uso para ensino e pesquisa na área de manipuladores paralelos. Nesta arquitetura existem três membros que conectam a base fixa ao efetuador e são compostos, cada um, por uma junta de rotação ativa, elo superior e elo inferior. O elo inferior é composto por uma sub-cadeia do tipo paralelogramo articulado com juntas esféricas. O efetuador, nesta arquitetura, fica restrito a três graus de liberdade, ficando com sua movimentação limitada as três translações no espaço sem a ocorrência de rotação.

2 Modelo cinemático do robô delta

A cinemática de robôs paralelos é diferente de robôs seriais. Enquanto nos robôs seriais a modelagem da cinemática direta é mais simples que a inversa, conforme Tsai (1999) e Craig (2012), mecanismos paralelos apresentam a modelagem da cinemática inversa bem mais simples que a direta.

A cinemática direta de robôs paralelos é complexa porque os ângulos das juntas passivas de um membro são dependentes dos ângulos das juntas ativas de todos os membros.

2.1 Cinemática inversa

O modelo, mostrado nesta seção, para a cinemática inversa é baseado em Tsai (1999) que desenvolve o equacionamento através de observação geométrica.

Considerando as figuras 2 e 3, três sistemas de coordenadas são utilizados na dedução das equações. Um sistema fixo de coordenadas (x,y,z) com origem no ponto O, um outro sistema fixo de coordenadas (x_i,y_i,z_i) com origem sobre a junta ativa A_i, e um sistema fixo na plataforma móvel de coordenadas (u,v,w) com origem no ponto P. Como não ocorre rotação entre os sistemas de coordenadas da base e da plataforma, tem-se que os pares de eixos $\vec{u}$ e $\vec{x}$, $\vec{v}$ e $\vec{y}$, $\vec{w}$ e $\vec{z}$ são paralelos para qualquer combinação possível dos ângulos das juntas A_i. O sistema de coordenadas (x_i,y_i,z_i) é orientado de tal forma que a projeção do elo 1_i sobre o plano (x_i,y_i) é colinear com o eixo x_i.

A cinemática inversa consiste em calcular os ângulos no espaço das juntas $(\theta_{1i},\theta_{2i},\theta_{3i})$, e por consequência o ângulo (θ_{1i}) no espaço dos atuadores, para uma dada posição do ponto P da plataforma no espaço cartesiano $(\vec{x},\vec{y},\vec{z})$.

O ângulo formado entre os eixos $\vec{x}$ e $\vec{x_i}$ é denominado de ϕ_i. O ângulo ϕ_i é convencionado como positivo no sentido anti-horário e é medido partindo do eixo $\vec{x}$. O valor do ângulo ϕ_i é constante e depende do projeto mecânico. A variável θ_{1i}, que representa ângulo entre o seguimento de reta $\overline{A_iB_i}$ em relação ao eixo $\vec{x_i}$.

Figura 2: Vista Frontal

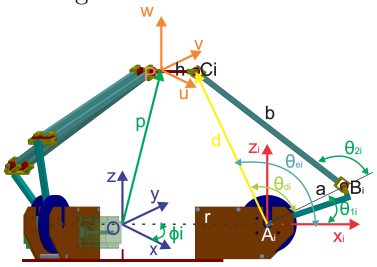
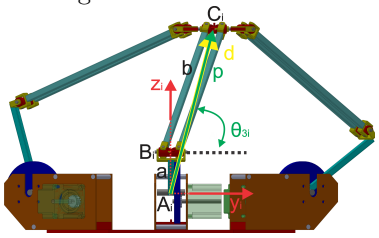


Figura 3: Vista Lateral



Com base nas figuras 2 e 3, a cinemática inversa calcula os ângulos $(\theta_{1i},\theta_{2i},\theta_{3i})$ como função da posição do ponto P. Este cálculo pode ser realizado em dois passos. No primeiro, os ângulos de junta θ_{2i} e θ_{3i} são determinados através de manipulação matemática da expressão

$$\overline{A_iB_i} + \overline{B_iC_i} = \overline{OP} + \overline{PC_i} - \overline{OA_i} \qquad (1)$$

onde os seguimentos de reta $\overline{OA_i}$, $\overline{A_iB_i}$, $\overline{B_iC_i}$ e $\overline{OP}$ representam os elos 0, 1_i, 2_i e 3 respectivamente. Essa formulação permite calcular as coordenadas (c_{xi},c_{yi},c_{zi}) do ponto C_i dadas no sistema de coordenadas fixado com a origem sobre a junta A_i. No segundo passo, pode-se aplicar a lei dos senos em um triângulo projetado no plano x_iz_i com lados dados pelos seguimentos $\overline{A_iB_i}$, $\overline{A_iC_i}$ e $\overline{B_iC_i}$ para obter θ_{1i}.

Pelo primeiro passo da solução da cinemática inversa, a variável θ_{3i} é calculada fazendo

$$\theta_{3i} = \cos^{-1}\left(\frac{c_{yi}}{b}\right) \qquad (2)$$

onde b é o comprimento do seguimento $\overline{B_iC_i}$ do elo 2_i. Pela função cosseno, há dois ângulos possíveis para θ_{3i}.

Também pelo primeiro passo da solução da cinemática inversa, a variável θ_{2i} resulta de

$$\theta_{2i} = \cos^{-1}\left(\frac{c_{xi}^2 + c_{yi}^2 + c_{zi}^2 - a^2 - b^2}{2ab\,\mathrm{sen}\theta_{3i}}\right) \qquad (3)$$

onde a é o comprimento do seguimento $\overline{A_iB_i}$ do elo $1i$. A variável θ_{2i} pode apresentar dois valores possíveis para cada valor possível de θ_{3i}.

Das equações 2 e 3, tem-se quatro possíveis conjuntos de soluções para as variáveis θ_{2i} e θ_{3i}. Entretanto, existem apenas dois valores distintos para a variável θ_{1i}.

Pelo segundo passo da solução da cinemática inversa, obtém-se a variável θ_{1i}, fazendo

$$\begin{aligned}\theta_{1i} = \quad & \tan^{-1}\left(\frac{c_{xi}}{c_{zi}}\right) \\ & - \mathrm{sen}^{-1}\left(\frac{b\cos\left(\frac{\pi}{2} - \theta_{3i}\right)\mathrm{sen}(\pi - \theta_{2i})}{\sqrt{C_{xi}^2 + C_{zi}^2}}\right)\end{aligned} \qquad (4)$$

onde, de acordo com a figura 2, o primeiro termo do lado direito da igualdade é o cálculo do ângulo θ_{ei} e o segundo termo é o cálculo de θ_{di}.

Assim ficam definidas todas as variáveis do espaço das juntas $(\theta_{1i},\theta_{2i},\theta_{3i})$, sendo que, as variáveis no espaço dos atuadores são os ângulos θ_{1i}.

Os cálculos de cinemática devem ser repetidos para cada um dos três membros.

Em análise dos resultados possíveis, pode-se imaginar a interseção entre uma esfera com centro em C_i e raio $\overline{B_iC_i}$ e um círculo com centro em A_i e raio $\overline{A_iB_i}$. Considerando que os elos estão perfeitamente ligados por juntas de rotação, há duas soluções possíveis: 1) Solução geral: O círculo perfura a esfera em dois pontos produzindo

duas soluções possíveis; 2) Solução singular: O círculo e as esfera são tangentes, com isto os segmentos $\overline{A_iB_i}$ e $\overline{C_iB_i}$ são colineares, resultando em uma única solução.

Diante do número de possibilidades de solução, as variáveis obtidas devem ser validadas através de um conjunto de regras mostradas a seguir: 1) Se algum resultado dos cálculos tiver valor complexo, significa que o ponto P especificado não faz parte do espaço de trabalho do robô; 2) Na variável θ_{3i}, tem-se dois valores de mesmo valor absoluto, mas com sinal contrário. Assim, levando-se em conta as limitações geométricas, o valor válido é o positivo; 3) Na variável θ_{2i}, tem-se também dois valores de mesmo valor absoluto e com sinal contrário. Assim, levando-se em conta as limitações geométricas, o valor válido é o positivo; e 4) O valor correto de θ_{1i} é obtido se $(d^2 + a^2) \geq [b\cos(\frac{\pi}{2} - \theta_{3i})]^2$. Senão, recalcula-se θ_{1i} assumindo que $\tan^{-1}\left(\frac{c_{xi}}{c_{zi}}\right) = \pi$.

2.2 *Cinemática direta*

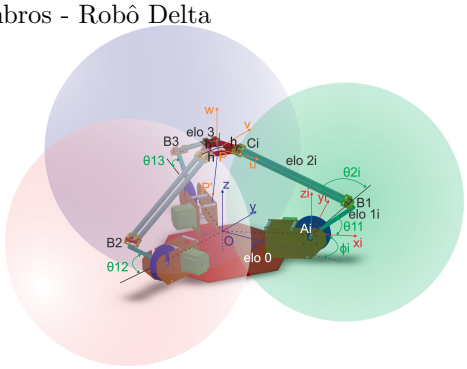
A solução da cinemática direta, segundo Tsai (1999), é obtida através da interpretação geométrica da intersecção de três esferas e uma reta. Cada uma das esferas tem centro em cada uma das juntas B_i e raio igual ao comprimento do seguimento $\overline{B_iC_i}$. Para solução geral, pela configuração geométrica e dimensional do robô, as três esferas se interceptam gerando uma figura geométrica com apenas dois vértices, um ponto na parte superior e outro na parte inferior. Uma reta passando por estes dois pontos, também passa pelo ponto P da plataforma. A solução da cinemática direta pode ser obtida pela solução de um sistema de equações que descrevem as esferas, a reta e calculam o ponto P. Um complicador é que este sistema de equações pode possuir múltiplas soluções. A figura 4 mostra a representação completa do robô com seus três membros e as três superfícies esféricas. Assim, a cinemática direta é dada como

$$\vec{p} = \vec{f}(\theta_{11}, \theta_{21}, \theta_{31}, \theta_{12}, \theta_{22}, \theta_{32}, \theta_{13}, \theta_{23}, \theta_{33}) \quad (5)$$

Detalhes do equacionamento são encontrados em Moraes (2016) e Tsai (1999).

Utilizando a intersecção das três superfícies esféricas para obter as coordenadas do ponto P, tem-se duas soluções possíveis para a cinemática direta: Solução geral - Ocorre a intersecção entre as três superfícies esféricas, onde tem-se duas soluções possíveis, as coordenadas dos pontos P e P'; Solução singular - Uma das superfícies esféricas é tangente ao círculo formado pela intersecção das outras duas. Tem-se uma única solução possível para a coordenada do ponto P;

Figura 4: Descrição cinemática direta para três membros - Robô Delta



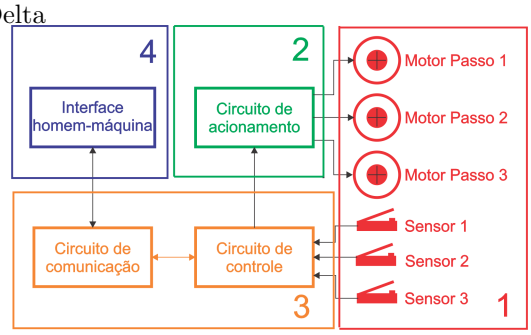
2.3 *Validação dos algoritmos de cinemática*

Os algoritmos desenvolvidos para cálculo das cinemáticas foram validados confrontando resultados da cinemática direta com a inversa. Deste modo, adotou-se os seguintes passos: 1) Arbitrou-se os valores das variáveis no espaço dos atuadores (θ_{1i}); 2) Calculou-se a posição do ponto P no espaço cartesiano através do equacionamento da cinemática direta; 3) Utilizou-se a posição calculada para o ponto P como entrada para o modelo matemático para cinemática inversa; e 4) Comparou-se com os valores das variáveis θ_{1i} com os valores arbitrados no passo 1.

3 *Hardware* Robô Delta

Esta seção descreve o *hardware* e *firmware* utilizados para acionamento e controle do robô. A figura 5 mostra o diagrama em blocos do *hardware* implementado.

Figura 5: Diagrama em blocos do *hardware* Robô Delta



3.1 *Atuadores e sensores*

Como atuadores foram utilizados três motores de passo 23km-c051-07v, que possuem as seguintes características (NMB Corporation, 2016): ângulo de passo $1,8° \pm 5\%$; resistência da bobina $3,4\Omega$ por fase; tensão nominal 5,10V por fase; e corrente nominal 1,5A por fase. Na ligação do motor de passo foi optado pela ligação bipolar. Conforme

Acarnley (2002), os enrolamentos da fase foram ligadas em série priorizando o torque.

Para os sensores, foram utilizadas três chaves de fim de curso com o contato normalmente aberto. As chaves foram fixadas uma em cada membro do robô, tendo como função sinalizar uma posição definida. Assim quando as três chaves estiverem acionadas indicará que o efetuador está nessa posição. Com isto, será possível posicionar o efetuador em uma posição inicial determinada e então inciar a sequência de movimento estabelecida pelo usuário.

3.2 Circuito para acionamento dos motores

Para o acionamento dos motores utilizou-se uma placa de acionamento para motores de passo baseada no *chip* TB6560AHQ fabricado pela Toshiba. Conforme Toshiba (2014), o *chip* TB6560AHQ é um *driver* para motores de passo do tipo PWM *chopper*, projetado para o controle bipolar com entrada de micropasso sinusoidal. O *chip* TB6560AHQ possui as seguintes características: tensão de alimentação de 6V; tensão de saída máxima, para os motores, de 40V; realização de micropasso com tamanho configurável de 1, 1/2, 1/8 e 1/16 passo; corrente máxima de pico na saída configurável em até 3,5A; frequência máxima de 15kHz.

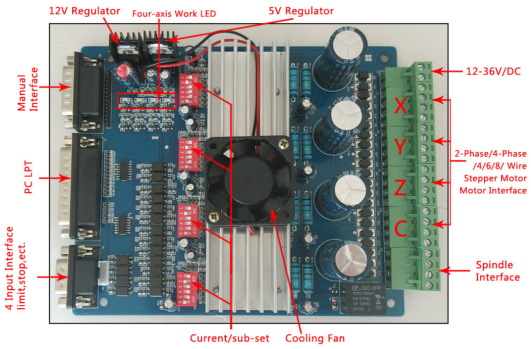
Para a execução de um passo, utilizando o *chip* TB6560AHQ, é necessário o controle de três sinais: O sinal de *enable*, que quando em nível alto ativa a saída para o motor; O sinal de direção que controla o sentido de rotação do motor (estando em nível baixo realiza passo no sentido horário e com nível alto realiza o passo no sentido anti-horário); e, por fim, o sinal de *clock* que ao ocorrer uma borda de subida produz um passo. A frequência máxima do sinal de *clock* é de 15kHz.

A placa de controle, figura 6, possui as seguintes características (Changzhou Wantai Eletrical Appliance, 2018): alimentação em tensão continua de 12V até 36V; controle de até quatro motores; corrente máxima de saída por motor de 3A; corrente de consumo da placa 2A; comunicação através de *interface* paralela ou *interface* manual; e Chaves *dip switch* para configuração do *chip* TB6560AHQ.

Para a alimentação da placa e motores foi utilizada uma fonte chaveada com tensão de 12V e capacidade de fornecimento de corrente de 12A. A corrente foi calculada de acordo com Changzhou Wantai Eletrical Appliance (2018), onde define que a corrente total necessária é a soma das correntes para os motores acrescido de 2A, resultando em 11A.

A configuração do *chip* TB6560AHQ é realizada utilizando as chaves *dip switch*. Na configuração correspondente ao limite de corrente fornecido ao motor foi selecionado 100%, representando

Figura 6: Placa para acionamento motor de passo



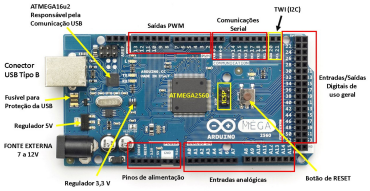
uma corrente de 3A. Para o modo de decaimento, que é o tempo de descarga da corrente no modo PWM, foi selecionado 50%. Selecionando o tamanho do micropasso como 1/8 e levando em consideração a relação de 14/70 entre as polias do acionamento, obtém-se o tamanho de cada passo fazendo

$$passo = \frac{1,8°}{8} \cdot \frac{14}{70} = 0,045° \qquad (6)$$

3.3 Circuito de controle e comunicação

No circuito de controle e comunicação foi utilizada uma placa de prototipagem Arduino Mega 2560, figura 7. O elemento principal desta placa é o microcontrolador ATmega2560 fabricado pela ATMEL.

Figura 7: Placa de prototipagem Arduino Mega 2560



Esta placa foi escolhida por apresentar um bom custo-beneficio, possuir compilador baseado em código livre, ter comunicação USB integrada e um grande número de de portas de entrada e saída (Arduino.cc, 2018b). Deste modo, possibilitando futuras expansões no projeto original, sem a necessidade de substituição da placa.

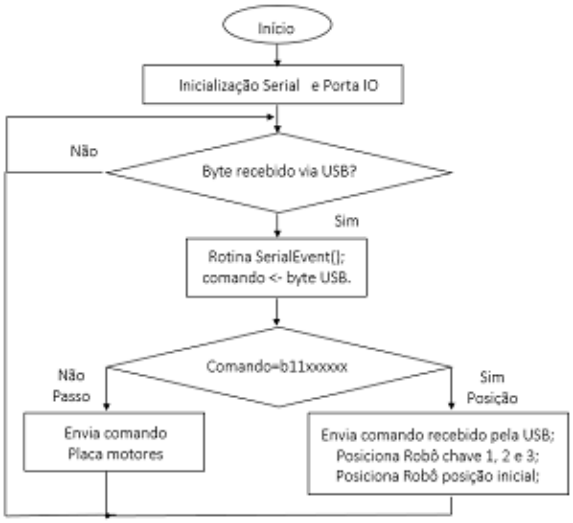
Na ligação do circuito de controle com os circuitos de acionamento, seção 3.2 , e sensores, seção 3.1, foram utilizados os pinos de entrada e saída de propósito geral da placa de prototipagem Arduino Mega 2560.

No circuito de comunicação foi utilizada a conexão USB integrada na placa de prototipagem Arduino Mega 2560. A realização da conversão da porta USART, do microcontrolador ATMEGA2560, em uma *interface* USB é realizada pelo microcontrolador ATMEGA16u2 fabricado

pela ATMEL. O microcontrolador ATMEGA16u2 vem integrado a placa de prototipagem, sendo utilizado por possuir o *hardware* para comunicação via *interface* USB integrado. Deste modo, através de um *firmware*, desenvolvido por Arduino.cc (2018b), que já vem gravado de fábrica no microcontrolador ATMEGA16u2, pode-se realizar a comunicação USB com placa de prototipagem Arduino Mega 2560.

Baseado no fluxograma da figura 8, desenvolveu-se o *firmware* utilizado na placa de prototipagem Arduino Mega 2560, usando-se a IDE de desenvolvimento Arduino versão 1.6.10 obtida em Arduino.cc (2018c).

Figura 8: Fluxograma do *firmware* - Arduino Mega 2560



Na realização da comunicação com a *interface* homem-máquina foi utilizada a biblioteca serial, com suas funções principais (Arduino.cc, 2018a).

O tráfego de dados foi definido em pacotes de um *byte*e a velocidade de transmissão e recepção foi configurada em 115200bps.

No controle dos motores e leitura dos sensores foram utilizadas as funções (Arduino.cc, 2018a): pinMode(A0,OUTPUT) - Configura o modo do pino definido. No caso o pino A0 foi definido como saída, para o pino ser definido como entrada substitui-se OUTPUT por INPUT; digitalWrite(A0,LOW) - Define o sinal de saída do pino A0 como nível lógico baixo, para o sinal ser definido com o nível lógico alto substitui-se LOW por HIGH; digitalRead(A0) - Executa a leitura do nível lógico do sinal do pino A0; PORTK - Escreve na saída os níveis lógicos dos sinais relativos a cada pino da porta k ao mesmo instante.

O controle dos motores foi realizado utilizando os pinos pertencentes a porta K do ATMEGA2560, isto possibilitou que os estados de todos os pinos fossem atualizados ao mesmo tempo através da escrita no registrador PORTK. Deste modo, cada *byte* recebido pela USB identificado como comando para os motores é gravado diretamente em PORTK; onde, os *bits* 5, 3 e 1 configuram a direção de giro dos motores e os *bits* 4, 2 e 0 executam uma borda de subida no sinal de *clock*, se estiverem com nível lógico alto. Após a passagem de $200\mu s$ os bits 4, 2 e 0 são zerados levando os pinos de *clock* para nível lógico baixo.

Quando o comando de posição inicial é recebido os membros do robô são movimentados, um por vez, até acionarem a chave de fim de curso, descrita na seção 3.1. Com a chave acionada o pino de monitoramento da chave é levado para nível logico alto, com isto é comandado um número determinado de passos ao motor levando o membro até a posição inicial definida. A posição inicial definida no espaço dos atuadores é $(0°;0°;0°)$, que corresponde a $(14,17;0;0)$cm no espaço cartesiano.

4 *Software* Robô Delta

O *software* Robô Delta foi desenvolvido utilizando a IDE para desenvolvimento Lazarus - Free Pascal versão 1.6 (Lazarus Team, 1993-2016). A IDE é baseada na linguagem de programação Pascal e utiliza como compilador o Free Pascal versão 3.0 (Free Pascal Team, 1993-2016). A escolha desta opção para o desenvolvimento do *software* foi devido a IDE Lazarus e o compilador Free Pascal possuírem código fonte aberto com capacidade para ser executado nos sistemas operacionais Windows, Mac e Linux. Outro fator foi que ambos possuem bibliotecas para o uso do Robô Delta Virtual, desenvolvido na linguagem VRML versão 2.0 (Web3D Consortium et al., 1997).

O *layout* do *software* é mostrado na figura 9 e o formulário de desenvolvimento com os componentes utilizados é mostrado na figura 10. Os blocos principais foram enumerados conforme: 1) Simulador; 2) Cinemática direta; 3) Cinemática inversa; 4) Sequência de movimentos; 5) Acionamento; 6) Execução; 7) Equipamento; 8) Comunicação.

Figura 9: *Layout* do *software* Robô Delta, com blocos principais enumerados.

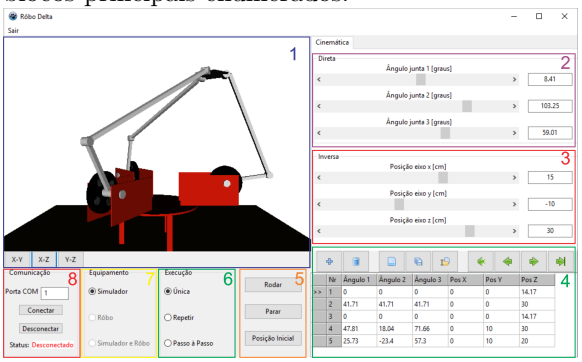
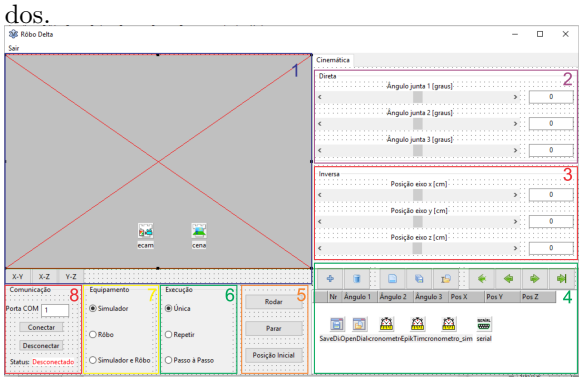


Figura 10: Formulário de desenvolvimento do *software* Robô Delta, com blocos principais enumerados.



4.1 Simulador

O bloco ”Simulador” tem a função de oferecer ao usuário a possibilidade de virtualização dos movimentos executados pelo Robô Delta, possibilitando assim a realização da sua programação *offline*.

Para isso foi desenvolvido o Robô Delta Virtual. O Robô Delta Virtual é um modelo 3D do Robô Delta, desenvolvido utilizando a linguagem VRML versao 2.0. O VRML ou *Virtual Reality Modeling Language* é uma linguagem de descrição que permite a criação, simulação, navegação e interação com objetos virtuais, criando cenas em três dimensões (Web3D Consortium et al., 1997)

A interação entre o modelo 3D do Robô e o software é realizada através da manipulação de seus nodos. Os nodos são responsáveis por descrever os objetos da cena junto com suas propriedades.

Para a construção do modelo foi utilizado o nodos geométricos *Cylinder* para simular as juntas do Robô. Assim, alterando a propriedade *rotation* do nodo é realizada uma rotação no modelo, simulando o movimento de uma junta. Como o modelo foi construído utilizando a hierarquia entre os nodos, esta rotação altera o posicionamento dos nodos de hierarquia inferior.

Para a ligação entre o *software* e o modelo virtual foi adaptada a biblioteca para desenvolvimento de jogos *Castle Game Engine* (Kamburelis, 2015). Os componentes utilizados no desenvolvimento do simulador foram: *TCastleScene* - carrega o arquivo com o modelo do Robô Delta Virtual; *TCastleControl* - realiza o controle e a exibição do modelo carregado; *TExamineCamera* - controla o modo de visualização do modelo, através do plano de visualização.

A seleção do plano de visualização é realizada através do acionamento dos botões ”X-Y”, ”X-Z” e ”Y-Z”. O ângulo de visualização também pode ser modificada com o uso do *mouse* sobre o modelo.

4.2 Cinemática direta e cinemática inversa

O bloco ”Cinemática direta” é onde o usuário pode alterar o valor das variáveis no espaço dos atuadores. A cada valor alterado a função que calcula a cinemática direta é executada e a sua saída é utilizada para atualizar os campos do bloco ”Cinemática inversa”.

Com os valores atualizados é executada a função ”anima” que possui como parâmetros de entrada as variáveis do espaço dos atuadores e as variáveis no espaço cartesiano. Esta função verifica se o equipamento está configurado como ”Robô” ou ”Simulador”. Caso esteja selecionado ”Robô”, os valores das variáveis no espaço dos atuadores passados como parâmetros são convertidos em quantidade de passos e o pacote de dados é montado utilizando o protocolo estabelecido na seção 3.3, sendo enviado ao circuito de controle através da comunicação USB. No modo ”Simulador” é executada a função que calcula a cinemática inversa, utilizando como entrada os valores das variáveis no espaço cartesiano passados como parâmetros e na saída desta função obtêm-se os valores das variáveis no espaço das juntas usadas para movimentar as juntas no simulador.

O envio de cada pacote de dados contendo o comando para execução dos passos é realizado a cada $1ms$ e os valores das variáveis no espaço das juntas do simulador são atualizados a cada $50ms$.

Para a marcação do tempo de envio e tempo de atualização foi utilizado o componente *TEpikTimer* da biblioteca *EpikTimer*. Este componente é um cronometro baseado na frequência de processamento da CPU (Lisjac, 2006).

A *interface* para a realização da comunicação comunicação USB com o *hardware* foi realizada utilizando o componente *TLazSerial* da biblioteca *LazSerial* (Team LazSerial, 2013).

O bloco ”Cinemática inversa” é onde o usuário pode alterar as variáveis do espaço cartesiano. A cada valor alterado a função que calcula a cinemática inversa é executada e a sua saída é utilizada para atualizar os campos do bloco ”Cinemática direta”.

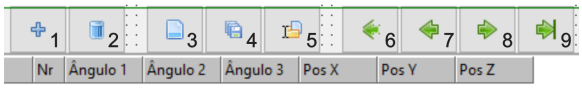
Com os valores atualizados é executada a função ”anima”, que procede como no bloco ”Cinemática direta”. Os componentes utilizados são os mesmos do bloco ”Cinemática direta”.

4.3 Sequência de movimento

O bloco ”Sequência de movimento” tem a função de armazenar a sequência de movimento definida pelo usuário. Para isto são utilizados os botões na parte superior do bloco, figura 11, com suas funções que permitem: inserir uma nova posição na sequência de movimentos, utilizando os valores dos blocos ”Cinemática direta” e ”Cinemática inversa”; apagar a posição atual da sequência de movimento; criar uma nova sequência de movimento;

salvar em arquivo a sequência de movimento; carregar uma sequência de movimento salva em arquivo; colocar o cursor na primeira posição da sequência de movimento; retornar o cursor uma posição na sequência de movimento; avançar o cursor uma posição na sequência de movimento; e colocar o cursor na última posição da sequência de movimento.

Figura 11: Detalhe do bloco ”Sequência de movimento”



4.4 Equipamento, execução e acionamento

O *software* possui três opções de configuração no bloco ”Equipamento”, que são ”Simulador”, ”Robô” e ”Simulador e Robô”. Esta configuração determina o tipo de programação executada, podendo ser *on-line* ou *off-line* e o equipamento que será executada a sequência de movimento programada.

Na programação *on-line*, em que o programador conduz fisicamente o efetuador final do robô através de uma sequência de posições desejadas, é selecionada com as opções ”Robô” ou ”Simulador e Robô”. A diferença entre as duas opções é que quando selecionado ”Robô” o simulador não é utilizado, já na outra opção são utilizados o robô e o simulador.

Para a programação *off-line* utiliza-se a opção ”Simulador”. Nesta opção não é necessário que o Robô Delta esteja conectado ao *software*, podendo realizar toda a programação utilizando apenas o simulador. Durante a execução da sequência de movimento programada, pode-se utilizar as mesmas configurações do equipamento para selecionar onde ocorrerá o movimento.

O bloco ”Execução” configura o modo que irá ocorrer as transições entre cada posição da sequência de movimento programada, tendo três opções: ”Única”, ”Repetir” e ”Passo a Passo”. Na configuração ”Única” a transição ocorre de maneira automática, sendo executada a sequência de movimento uma única vez. Com a opção ”Repetir” a transição ocorre de maneira automática, com a sequência de movimento sendo executada indefinidamente. No ”Passo a Passo” para que ocorra a transição entre cada posição é necessário o comando do usuário.

No bloco ”Acionamento” o *software* possui três opções que são escolhidas através dos botões ”Rodar”, ”Parar” e ”Posição Inicial”.

Acionando o botão ”Rodar” é iniciada a sequência de movimento estabelecida no bloco ”Sequência de movimento”, sendo o modo de transição entre cada posição definida no bloco ”Execução”. Caso o método de transição configurado seja o ”Passo a Passo” é necessário pressionar o botão ”Rodar” para que a ocorra cada transição.

A opção ”Parar” é utilizada quando o método de transição configurado for ”Única” ou ”Repetir”, sendo utilizado para interromper a sequência de movimento.

O botão ”Posição Inicial” é utilizado para levar o efetuador final para a posição inicial definida, para isto no simulador as variáveis do espaço cartesiano são definidas como (0;0;14,17)cm e no Robô Delta, conforme o protocolo definido na seção 3.3, é enviado um pacote de dados com um *byte* de valor 255 para o circuito de comunicação e controle.

4.5 Comunicação

O bloco ”Comunicação” é responsável por configurar, estabelecer e informar o *status* da comunicação com o *hardware*.

Inicialmente, o usuário define a porta no qual o *hardware* está conectado, utilizando o campo ”Porta”. Após, acionando o botão ”Conectar” é iniciada a conexão com o *hardware* e o comando de posição inicial é enviado. Com a sequência de posicionamento concluída o *software* recebe o *byte* 255, sinalizando que a conexão foi estabelecida. Caso isto não ocorra a conexão é interrompida e uma mensagem de erro é mostrada.

O botão ”Desconectar” é utilizado para encerrar a conexão com o *hardware*. Para isto o comando de posição inicial é enviado e o *software* fica aguardando o recebimento do *byte* 255, indicando a conclusão da sequência de posicionamento. Com o recebimento do *byte* a conexão é encerrada, caso isto não ocorra uma mensagem de erro é mostrada.

O campo ”Status” é utilizado para informar ao usuário o estado da conexão com o *hardware*, informando se está conectado ou desconectado.

5 Conclusão

Neste trabalho apresentou-se o *hardware* e o *software* desenvolvidos para um robô delta. Foram apresentados os modelos matemáticos da cinemática direta e da cinemática inversa. Os modelos propostos, através de seus algoritmos, mostraram-se precisos e exatos para uso no Robô Delta e com um tempo de execução que permitiram o comando de um passo a cada $1ms$ no *software* Robô Delta.

No *hardware* proposto, o uso da placa de acionamento de motores, baseada no *chip* TB6560AHQ, mostrou-se uma escolha eficaz, pois propiciou a diminuição do *firmware* necessário para o acionamento dos motores, simplificando o seu desenvolvimento, bem como a execução simplificada de micropassos, melhorando a precisão

dos atuadores de $1,8^\circ$ para $0,045^\circ$ e com possibilidade de chegar a $0,0225^\circ$.

A placa de prototipagem Arduino Mega 2560 permitiu, através do uso do sua IDE de desenvolvimento, a elaboração de um código para o *firmware* simples e de rápido desenvolvimento, através do uso de suas bibliotecas disponíveis.

Deste modo, o *hardware* mostrou-se eficiente no acionamento dos motores e com possibilidades de serem desenvolvidas novas funções, como o sensoreamento da posição das juntas, sem a necessidade de reformulação do *hardware*, apenas incluindo novos circuitos.

O *software* Robô Delta desenvolvido, com o uso de ambiente e compilador com licença GLP, mostrou-se intuitivo, de fácil manuseio e estável. Não possuindo a necessidade de instalação de *drives* adicionais para seu uso.

Nos trabalhos futuros pretende-se desenvolver um sistema de controle em malha fechada para posição, velocidade e aceleração do robô, bem como a inserção e controle de ferramentas na plataforma móvel.

Referências

Acarnley, P. P. (2002). *Stepping Motors - A guide to theory and pratice*, 4 edn, The Institution of Eletrical Engineer.

Arduino.cc (2018a). Arduino language references, Disponível em: <https:// www.arduino.cc/ reference/ en/>. Acessado em 03 jul. 2018.

Arduino.cc (2018b). Arduino mega, Disponível em: <https:// www.arduino.cc/ en/ Main/ ArduinoBoardMega>. Acessado em 03 jul. 2018.

Arduino.cc (2018c). Arduino software, Disponível em: <https:// www.arduino.cc/ en/ Main/ Software>. Acessado em 03 jul. 2018.

Changzhou Wantai Eletrical Appliance (2018). User guide for 4 axis tb6560 driver board.

Clavel, R. (1988). Delta, a fast robot with parallel geometry, *Proc. Int. Symposium on Industrial Robots*, pp. 91–100.

Craig, J. J. (2012). *Introduction to robotics: mechanics and control*, 3 edn, Pearson Education do Brasil.

Free Pascal Team (1993-2016). Free pascal: A 32, 64 and 16 bit professional pascal compiler, Disponível em: <http://www.freepascal.org>. Acessado em 03 jul. 2018.

Jian, S. e Lou, Y. (2017). Application of motion control system for delta parallel robot, *Information and Automation (ICIA), 2017 IEEE International Conference on*, IEEE, pp. 732–736.

Kamburelis, M. (2015). Castle game engine versão 5.2.0, Disponível em: <http:// wiki.freepascal.org/ CastleGameEngine>. Acessado em 03 jul. 2018.

Kuo, Y.-L. e Huang, P.-Y. (2017). Experimental and simulation studies of motion control of a delta robot using a model-based approach, *International Journal of Advanced Robotic Systems* **14**(6).

Lazarus Team (1993-2016). Lazarus: The professional free pascal rad ide, Disponível em: <http:// www.lazarus-ide.org>. Acessado em 03 jul. 2018.

Lisjac, T. (2006). Epiktimer versão 1.0 06-10-2006, Disponível em: <http:// wiki.freepascal.org/ EpikTimer>. Acessado em 03 jul. 2018.

Moraes, F. S. (2016). Implementação, controle e simulação virtual de um robô de arquitetura paralela tipo delta. Monografia (Bacharel em Engenharia Elétrica), IFSUL (Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense), Pelotas, Brasil.

NMB Corporation (2016). Precision step motors, 23km−c 1.8° hybrid (high torque), Disponível em: <http://www.nmbcorp.com>. Acessado em 03 jul. 2018.

Robotics Industries Association - RIA (n.d.). Disponível em: <http://www.robotics.org/>. Acessado em 03 jul. 2018.

Team LazSerial (2013). Lazserial versão v0.1, Disponível em: <https:// github.com/ me2d13/ luamacros / tree/ master/ inc/ LazSerial>. Acessado em 03 jul. 2018.

Toshiba (2014). Datasheet tb6560ahq, tb6560afg, Disponível em: <http:// toshiba.semicon-storage.com/ lapt/ product/ linear/ motor-driver/ detail.TB6560AHQ.html>. Acessado em 03 jul. 2018.

Tsai, L.-W. (1999). *Robot analysis: the mechanics of serial and parallel manipulators*, 1 edn, John Wiley & Sons.

Web3D Consortium et al. (1997). Vrml-virtual reality modeling language, Disponível em: <http:// www.web3d.org>. Acessado em 03 ago. 2016.