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

NILTON CESAR DIAS
RAFAEL CRISPIM

PROTÓTIPO DE UM ROBÔ PARALELO PLANAR COM DOIS GRAUS DE LIBERDADE

TRABALHO DE CONCLUSÃO DE CURSO

NILTON CESAR DIAS
RAFAEL CRISPIM

PROTÓTIPO DE UM ROBÔ PARALELO PLANAR COM DOIS GRAUS DE LIBERDADE

JOINVILLE, 2017

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

NILTON CESAR DIAS
RAFAEL CRISPIM

PROTÓTIPO DE UM ROBÔ PARALELO PLANAR COM DOIS GRAUS DE LIBERDADE

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

JOINVILLE, 2017

Dias, Nilton César.
Protótipo de um robô paralelo planar com dois graus de liberdade / Nilton César Dias. Rafael Crispim – Joinville: Instituto Federal de Santa Catarina, 2017. 77 p.

Trabalho de Conclusão de Curso - Instituto Federal de Santa Catarina, 2017. Graduação. Curso Superior de Tecnologia em Mecatrônica Industrial. Modalidade: Presencial.
Orientador: Michael Klug, Dr.

1. Mecatrônica 2. Robô 3. Planar.

I. PR OTÓTIPO DE UM ROBÔ PARALELO PLANAR COM DOIS GRAUS DE LIBERDADE.

PROTÓTIPO DE UM ROBÔ PARALELO PLANAR COM DOIS GRAUS DE LIBERDADE

NILTON CESAR DIAS
RAFAEL CRISPIM

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, 20 de abril de 2017.

Banca Examinadora:

_____
Prof. Michael Klug, Dr. Eng.
Orientador

_____
Prof. Ivandro Bonetti, Me. Eng.
Avaliador

_____
Prof. Carlos Toshiyuki Matsumi, Me. Eng.
Avaliador

AGRADECIMENTOS

Agradecemos a Deus pela força, saúde e inteligência sem a qual não poderíamos sequer desenvolver qualquer parte deste trabalho.

Aos nossos pais pela educação e formação como cidadãos que nos fizeram através de seus exemplos buscar soluções nas dificuldades.

As pessoas mais próximas que nos apoiaram no projeto direta e indiretamente.

Ao orientador que nos forneceu a estrutura necessária para o aprendizado e desenvolvimento do projeto.

Ao IFSC que nos concedeu espaço, treinamento e materiais necessários a execução desse projeto.

RESUMO

Neste trabalho abordaremos a importância dos conceitos da mecatrônica aplicados a cinemática de robôs industriais presentes nos mais diversos processos fabris.

Constantemente os processos de produção vem buscando maior qualidade, padronização e produtividade, sendo assim o estudo sobre robôs tem grande importância do ponto de vista socioeconômico.

Inicialmente serão vistos os formatos de construção de robôs com arquiteturas seriais e paralelas mais comuns no dia-a-dia e suas aplicações, verificando os estudos já realizados dessas formas de arquitetura, buscando atender as áreas com menos estudos desenvolvidos.

Também será abordada a arquitetura paralela, que é relativamente nova quando comparada com a arquitetura serial, demonstrando a importância do estudo dessas formas de arquitetura. Bem como a apresentação das qualidades dos robôs paralelos como tipo de estrutura, precisão; rapidez; leveza e alta capacidade de carga.

Tendo como foco o estudo de um robô com arquitetura paralela planar contemplando a construção de um protótipo com fins didáticos e que possa ser utilizado para desenvolvimento nos testes dos métodos de controle, utilizando as plataformas *Visual Studio* e IDE Arduino para o desenvolvimento dos métodos de controle.

Por fim a demonstração dos resultados de como protótipo se comportou nos testes, operando através da cinemática direta e inversa, projeção de uma trajetória com valores de coordenadas xy inseridos através do programa desenvolvido no Visual C# da plataforma Windows.

Palavras-chave: Mecatrônica; Robô Paralelo; Cinemática.

ABSTRACT

In this work we will focus on the importance of the concepts of mechatronics applied to the kinematics of industrial robots present in the most diverse manufacturing processes.

Constantly the production processes have been seeking higher quality, standardization and productivity, so the study on robots has great importance from the socioeconomic point of view.

Initially, it will be seen the construction of robots with the most common serial and parallel architectures in their day-to-day and their applications, verifying the already realized studies of these forms of architecture, seeking to attend to the areas with less developed studies.

Parallel architecture will also be approached, which is relatively new when compared to the serial architecture, demonstrating the importance of the study of these forms of architecture. As well as the presentation of the qualities of the parallel robots as type of structure, precision; quickness; lightweight and high load capacity

The objective of this study was to study a robot with parallel planar architecture contemplating the construction of a prototype with didactic purposes and that can be used for development in the testing of control methods using the Visual Studio and Arduino IDE platforms for the development of control methods.

Finally the demonstration of the prototype results behaved in the tests, operating through direct and inverse kinematics, projection of a trajectory with values of xy coordinates inserted through the program developed in Visual C # of the Windows platform.

Keywords; Mechatronics; Parallel Robot; Kinematics.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

A história da automação industrial é caracterizada por períodos de rápida mudança, como uma causa ou um efeito, onde esses períodos de mudança de técnicas de automação parecem ligados à economia mundial (Craig, 1989).

Dentre estas mudanças notou-se um impulso significativo para o desenvolvimento de mecanismos robóticos paralelos e mecanismos semelhantes a humanos ou animais, a fim de obter um melhor entendimento sobre esses projetos. Isso possibilitou o surgimento de novos avanços tecnológicos, tais como novos materiais e atuadores conceitualmente diferentes (Lenarcic,2013).

1.2 Justificativa

Com o passar do tempo a tecnologia evolui para dar maior assistência ao trabalho desempenhado pela sociedade. O uso de robôs em atividades insalubres e perigosas minimiza ou elimina os riscos que pessoas seriam expostas. O robô de arquitetura paralela em específico vem sendo visto como a nova geração em matéria de conceito mecânico, dando uma maior flexibilidade à manufatura atual e resolvendo diversos problemas. Por ser uma arquitetura relativamente nova, ainda existem muitas questões mecânicas, eletrônicas e de modelagem a serem trabalhadas (Tartari Filho, 2006).

Há também um termo hoje sendo muito utilizado que é o da Indústria 4.0, esta que engloba algumas tecnologias para automação e troca de dados utilizando conceitos de sistemas ciber-físicos, internet das coisas e computação em nuvem, considerada a quarta revolução industrial. Apesar de sua proposta parecer utópica sua realidade não está muito distante da nossa caso faça jus ao seu potencial.

1.3 Objetivo geral

Desenvolver um protótipo de um robô de arquitetura paralela planar com dois graus de liberdade e configuração de um mecanismo de cinco barras.

1.4 Objetivos específicos

- Aplicação de conceitos da cinemática de mecanismos;

- Projeto e construção de um mecanismo de cinco barras com dois graus de liberdade, capaz de fornecer movimentos de acordo com a sua arquitetura;
- Desenvolvimento do programa utilizado no controle do robô.

1.5 Robô – definição

O termo robô pode ser definido de várias formas, seguem abaixo dois exemplos consistentes:

> *"Uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial" Norma ISO (International Organization for Standardization) 10218, (Agostini,2012).*

> *"Um manipulador reprogramável e multifuncional projetado para movimentar materiais, partes, ferramentas ou outros aparelhos especializados através de vários movimentos programados a fim de realizar uma grande variedade de tarefas". Instituto Americano de Robótica (1979).*

Através dessas definições torna-se possível perceber o grau de importância do estudo e desenvolvimento de estruturas robóticas, com objetivo de facilitar e solucionar processos de produção que necessitem de execução contínua com precisão e rapidez.

1.6 Classificação dos robôs

Os robôs podem ser classificados através de vários critérios, tais como:

- Autonomia do sistema de controle;
- Mobilidade da base: define se o robô será fixo ou móvel;
- Estrutura cinemática: refere-se ao modelo de construção dos elos e juntas do mecanismo;
- Forma de acionamento: determina se o robô terá movimento programado ou controlado por operador, também o meio de transmissão de dados e controle;
- Graus de liberdade: diz respeito a capacidade de posicionamento e orientação que um robô consegue ter da sua base até ponto de interesse;
- Geometria do espaço de trabalho: está diretamente ligada aos graus de liberdade, pois estes determinam a área no espaço onde o robô pode operar. Criando uma geometria condizente as suas limitações.

A Figura 1 mostra um exemplo de ramificação que pode ser utilizado para classificar os robôs.

A balho de
conclusã

-
-
-

A gens que
o robô d erial.
S resadora
paralela , leveza,
rapidez, que justi-
ficam o n, Hess-
Coelho,

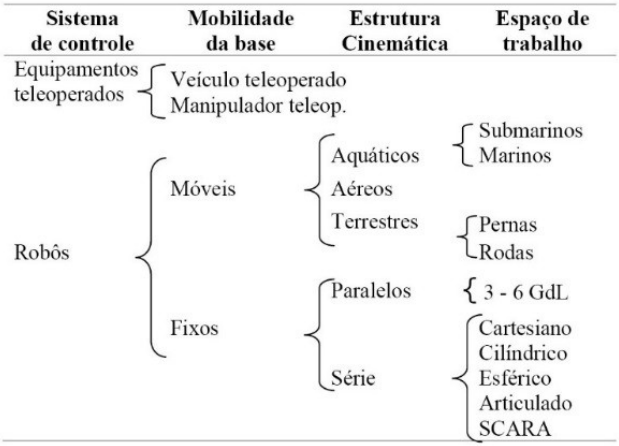


Figura 1 - Classificação dos robôs. Página da UFABC
Fonte: Luis A.M.Riascos

1 A estrutura do robô permite a configuração de três graus de liberdade, porém o desenvolvimento desse projeto será com apenas dois graus de liberdade.

1.6.1 Classificação geral - robô paralelo

Definição de robô paralelo segundo livro *Dynamics of Parallel Robots*

> *Os robôs paralelos, também chamados de manipuladores paralelos ou máquinas paralelas (PKM), são definidos em (Leinonen 1991) como robôs que controlam o movimento de seu posicionamento final por meio de pelo menos duas cadeias Cinemáticas que vão desde de o instrumento terminal até a base fixa (apud Briot, Khalil,2015).*

A partir dessa definição é possível analisar os elementos que compõem a configuração de um robô paralelo. Como observado na Figura 2, esses elementos podem ser divididos da seguinte maneira:

a) Base;
b) Plataforma móvel ou ponto de interesse;
c) Cadeias cinemáticas (ligam a base até o ponto de interesse).

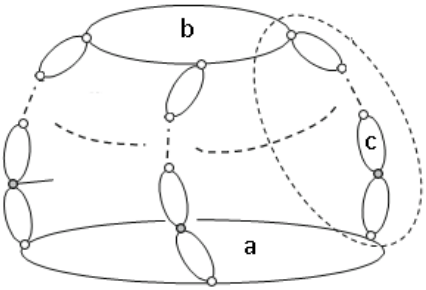


Figura 2 - Exemplo de estrutura – robô paralelo
Fonte: (Briot, Khalil, 2015, p.4)

1.7 Modelos de robôs paralelos

Os robôs paralelos podem ser utilizados em várias aplicações onde haja a necessidade de tal estrutura com intuito de otimizar o processo, se comparado com as soluções de cadeias seriais.

Com base no contexto anterior e nas características dos robôs paralelos, algumas aplicações podem ser observadas nas figuras a seguir.

Figura 3 - Robôs Adept Quattro pick and place "pegar e colocar" chocolates
Fonte: (Briot, Khalil, 2015, p.5.)

Figura 4 - Simulador de voo
Fonte: (Briot, Khalil,2015, p.6.)

Figura 5 - O omega6 baseado em uma arquitetura Delta -like
Fonte: (Briot, Khalil,2015, p.6.)

1.7.1 Classificação Especifica - robô paralelo

De acordo com *Dynamics of Parallel Robots,* não existem métodos de classificação para robôs paralelos, então o que tem sido feito é agrupá-los em função do número de graus de liberdade da sua plataforma (Briot, Khalil,2015). Dessa maneira é possível separar os robôs paralelos e entendê-los melhor.

Os robôs paralelos podem ser divididos em dois grandes grupos: os robôs planares e os robôs espaciais. O foco nessa pesquisa será os robôs planares.

1.8 Robôs planares

A classificação de robôs planares é dada aos robôs que tem a capacidade de movimentar objetos em um plano, podendo ser definidos através de três grupos principais:

- Robôs com dois graus de liberdade capazes de posicionar um ponto no plano (Figura 6,Figura 7);
- Robôs com dois graus de liberdade capazes de posicionar um dispositivo com orientação constante no plano (Figura 8);
- Robôs com três graus de liberdade capazes de posicionar um dispositivo no plano utilizando dois graus de liberdade e rotação em torno do eixo normal ao plano (Figura 9).

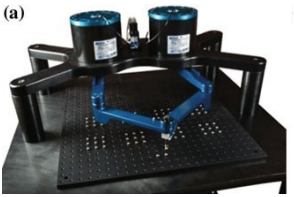


Figura 6 – Dextar : um mecanismo de cinco barras (arquitetura planar RRRRR)
Fonte: (Briot, Khalil,2015, p.10.)

Figura 7 - Paraplacer (PRRRP arquitetura planar)
Fonte: (Briot, Khalil,2015, p.10.)

Figura 8 - Robô PacDrive Delta
Fonte: (Briot, Khalil,2015, p.10.)

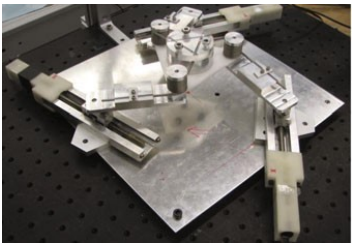


Figura 9 - Protótipo de um robô 3 – PRR
Fonte: (Briot, Khalil,2015, p.11.)

O objeto de pesquisa desse trabalho será o robô paralelo de arquitetura planar (RRRRR) semelhante ao Dextar da Figura 6 encontrada. Cuja estrutura é composta de um mecanismo de cinco barras.

1.9 Elementos de construção – robô planar

Partindo do princípio existente de um robô planar com mecanismo de cinco barras, pode-se averiguar os elementos que se fazem necessários para sua construção, assim também como adaptações e melhorias que podem ser desenvolvidas a fim de que esse mecanismo venha a atender as necessidades exigidas. No capítulo 2 será abordado o estudo de revisão bibliográfica a respeito dos principais componentes utilizados nessa forma de estrutura.

2 REVISÃO BIBLIOGRÁFICA

Os pontos de estudo e revisão literária deste projeto serão a cinemática, direta e inversa, e os componentes utilizados.

2.1 Cinemática

A cinemática segundo o livro Cinemática e Dinâmica dos Mecanismos, estuda o movimento desconsiderando as forças que o causaram. Dessa forma é possível entender que são desconsideradas além da força, grandezas como velocidade e aceleração, pois essas grandezas são produtos da força.

A cinemática direta proporciona a posição do ponto de interesse a partir do ângulo dos atuadores (motores), diferentemente da cinemática inversa que a partir do ponto de interesse obtém-se a posição de cada atuador.

Não existe um procedimento geral para o cálculo da cinemática inversa ou direta de mecanismos paralelos, já que suas estruturas cinemáticas são muito diversas. Em contraste com os mecanismos seriais, resolver o problema da cinemática direta de mecanismos paralelos é um problema complexo, enquanto que o problema da cinemática inversa é mais simples.

Cada arquitetura paralela possui características particulares, dificultando a aplicação de equações gerais, somente obtendo-as a partir da análise de cada uma, relacionando o ponto de interesse com a origem do sistema.

2.2 Controle

Devido à sua alta rigidez, os robôs paralelos permitem que alguns modelos operem com altíssimas velocidades e acelerações, ao qual gerou a necessidade do desenvolvimento de sistemas de controle de alto desempenho (Merlet, 2000). Por tratar-se de um projeto didático, a lógica de controle será compatível com o controlador que será utilizado.

2.3 Motor

A escolha do motor em um robô é de extrema importância, pois define o tipo de sistema de controle que será utilizado e a precisão do mesmo, assim também como sua potência e velocidade. Como o intuito nesse TCC é desenvolver um robô industrial com conceito didático e de baixo investimento, optou-se pela utilização de motores de passo que proporcionam boa precisão e facilidade de controle de posição. Este motor está disponível para uso na instituição.

2.3.1 Classificação de motores de passo

O motor de passo possui esse nome, pois seu deslocamento angular é feito em “passos”, esses passos são frações de uma revolução completa.

Partindo da ideia que para completar uma revolução completa um motor deva realizar por exemplo 200 passos, pode-se dizer que cada passo desse motor corresponde a 1,8°

A definição do funcionamento segundo um artigo de pesquisa e desenvolvimento de produtos da empresa de instrumentação e eletrônica Maxwell Bohr é:

> *No seu interior há estatores formados por bobinas que geram, quando percorridos por uma corrente elétrica, o campo magnético necessário para o movimento do rotor, que é construído com imãs permanentes confeccionados de acordo com o número de passos. Essa rotação é controlada por meio de um circuito externo que promove a oscilação do sinal que percorrerá os pares de estatores e, por isso, não pode ser conectado diretamente à alimentação, pois desse modo não haveria a pulsação necessária para que o motor possa girar (Patsko,2006).*

Segundo a definição acima, entende-se que a construção do motor de passo definirá a sua resolução normal, sendo essa de acordo com a quantidade de bobinas do estator. Também é possível observar que o motor de passo necessita de um controle externo que execute a oscilação de energia nas bobinas do estator de maneira conveniente com o deslocamento angular que se deseja obter. Essa execução de oscilação de energia pode ser vista através da Figura 10.

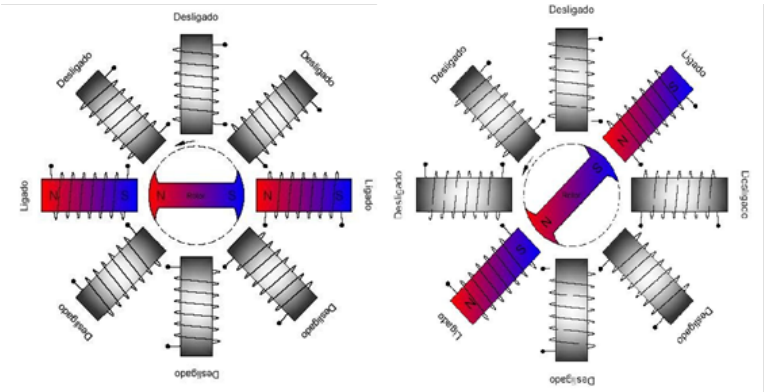


Figura 10 - Rotor se orienta de acordo com as bobinas do estator que estão energizadas.
Fonte: (Patsko,2006, p.2.)

2.3.2 Características elétricas

Analisando a Figura 10 do tópico anterior facilita-se muito a compreensão do funcionamento do motor de passo. A ligação elétrica de um motor de passo pode ser feita de três modos clássicos: unipolar, bipolar em série e bipolar em paralelo.

Para explicar esses tipos de ligação, segue abaixo a Figura 11, retirada de um manual de motor de passo HT23-394 do fabricante Kalatec.

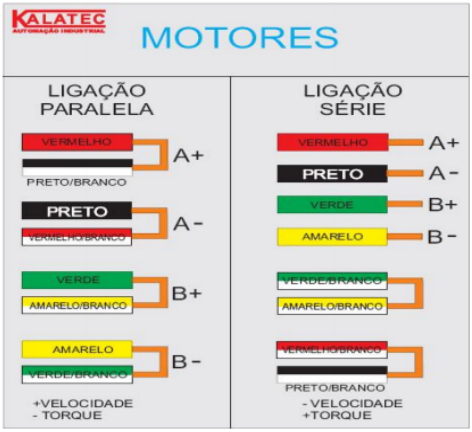


Figura 11 - Esquema de ligação – motor de passo.
Fonte: (KALATEC. Como são classificados os motores de passo. Em: <http://www.kalatec.com.br/motoresdepasso/motor-de-passo>. Acesso em: 22 outubro 2016.)

A Figura 11 mostra exemplos de ligação para o motor de passo, onde é possível definir a escolha da ligação de acordo com o objetivo da aplicação ou com limitações dos conjuntos envolvidos (drivers, bitola de cabos, consumo etc.). Quando o objetivo é se obter mais velocidade, é realizada a ligação em paralelo, caso se necessita de mais torque se realiza a ligação em série.

Como a ligação do motor pode ser alterada mas as propriedades físicas não, deve-se ficar atento a corrente de consumo para cada uma dessa ligações. O fabricante Kalatec especifica, por exemplo no manual referente a Figura 11, que para ligação bipolar em série deve-se ajustar a corrente no driver para 30% menor que a descrita no motor, já para liga-

ção bipolar em paralelo, ajustar a corrente no driver para 30% maior que a descrita no motor. Quando a ligação é feita de modo unipolar, deve-se seguir a ligação série, onde os pares não conectados nas fases do driver são conectados no 0 ou +VDC da fonte.

2.3.3 Características mecânicas

Além das características elétricas do motor de passo, também devem ser analisadas as características mecânicas (torque, espaço físico, peso e etc.), que são de extrema importância para a escolha do motor.

Segundo o fabricante Kalatec Automação, os motores de passo são classificados de acordo com o seu tamanho pela norma internacional NEMA, por exemplo um motor NEMA 23 significa que a aresta do flange do motor possui o tamanho de 2,3 polegadas.

O comprimento de um motor de passo é determinado através do número de *stacks* (rotores) presentes na construção do motor. Os motores de passo são montados em pares de *stacks*. Através da Figura 12 e Figura 13 pode ser visto o que são esses *stacks*.

Figura 12 - *Stack* de motor de passo
Fonte: (KALATEC. Como são classificados os motores de passo. Em: <http://www.kalatec.com.br/motoresdepasso/motor-de-passo> Acesso em: 22 outubro 2016.)

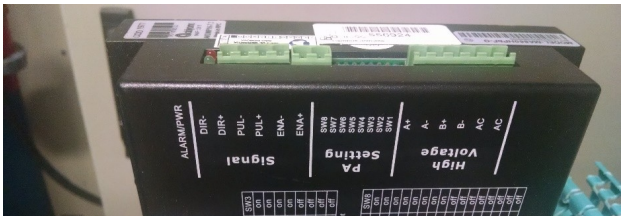
Como analisado na Figura 12 e Figura 13 o número de *stacks* determina o comprimento do motor de passo. Isso também influenciará diretamente no torque estático do motor, por exemplo: se um rotor com um *stack* resultar em um torque de 0,9Nm, um motor com dois *stacks* geraria um torque estático de 1,8 N·m.

2.4 *Driver* – motor de passo

O motor de passo como visto anteriormente não pode ser alimentado diretamente à rede elétrica, necessitando assim de um sistema de controle que alterne as alimentações em suas bobinas, esse sistema é chamado de *driver* de motor de passo. Para entender melhor como seria essa modulação de pulsos alternando nas bobinas, segue a explicação segundo o livro "AVR e Arduino técnicas de projeto".

> *O controle do motor de passo é feito com a sequência correta de energização das bobinas. Podem ser acionadas uma ou mais bobinas ao mesmo tempo. Com o acionamento de duas bobinas simultaneamente, existe a possibilidade de se obter um meio passo de giro ou um torque maior (Lima, Villaça,2012).*

Os *drivers* de motor de passo são dispositivos eletrônicos que recebem um sinal de controle e o transformam em sinal de potência. A po-



tência, desses sinais devem ser compatíveis com o motor que se deseja alimentar. Na Figura 14 pode-se ver um exemplo de driver do fabricante Akiyama modelo MA860m com capacidade até 7,2 A.

Para compreender um pouco mais sobre os sinais emitidos por esses *drivers*, faz-se necessário analisar a forma com que são empregados.

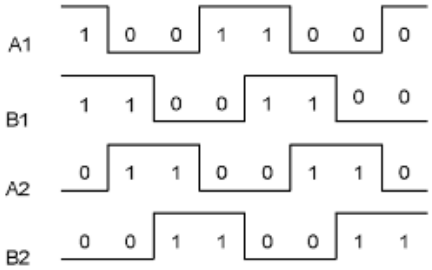


Figura 16 - Acionamento de um passo por vez, em duas bobinas.
Fonte: (Lima, Vilhaça,2012, p.236.)

Na Figura 15 é representado o sinal de uma bobina energizada por vez, gerando o acionamento de um passo. Essa é a opção que apresenta menos torque e mais economia.

Quando se deseja obter um torque maior, com a mesma aplicação,

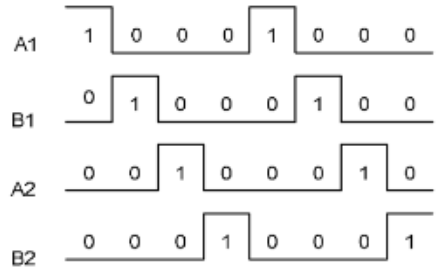


Figura 15 - Acionamento de um passo por vez, em uma bobina.
Fonte: (Lima, Vilhaça,2012, p.236.)

pode-se empregar o acionamento simultâneo de duas bobinas, o que acarretará no dobro da corrente e aumento do torque, mantendo o acio-

namento de um passo por vez. O exemplo desse tipo de acionamento pode ser visto na Figura 16.

Na necessidade de uma maior precisão pode-se utilizar acionamento simultâneo de duas bobinas intercalando com o acionamento de apenas uma bobina, caracterizando meio passo por acionamento. Na Figura 17 é possível analisar como seria o sinal binário.

2.5 Plataforma de hardware de controle

A plataforma de hardware de controle é a interface entre os sensores e atuadores de um robô, com possiblidade de ser programada através de softwares compatíveis, utilizando linguagens de programações conhecidas como: C++, *Ladder*, *Assembly*, entre outras específicas desenvolvidas pelo fabricante.

Para o desenvolvimento desse projeto poderiam ser utilizadas duas plataformas de hardware: CLP (Controlador Lógico Programável) ou placa de prototipagem eletrônica (microcontroladores).

Desta forma surgiu a necessidade de entender as duas plataformas de controle e conhecer as suas diferenças. Segundo estudo comparativo entre CLP e microcontrolador em um elevador de baixa complexidade publicado pela revista da FATEC (Faculdade de Tecnologia de Garça), a estrutura de um microcontrolador é semelhante a de um CLP, que é

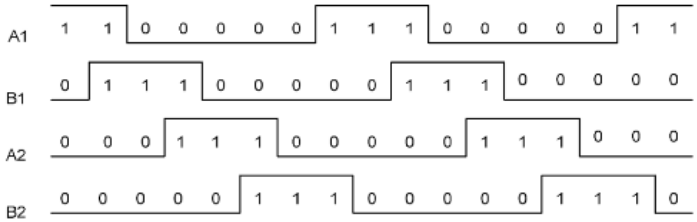


Figura 17 - Acionamento de meio passo por vez, maior precisão.
Fonte: (Lima, Vilhaça,2012, p.236.)

composta de *CPU*, memórias e portas *I/O*, com a única diferença sendo o sistema de *clock* e periféricos.

> *O sistema de clock é responsável pelo fornecimento da frequência em que o CPU trabalha, que tem por função controlar o sequenciamento das instruções da CPU. Já os periféricos são circuitos auxiliares que possibilitam o controle de dispositivos, tais como, contadores, conversores analógico/digital*

e digital/analógico, temporizadores e entre outros. (Tófoli, Higa, Mancuso,2014, Vol.4).

Na pesquisa para encontrar quais plataformas seriam compatíveis, o objetivo principal era encontrar soluções que apresentassem ótimo custo benefício. Verificou-se então que existem placas de prototipagem eletrônica de baixo custo como por exemplo a placa Arduino Uno que utiliza um microcontrolador ATmega328P, desenvolvida especialmente para projetos rápidos e de baixa a média complexidade. Esta placa tem característica *open source* e pode ser usada por qualquer pessoa. Outra alternativa analisada foi a possibilidade de uso de um CLP de mais baixo custo do fabricante nacional WEG, modelo CLIC 02, compacto e de fácil programação.

Como visto anteriormente no estudo comparativo feito pela FATEC, dois pontos devem ser levados em consideração: velocidade de trabalho "*clock"* e circuitos auxiliares de controle chamados de "periféricos".

A CPU da placa Arduino trabalha com *clocks* de 16 MHz, frequência que atenderia tranquilamente a aplicação, já o Clic 02 não especifica no manual a frequência da *CPU* mais cita a velocidade de processamento como sendo de 10 ms por ciclo.

Quanto aos periféricos o Click 02 comporta apenas uma saída de frequência rápida que trabalha com 1 kHz, utilizada para PWM, já o Arduino dispõe de 14 pinos digitais de entrada/saída configuráveis (dos quais seis podem ser utilizados como saídas *PWM*), e como nessa aplicação serão necessárias duas saídas *PWM*, no caso de usar o Clic 02, seriam necessárias duas unidades do CLP.

A Figura 18 mostra a placa Arduino e a Figura 19 o Clic02.

3 METODOLOGIA

Este projeto foi dividido em oito etapas para que sua realização fosse possível:

1. Escolha da aplicação;

Figura 18 - Arduino Uno e Genuíno – placa de prototipagem.
Fonte: (Arduino. Em: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 09 outubros 2016.)

Figura 19 - Clic02 WEG – CLP de baixo custo.
Fonte: (WEG. Em: < http://ecatalog.weg.net/files/wegnet/WEG-rele-programa-vel-clic-02-3rd-manual-portugues-br.pdf>. Acesso em: 31 de janeiro de 2016.)

2. Avaliação dos requisitos;
3. Estudo da cinemática;
4. Modelamento do robô (mecânica, elétrica);
5. Usinagem das peças;
6. Desenvolvimento da lógica de controle;
7. Montagem do robô;

3.1 Escolha da aplicação

A finalidade deste protótipo de robô no caso de uma aplicação real poderia ser definida como:

- Montagem de componentes em placa eletrônica
- Operações *pick-and-place* (operação de pegar e largar um objeto).

As aplicações acima citadas demandam alta velocidade e basicamente necessitam de operações em somente um plano. Deste modo chegamos à conclusão de que um robô serial seria um pouco complexo em relação ao robô de arquitetura paralela, logo selecionamos a última opção para desenvolvimento.

3.2 Avaliação dos requisitos

Com a aplicação definida deve-se avaliar as características do robô, como:

- Número de graus de liberdade;
- Área de trabalho necessária.

Para realizar as atividades de montagem de componentes e *pick-and-place* precisamos de movimentos no plano cartesiano (x, y) e atuação na cota (z). Para estas ações são necessários três graus de liberdade, porém utilizaremos somente o movimento de translação, isto é, no plano *xy*, desconsiderando a atuação no eixo (z), pois o foco é a cinemática do mecanismo de cinco barras e desta forma não será construído o mecanismo de atuação do eixo (z).

A área de trabalho está diretamente ligada à aplicação em que este robô será inserido. Por se tratar de um projeto didático a área de trabalho será limitada pelas dimensões físicas do robô.

3.3 Estudo da cinemática

De acordo com *Robot Mechanics*, de Dr. Robert L. Williams II, existem duas análises diferentes para mecanismos paralelos, sendo:

- Cinemática direta;
- Cinemática inversa.

Para cada uma das análises devemos escolher o método para obter o modelo matemático. Todas as análises foram feitas baseadas em modelo de robô de arquitetura paralela planar com dois graus de liberdade e configuração de um mecanismo de cinco barras.

O robô de arquitetura paralela planar com dois graus de liberdade e configuração de um mecanismo de cinco barras assemelha-se ao mecanismo de quatro barras de um grau de liberdade. Este possui uma barra extra e uma junta de revolução, fornecendo dois graus de liberdade ao invés de um grau de liberdade. Portanto, temos duas variáveis independentes de entrada, fixas ao solo, como mostra a Figura 20:

A designação **RRRRR** significa que existem cinco juntas de revolução neste robô paralelo, sendo a primeira e a última junta de revolução, fixas ao solo, controladas por motores independentes.

O mecanismo de cinco barras tem um total de cinco barras rígidas (quatro se movem e uma é o solo) e cinco juntas de revolução, cada uma

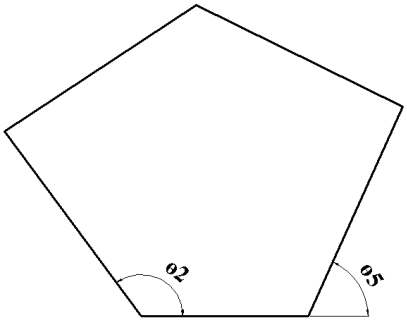


Figura 20 - Robô de arquitetura paralela planar com dois graus de liberdade e configuração de um mecanismo de cinco barras RRRRR.
Fonte: Autores

com um grau de liberdade. Logo a mobilidade deste sistema é:

$$M = 3(N-1) - 2J_1 - 1J_2 \quad \textbf{(0)}$$

$$M = 3(5-1) - 2(5) - 1(0) \quad \textbf{(0)}$$

$$M = 2 \quad \textbf{(0)}$$

Sendo $M>1$, o mecanismo de cinco barras planar pode ser considerado um robô, e também pelo fato de possuir dois trajetos a partir do solo até o ponto de interesse.

O ponto de interesse pode ser localizado no ponto “B, localizado na junta de revolução das barras 3 e 4, conforme Figura 21.

3.3.1 Cinemática direta

De acordo com a Figura 21 a barra 1 é fixa ao solo. Todos os ângulos θ_1 a θ_5 são absolutos, medidos a partir do eixo x no sentido anti-horário.

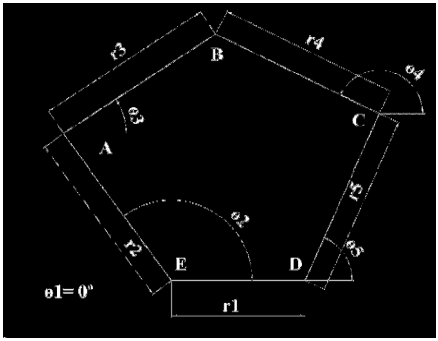


Figura 21 – Distribuição das barras e ângulos do mecanismo
Fonte: Autores

Os valores de $r1, r2, r3, r4, r5$ e θ_1 (θ_1 é o ângulo da barra r_1 em relação ao eixo x , mais os ângulos de entrada θ_2 e θ_5 são fornecidos, logo precisamos descobrir o ponto de interesse $\underline{B}=\left\{B_x \quad B_y\right\}^T$ e os ângulos θ_3 e θ_4 .

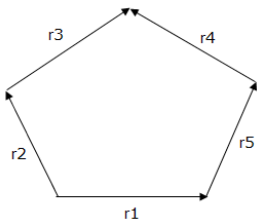


Figura 22 – Sentido dos vetores
Fonte: Autores

Conforme Figura 22, podemos começar a análise desenhando os vetores do mecanismo a partir da origem com sentido voltado para o ponto de interesse.

A partir do desenho podemos obter a equação para os trajetos até o ponto de interesse:

$$\underline{r}_2+\underline{r}_3=\underline{r}_1+\underline{r}_5+\underline{r}_4 \qquad \textbf{(0)}$$

Separamos a Equação (4) em componentes XY e seus componentes escalares:

$$r_2 c_2+r_3 c_3=r_1 c_1+r_5 c_5+r_4 c_4 \qquad \textbf{(0)}$$
$$r_2 s_2+r_3 s_3=r_1 s_1+r_5 s_5+r_4 s_4 \qquad \textbf{(0)}$$

Onde as seguintes abreviações foram utilizadas:

$$c_i=\cos\theta_i$$
$$s_i=sen\,\theta_i$$
$$i=1,2,3,4,5$$

As equações (5) e (6) podem ser resolvidas analiticamente, semelhante ao mecanismo de um grau de liberdade com quatro barras. Logo reescrevemos as equações (5) e (6) isolando o ângulo θ_3 :

$$r_3 c_3=r_1 c_1+r_5 c_5+r_4 c_4-r_2 c_2=a+r_4 c_4-r_2 c_2 \qquad \textbf{(0)}$$
$$r_3 s_3=r_1 s_1+r_5 s_5+r_4 s_4-r_2 s_2=b+r_4 s_4-r_2 s_2 \qquad \textbf{(0)}$$

onde:

$$a=r_1 c_1+r_5 c_5$$
$$b=r_1 s_1+r_5 s_5$$

Então elevamos e adicionamos as equações (7) e (8) e eliminamos θ_3 usando a identidade trigonométrica $\cos^2\theta_3+sen^2\theta_3=1$. Isto resulta em uma equação transcendental não linear com apenas uma variável θ_4 :

$$E\cos\theta_4 + F\,sen\,\theta_4 + G = 0 \quad \textbf{(0)}$$

onde:

$$E = 2r_4\left(a - r_2 c_2\right) \quad \textbf{(0)}$$
$$F = 2r_4\left(b - r_2 c_2\right) \quad \textbf{(0)}$$
$$G = a^2 + b^2 = r_2^2 - r_3^2 + r_4^2 - 2r_2\left(a c_2 + b s_2\right) \quad \textbf{(0)}$$

Podemos resolver esta equação para θ_4 usando a substituição de meia tangente.

Se definirmos $t = \tan\frac{\theta_4}{2}$ então temos $\cos\theta_4 = \frac{1-t^2}{1+t^2}$ e $sen\,\theta_4 = \frac{2t}{1+t^2}$.

Utilizando a substituição de meia tangente na Equação (9):

$$E\left(\frac{1-t^2}{1+t^2}\right) + F\left(\frac{2t}{1+t^2}\right) + G = 0 \quad \textbf{(0)}$$
$$E\left(1-t^2\right) + F\left(2t\right) + G\left(1+t^2\right) = 0 \quad \textbf{(0)}$$
$$\left(G-E\right)t^2 + \left(2F\right)t + \left(G+E\right) = 0 \quad \textbf{(0)}$$

Podemos resolver a Equação (15) utilizando a fórmula quadrática:

$$t_{1,2} = \frac{-F \pm \sqrt{E^2 + F^2 - G^2}}{G-E} \quad \textbf{(0)}$$

Resolvemos θ_4 invertendo a substituição de meia tangente original:

$$\theta_{41,2} = 2\tan^{-1}\left(t_{1,2}\right) \quad \textbf{(0)}$$

A Equação (17) possui duas soluções possíveis. Ambos os resultados estão corretos, sendo que existem duas condições possíveis, o pon-

to de interesse para cima e para baixo. Para encontrar θ_3 retornamos para a Equação (4) isolando θ_3 :

$$\theta_{31,2} = atan2\left(r_1 s_1 + r_5 s_5 + r_4 s_{41,2} - r_2 s_2, r_1 c_1 + r_5\right) \quad (0)$$

A Equação (18) calcula um valor único para θ_3 para cada valor de $\theta_{41,2}$.

O ponto "B" agora pode ser encontrado, sendo que θ_3 e θ_4 são conhecidos. Desta forma é possível encontrar dois valores para "B", novamente correspondendo ao ponto de interesse para cima ou para baixo. As duas equações abaixo resultarão nos mesmos valores para "B".

$$\underline{B} = \underline{r}_2 + \underline{r}_3 \quad (0)$$

$$\begin{Bmatrix} B_x \\ B_y \end{Bmatrix}_{1,2} = \begin{Bmatrix} r_2 c_2 + r_3 c_{31,2} \\ r_2 s_2 + r_3 s_{31,2} \end{Bmatrix} \quad (0)$$

$$\underline{B} = \underline{r}_1 + \underline{r}_5 + \underline{r}_4 \quad (0)$$

$$\begin{Bmatrix} B_x \\ B_y \end{Bmatrix}_{1,2} = \begin{Bmatrix} r_1 c_1 + r_5 c_5 + r_4 c_{41,2} \\ r_1 s_1 + r_5 s_5 + r_4 s_{41,2} \end{Bmatrix} \quad (0)$$

3.3.2 Cinemática inversa

Temos os valores de $r1, r2, r3, r4, r5, \theta_1$ mais as coordenadas do ponto de interesse $\underline{B} = \begin{Bmatrix} B_x & B_y \end{Bmatrix}^T$, logo precisamos encontrar os ângulos dos atuadores θ_2 e θ_5 mais os ângulos das juntas passivas θ_3 e θ_4 .

O ângulo dos atuadores θ_2 e θ_5 podem ser encontrados independentemente através das equações apresentadas na seção Cinemática Direta 3.3.1. O ângulo passivo θ_3 pode ser encontrado junto de θ_2 , e o ângulo passivo θ_4 pode ser encontrado junto de θ_5 .

$$\underline{B} = \underline{r}_2 + \underline{r}_3 \quad (0)$$

$$\begin{Bmatrix} B_x \\ B_y \end{Bmatrix} = \begin{Bmatrix} r_2 c_2 + r_3 c_3 \\ r_2 s_2 + r_3 s_3 \end{Bmatrix} \quad (0)$$

$$\underline{B} = \underline{r}_1 + \underline{r}_4 + r_5 \quad (0)$$

$$\begin{Bmatrix} B_x \\ B_y \end{Bmatrix} = \begin{Bmatrix} r_1 c_1 + r_5 c_5 + r_4 c_4 \\ r_1 s_1 + r_5 s_5 + r_4 s_4 \end{Bmatrix} \quad (0)$$

Isolando os ângulos passivos desconhecidos para cada uma das equações XY escalares acima , tem-se:

$$r_3 c_3 = B_x - r_2 c_2 \quad (0)$$
$$r_3 s_3 = B_y - r_2 s_2 \quad (0)$$
$$r_4 c_4 = B_x - r_1 c_1 - r_5 c_5 \quad (0)$$
$$r_4 s_4 = B_y - r_1 s - r_5 s_5 \quad (0)$$

Elevando ao quadrado e somando as duas equações separadamente eliminamos as juntas passivas desconhecidas:

$$E_2 \cos\theta_2 + F_2 sen\theta_2 + G_2 = 0 \quad (0)$$
$$E_2 = -2 r_2 B_x \quad (0)$$
$$F_2 = -2 r_2 B_y \quad (0)$$
$$G_2 = r_2^2 - r_3^2 + B_x^2 + B_y^2 \quad (0)$$

$$E_5 \cos\theta_5 + F_5 sen\theta_5 + G_5 = 0 \quad (0)$$
$$E_5 = 2 r_5 \left(-B_x + r_1 c_1\right) \quad (0)$$
$$F_5 = 2 r_5 \left(-B_y + r_1 s_1\right) \quad (0)$$
$$G_5 = r_1^2 + r_5^2 - r_4^2 + B_x^2 + B_y^2 - 2 r_1 \left(B_x c_1 + B_y s_1\right) \quad (0)$$

Resolvendo independentemente para os dois atuadores usando a substituição de meia tangente duas vezes, obtém-se:

$$t_{21,2} = \frac{-F_2 \pm \sqrt{E_2^2 + F_2^2 - G_2^2}}{G_2 - E_2} \quad (0)$$

$$t_{51,2}=\frac{-F_5 \pm \sqrt{E_5^2+F_5^2-G_5^2}}{G_5-E_5} \quad (0)$$

$$\theta_{21,2}=2\tan^{-1}(t_{21,2}) \quad (0)$$

(0)

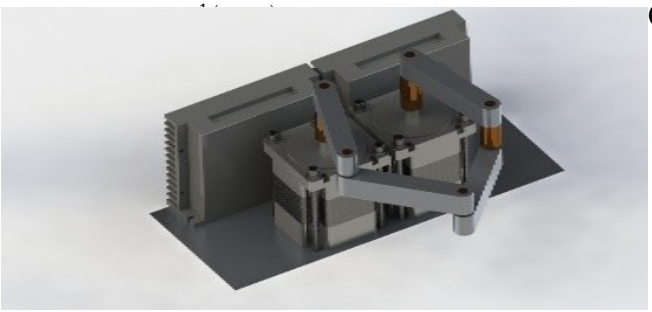


Figura 23 – Prévia do modelo feito em CAD
Fonte: Autores

Agora retornamos para as equações escalares XY originais para resolvermos as juntas passivas desconhecidas:

$$\theta_3=atan2(B_y-r_2 s_2, B_x-r_2 c_2) \quad (0)$$
$$\theta_4=atan2(B_y-r_1 s_1-r_5 s_5, B_x-r_1 c_1-r_5 c_5) \quad (0)$$

Existem dois valores possíveis (trajetos esquerdo e direito) para os ângulos desconhecidos θ_2 e θ_3 ; independentemente existem dois valores possíveis (trajetos esquerdo e direito) para os ângulos desconhecidos θ_5 e θ_4 . No entanto, existem quatro soluções possíveis para o problema da Cinemática Inversa para o robô planar de cinco barras.

3.4 Modelamento do robô

Para validar o modelo matemático deve-se pelo menos construir o modelo do robô em CAD para definir as dimensões de seus componentes. O objetivo foi ter um modelo compacto utilizando materiais disponibilizados pelo IFSC e que os autores já possuíam, sendo este o critério que limitaria o tamanho, materiais utilizados e desempenho do robô. Os

desenhos feitos para o robô estão nos apêndices deste trabalho. A Figura 23 exibe uma prévia do que seria construído

3.5 Usinagem das peças

A partir das dimensões obtidas no modelamento foram usinadas as peças do robô em máquinas convencionais, obedecendo as margens de tolerância de dois décimos de milímetro.

O material utilizado na maioria das peças foi o alumínio especificado em projeto, devido a sua leveza e facilidade de usinagem. Na Figura 24 é possível ver a usinagem de desbaste de material em uma das barras "braços" do robô.

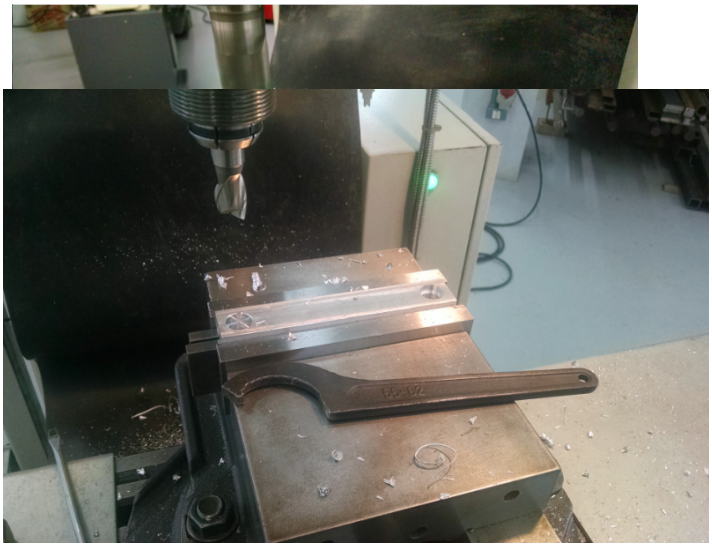


Figura 25 – Barras – usinagem de furos e rebaixos.
Fonte: Autores

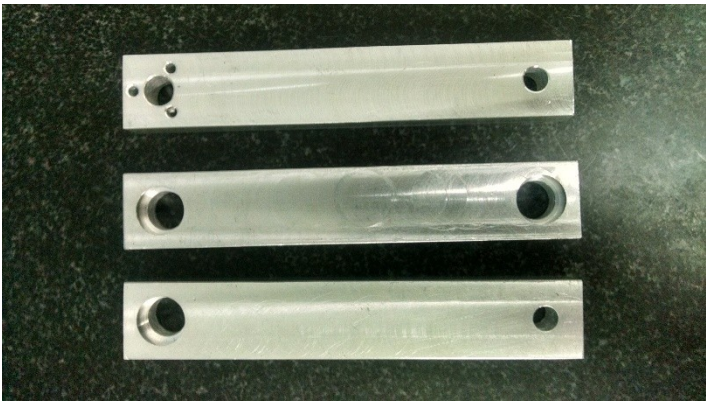


Figura 26 – Elementos – semiacabados.

Após desbastar os excessos de material foram realizados os furos e rebaixos para encaixe dos rolamentos vistos nas Figura 25 e Figura 26.

Figura 27 – Painel de controle – função múltiplos furos.
Fonte: Autores

Afim de retirar os cantos vivos das barras, utilizamos a função múltiplos furos da fresadora dessa forma foi possível realizar o arredondamento das barras.

Na Figura 27 pode-se ver a foto do painel de controle da fresadora na execução do décimo furo, mostrando as coordenadas em (x) e (y).

Na Figura 28 a usinagem do canto das barras em andamento.

Para suportes das barras principais foram usinadas as duas buchas de fixação entre as barras e os motores, modeladas em projeto, também

Figura 29 – Bucha – cortando a rosca.
Fonte: Autores

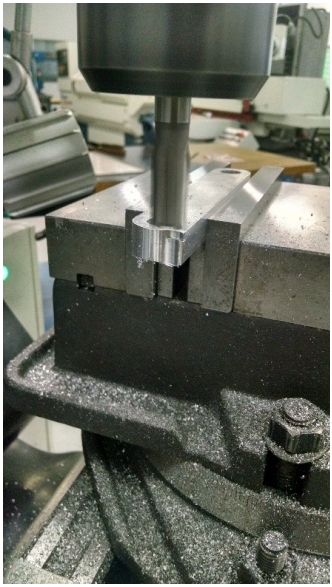


Figura 28– Semicírculo feito na usinagem.
Fonte: Autores

feitas em alumínio. Na Figura 29 pode ser observado a execução manual de cortar a rosca que servirá para fixação das barras.

3.6 Desenvolvimento da lógica de controle

Para que o robô funcione corretamente deve ser criada uma lógica de controle que garanta não somente o posicionamento correto do ponto de interesse como também leve em consideração as suas limitações físicas.

3.6.1 Motores

Os motores disponibilizados pelo IFSC são os modelos AK85H3.75-1.8 (Figura 30) e AK85H8_3.36-1.8 (Figura 31), ambos são motores de passo com a resolução de 200 passos por revolução.

3.6.2 Drivers

Os drivers utilizados no projeto são diferentes na questão de corrente máxima e micro passos. O modelo MA860m, observado na Figura 32 possui a tensão de trabalho de 60 VAC ou 80 VDC com capacidade de corrente de 7,2 A e consegue dividir uma revolução em até 40 mil passos por revolução (PPR) sendo que o outro driver Figura 33 possui a tensão de trabalho de 60 VAC ou 80 VDC com capacidade de corrente de 2,2 A e divide no máximo uma revolução em 6400 mil passos. Mesmo tendo essas diferenças é possível trabalhar com os dois drivers em uma mesma configuração, que neste caso foi adotada a de 6400 passos por revolução. A seguir os motores na Figura 30 e Figura 31.

Figura 30 – Motor AK85H3.75-1.8
Fonte: Autores

Figura 31 – Motor AK85H8_3.36-1.8
Fonte: Autores

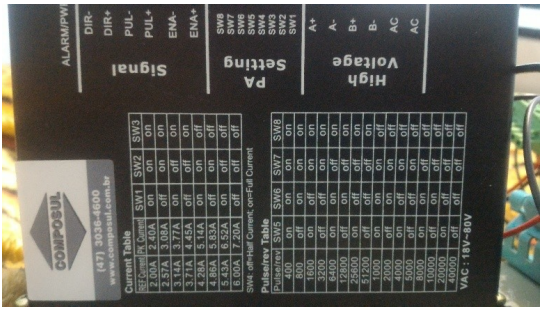


Figura 32 – Driver MA860m, resolução máxima de 40000 ppr
Fonte: Autores

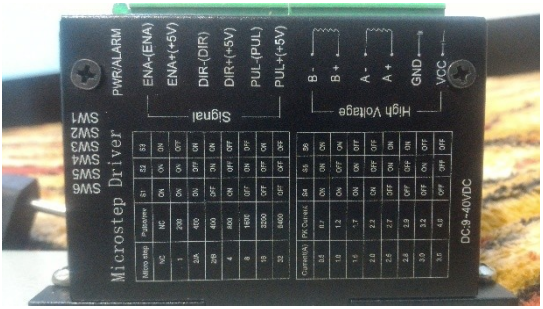


Figura 33 – Driver com resolução máxima de 6400 ppr
Fonte: Autores

3.6.3 Controlador e interface

Para controlar os drivers dos motores de passo foi escolhido o Arduino Uno, que possui seu próprio ambiente de desenvolvimento integrado (*IDE, do inglês Integrated Development Environment*) visto na Figura 34, proporcionando o acesso a várias bibliotecas, sendo estas padrões ou de terceiros.

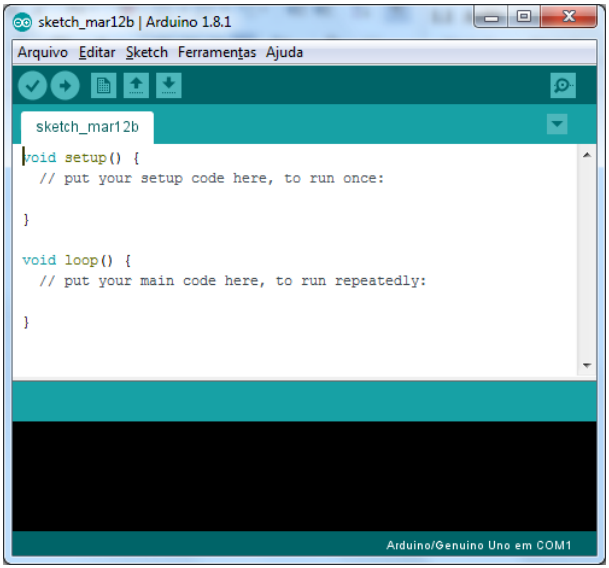


Figura 34 – Ambiente de desenvolvimento integrado Arduino IDE
Fonte: Autores

3.6.4 Programação

3.6.4.1 Configurações dos motores

Para o controle de motores de passo há uma biblioteca chamada *AccelStepper*, que é uma interface orientada a objetos para o controle de motores e *drivers* de passo de 2, 3 e 4 pinos. O diferencial desta biblioteca em relação à biblioteca padrão é:

- O suporte de aceleração e desaceleração;
- Múltiplos motores funcionando simultaneamente, com o controle de passos independente para cada motor;
- Funções que não utilizam o comando *delay()*, ou seja, não bloqueia o código;
- Velocidades muito baixas, entre outros atributos.

Para utilizarmos a biblioteca precisamos declará-la no programa conforme o Erro: Origem da referência não encontrada:

```
// Biblioteca motor de passo
#include <AccelStepper.h>
```

Código – Inclusão da biblioteca

Também devemos declarar quais pinos do controlador irão ser utilizados:

```
// Configuração motores
AccelStepper stepper1(1, 4, 3); // Pinos do arduino em que o
driver está conectado
AccelStepper stepper2(1, 7, 6); // Pinos do arduino em que o
driver está conectado
```

Código 1 – Pinos utilizados pela biblioteca

O número "1" indica o modo de controle da biblioteca, que no caso é DRIVER, ou seja, utilizamos um *driver* para receber os dados do Arduino e transferi-los para o motor, o número "4" representa o pino que irá controlar os pulsos e o número "3" o pino de controle de direção do motor.

Para facilitar a programação criamos variáveis referentes a estes pinos de controle (Erro: Origem da referência não encontrada):

```
// Configuração do driver
const int ena1 = 2; // Pino enable do driver 1
const int dir1 = 3; // Pino direção do driver 1
const int pul1 = 4; // Pino pulso do driver 1
const int ena2 = 5; // Pino enable do driver 2
const int dir2 = 6; // Pino direção do driver 2
```

Código – Configuração do driver

As funções *setMaxSpeed()* e *setAcceleration()* configuram qual será a máxima velocidade e aceleração de cada motor, bastando apenas referenciá-las ao objeto *stepper* criado anteriormente.

```
stepper1.setMaxSpeed(6400.0);     //Máxima velocidade em passos por segundo
stepper1.setAcceleration(6400.0); //Máxima aceleração em passos por segundo
stepper2.setMaxSpeed(6400.0);     //Máxima velocidade em passos por segundo
stepper2.setAcceleration(6400.0); //Máxima aceleração em passos por segundo
```

Código – Configuração de velocidade e aceleração dos motores

```
pinMode(ena1, OUTPUT);   //Habilita o pino enable como saída
pinMode(dir1, OUTPUT);   //Habilita o pino direção como saída
pinMode(pul1, OUTPUT);   //Habilita o pino pulso como saída

pinMode(ena2, OUTPUT);   //Habilita o pino enable como saída
pinMode(dir2, OUTPUT);   //Habilita o pino direção como saída
pinMode(pul2, OUTPUT);   //Habilita o pino pulso como saída

digitalWrite(ena1, HIGH); //Enable em nível alto
digitalWrite(dir1, LOW);  //Direção em nível alto
digitalWrite(pul1, LOW);  //Pulso em nível alto

digitalWrite(ena2, HIGH); //Enable em nível alto
digitalWrite(dir2, LOW);  //Direção em nível alto
digitalWrite(pul2, LOW);  //Pulso em nível alto
```

Código - Configuração dos pinos

Na função de configuração do programa temos que habilitar os pinos declarados.

Um detalhe a ser observado no Erro: Origem da referência não encontrada é que, para que o programa inicie com o driver desligado por motivos de segurança, é necessário que o nível lógico do pino ENABLE seja alto.

Outros comandos utilizados pela biblioteca estão listados abaixo (Erro: Origem da referência não encontrada):

```
stepper1.run();
stepper1.currentPosition();
stepper1.setCurrentPosition();
stepper1.moveTo();
```

Código - Outras configurações

- *run()*: Inserido dentro da função principal, roda continuamente;

- *currentPosition()*: Informa a posição atual do motor, em passos, desde o início do funcionamento;
- *setCurrentPosition()*: Define a nova posição, em passos, do motor;
- *moveTo()*: Move o motor para a posição desejada.

3.6.4.2 Comunicação

Com as configurações iniciais definidas, foi necessário estudar um meio de comunicação entre o Arduino e o usuário. Uma alternativa que o Arduino oferece é a possibilidade do uso do padrão de comunicação serial, utilizando a própria biblioteca fornecida pela *IDE*.

Para iniciar a comunicação é suficiente a utilização do comando *Serial.begin()* e a respectiva definição da taxa de transmissão de dados, Erro: Origem da referência não encontrada) A *IDE* do Arduino possui uma janela chamada "Monitor serial", que permite a leitura e escrita de dados via comunicação serial sem a necessidade de outra aplicação.

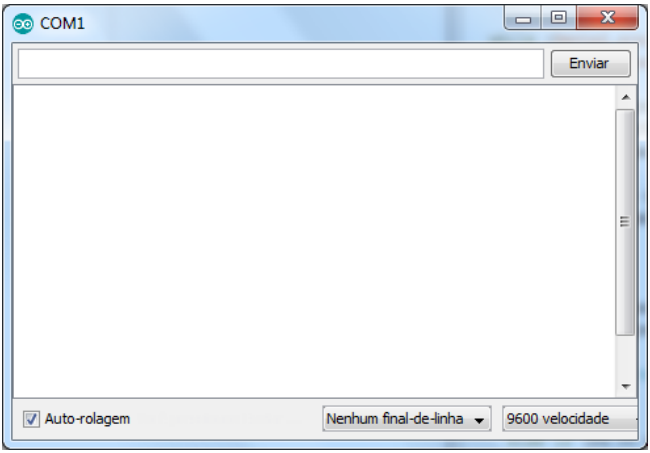


Figura 35 – Monitor serial.
Fonte: Autores

Para que o programa execute corretamente suas funções recebidas pela serial foi utilizado um método de codificação dos comandos enviados e recebidos pela serial, conforme código abaixo:

```
Serial.begin(9600); //Taxa de transmissão da serial
```

Código - Configuração da serial

```
// Configuração serial
const byte numCarac = 32;      //Máximo de caracteres
char caracRec[numCarac];       //Vetor que armazena valores
recebidos
char caracTemp[numCarac];      //Vetor temporário para análise

// Variáveis que armazenam os dados analisados
char tipoMens[numCarac] = {0};
char funcMens[numCarac] = {0};
int dadosMens = 0;
```

Código - Padrão de codificação da serial

Código 2 - Continuação codificação da serial

```
void recComDelim() {
  static boolean recEmProg = false;
  static byte ndx = 0;
  char delimIni = '<';
  char delimFin = '>';
  char leSerial;

  while (Serial.available() > 0 && novoDado == false) {
    leSerial = Serial.read();

    if (recEmProg == true) {
      if (leSerial != delimFin) {
        caracRec[ndx] = leSerial;
```

Primeiro foi definido o máximo número de caracteres que podem ser recebidos/enviados via serial (Código 2, numCarac), depois estes valores são armazenados em um vetor (caracRec[]) e por último copiamos estes valores para um vetor temporário que é utilizado na análise dos dados recebidos (caracTemp[]). Também temos variáveis para armazenar

os dados que serão analisados e processados (tipoMens[] – Item, funcMens[] – Subitem, dadosMens[] – Valores)

Para este programa o padrão utilizado foi "<XX,XX,XXXX>" (Código 2) onde:

- "<" e ">": São os delimitadores dos dados, "<" indica o início e ">" o final dos dados recebidos/enviados;
- XX: Identificador do item a ser executada ação;
- XX: Identificador do subitem;
- XXXX: Identificador de valores a serem interpretados pelo programa em formato hexadecimal.

Toda informação que não esteja dentro deste padrão de dados não

```
void analisaDado() {      // Divide os dados em partes
  char * strtokIndx;      // Usado por strtok() como índice
  strtokIndx = strtok(caracTemp, ","); // Continua a partir da
última chamada
  strcpy(tipoMens, strtokIndx);
  strtokIndx = strtok(NULL, ",");
  strcpy(funcMens, strtokIndx);
  strtokIndx = strtok(NULL, ",");
  dadosMens = atoi(strtokIndx);
}
```

Código - Continuação codificação da serial

é interpretada pelo programa.

A rotina é iniciada toda vez que há algum dado na serial (*Serial.available()*) e que atende aos requisitos do padrão adotado. Por fim, os dados são separados (função analisaDado()) para que pudéssemos comparar o item, subitem e valores conforme a ação necessária.

3.6.4.3 Interface usuário

Para que o controle do robô seja didático foi criado um software na IDE Visual Studio, com a linguagem de programação C#, onde é possível controlar os motores independentemente ou através de uma coordenada no plano, como também informando a posição atual do ponto de interesse e ângulos do robô. Esta plataforma oferece o modo de programação por código (Figura 36) como também a criação de executáveis pelo modo gráfico (Figura 37).

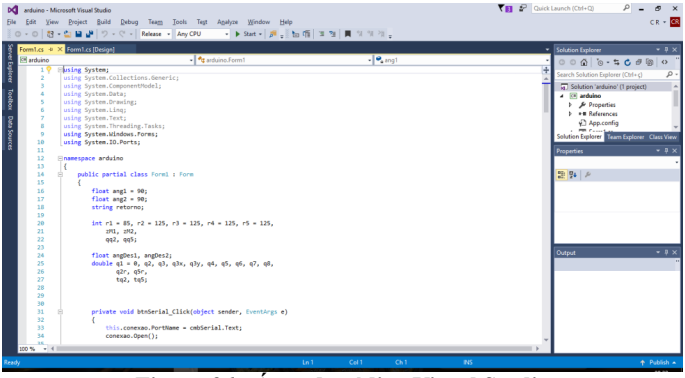


Figura 36 - Área de código Visual Studio
Fonte: Autores

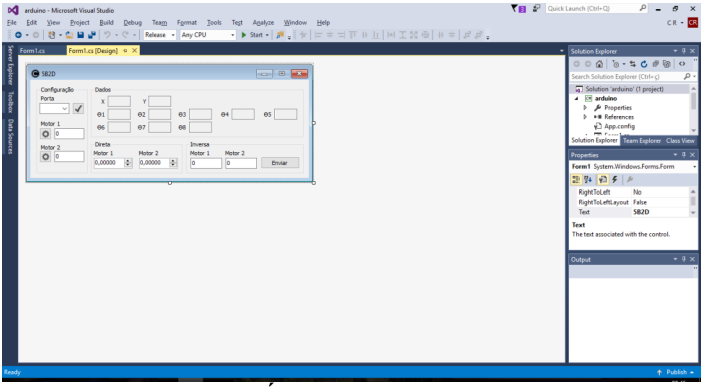


Figura 37 - Área de programação gráfica
Fonte: Autores

O fato deste projeto não possuir encoder, seja ele incremental ou absoluto, levou à criação de um método de zeramento dos motores, sendo feito através do programa de interface com o usuário no momento do primeiro funcionamento, garantindo seu correto posicionamento até a sua desenergização.

Assim como no Arduino aqui também há a necessidade de ser iniciada a comunicação serial através do comando *using System.IO.Ports;* , que habilita o controle de portas de entrada e saída de dados. O Erro: Origem da referência não encontrada habilita a comunicação serial pelo computador, onde precisamos declarar o objeto **conexão**, preencher uma *combobox* com as portas seriais disponíveis. Após o usuário selecionar a porta serial atribuída ao Arduino Uno a

```
SerialPort conexao = new SerialPort();
cmbSerial.DataSource = SerialPort.GetPortNames();
this.conexao.PortName = cmbSerial.Text;
conexao.Open();
```

Código - Continuação codificação da serial

conexão serial com o mesmo é aberta.

Os cálculos obtidos para a cinemática direta e inversa, dependendo do quadrante em que o ângulo de cada motor se encontra, podem apresentar valores negativos em que certas combinações podem ocasionar a colisão das partes móveis do robô. Para contornar esta situação foi criada uma lógica que trata cada combinação de ângulo dos motores (Erro: Origem da referência não encontrada).

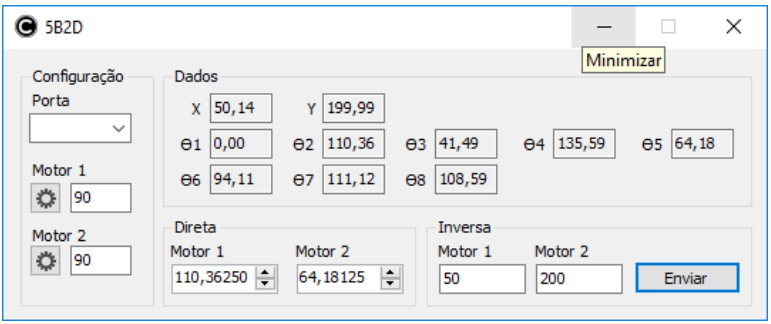


Figura 38 - Interface robô – usuário
Fonte: Autores

```
//Se menor que zero muda
            if (q2 < 0) q2 = 360 + q2;
            //Verifica quadrante q2
            if (0 < q2 && q2 < 90)    qq2 = 1;
            if (90 < q2 && q2 < 180)  qq2 = 2;
            if (180 < q2 && q2 < 270) qq2 = 3;
            if (270 < q2 && q2 < 360) qq2 = 4;
            q3 = Math.Atan2(By - r2 * Math.Sin(Math.PI * q2 / 180), Bx -
r2 * Math.Cos(Math.PI * q2 / 180)) * 180 / Math.PI;
            q5 = 2 * Math.Atan(t5) * 180 / Math.PI;
            tq5 = q5;
            //Se menor que zero muda
            if (q5 < 0) q5 = 360 + q5;
            //Verifica quadrante q5
            if (0 < q5 && q5 < 90) qq5 = 1;
            if (90 < q5 && q5 < 180) qq5 = 2;
            if (180 < q5 && q5 < 270) qq5 = 3;
            if (270 < q5 && q5 < 360) qq5 = 4;
            //Ângulo conforme combinação de quadrantes
            if (qq2 == 1 && qq5 == 3) q5 = tq5;
            if (qq2 == 1 && qq5 == 4) q5 = tq5;
            if (qq2 == 1 && qq5 == 1) q5 = tq5;
            if (qq2 == 1 && qq5 == 2) q5 = tq5;
            if (qq2 == 2 && qq5 == 1) q5 = tq5;
            if (qq2 == 2 && qq5 == 2) q5 = tq5;
            if (qq2 == 3 && qq5 == 2) q5 = tq5;
            if (qq2 == 4 && qq5 == 2) q5 = tq5;
```

Código 12 – Lógica para as combinações

3.7 Montagem do Robô

As peças do robô foram usinadas com cuidado para manter a tolerância em torno de dois décimos de milímetro com o intuito de facilitar a montagem, evitando ajustes desnecessários e provendo certa precisão no mecanismo. A estrutura de aspecto simples facilitou o processo de montagem, basicamente são quatro barras e duas buchas de fixação conforme a Figura 39.

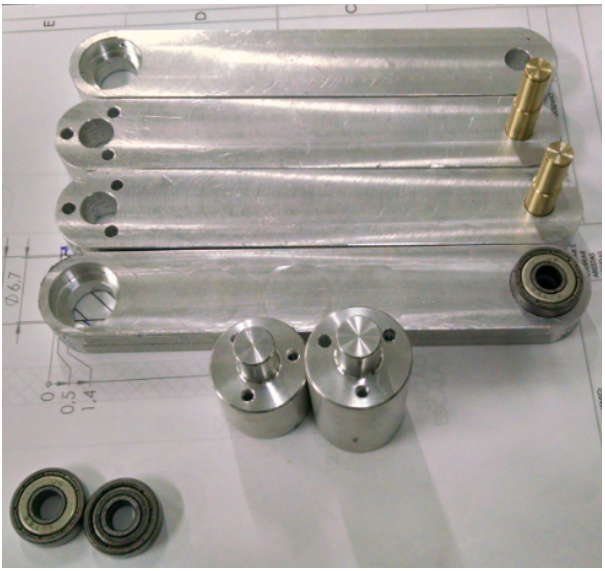


Figura 39 – Barras, buchas, pinos e rolamentos utilizados na montagem.
Fonte: Autores

O encaixe dos rolamentos foi realizado através de pressão com morsa de bancada, tendo o devido cuidado para não comprometer a qualidade dos mesmos e seguindo as tolerâncias indicadas pelo fabricante. Os pinos também foram encaixados sobre pressão com a ideia de manter a estrutura firme nos pontos de junção. Para auxiliar a fixação dos pinos as barras foram utilizados anéis de retenção de eixo, conforme previsto em projeto, Figura 40.

A fixação das buchas com as barras e os eixos do motor foi realizada com parafusos de cabeça cilíndrica com sextavado interno. Cada bucha possui três roscas de fixação das barras Figura 41 e uma rosca de fixação para utilizar no eixo do motor, conforme Figura 42.

Figura 41 – Fixação entre barra e bucha.
Fonte: Autores

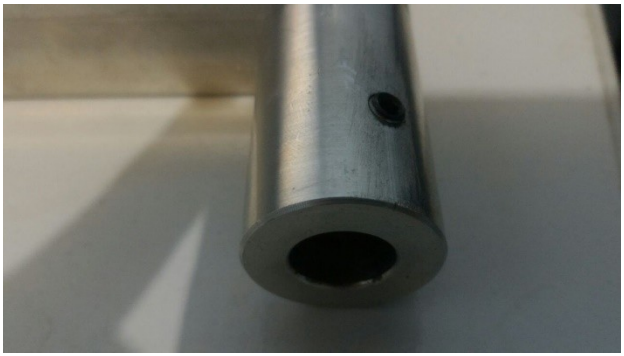


Figura 42 – Fixação entre bucha e eixo do motor.
Fonte: Autores

A montagem do mecanismo de cinco barras do robô paralelo planar, ainda sem sua base, pode ser vista na Figura 43. Depois de montado foi possível ver que o torque exigido aos motores será realmente leve, os rolamentos fizeram essa função muito bem.

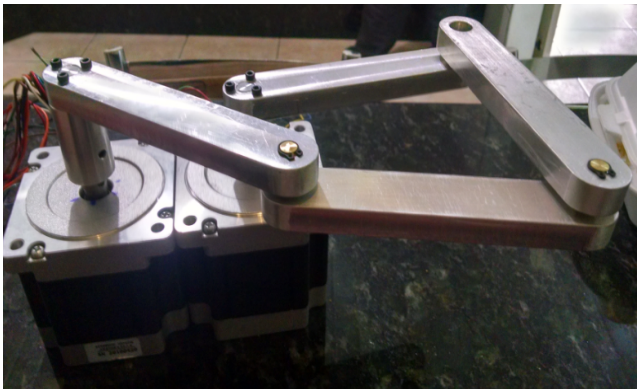


Figura 43 – Robô paralelo planar – montagem sem a base.
Fonte: Autores

Após montado o robô, facilitou-se a compreensão do plano de trabalho, com isso ficou mais fácil escolher a chapa de metal para confecção da base. Utilizou-se uma chapa de aço carbono 705x712 mm e espessura de 1,5 mm com dobras em três dos quatro lados aumentando a resistência para evitar a "flambagem". A chapa vista na Figura 44 possui pintura em bege Ral 7032.

Figura 44 – Base robô – furações e retoque de pintura.
Fonte: Autores

Para a fixação dos motores de passo foram utilizadas barras roscadas M6 que são do diâmetro aproximado dos furos de fixação do motor de passo, as barras podem ser vistas na Figura 45.

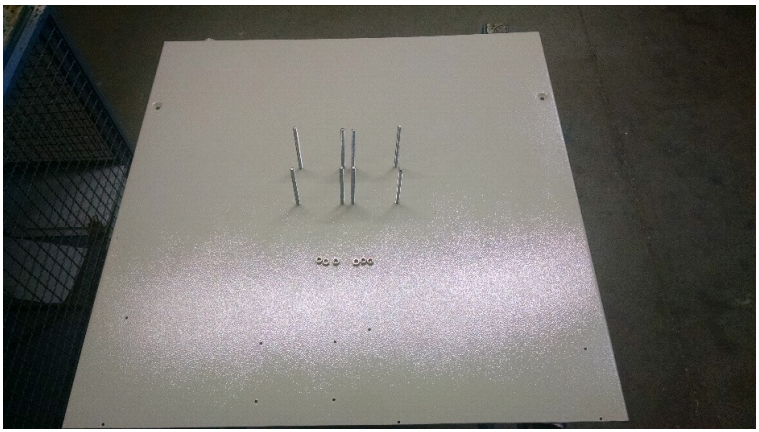


Figura 45 – Base robô – barras roscadas e porcas.
Fonte: Autores

Já para os demais componentes foi feita fixação com parafusos M4 roscados direto na chapa, como por exemplo na Figura 46 o driver de motor de passo.

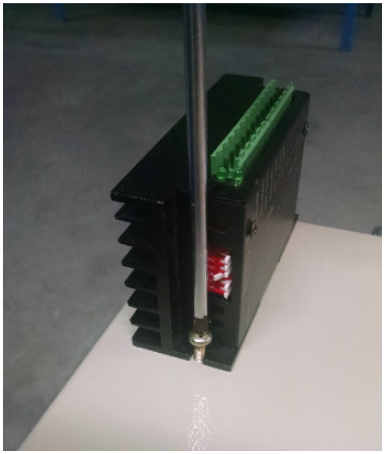


Figura 46 – Base robô – driver fixado com parafuso.
Fonte: Autores

Na Figura 47 é possível ver a fixação dos principais componentes a estrutura da base.

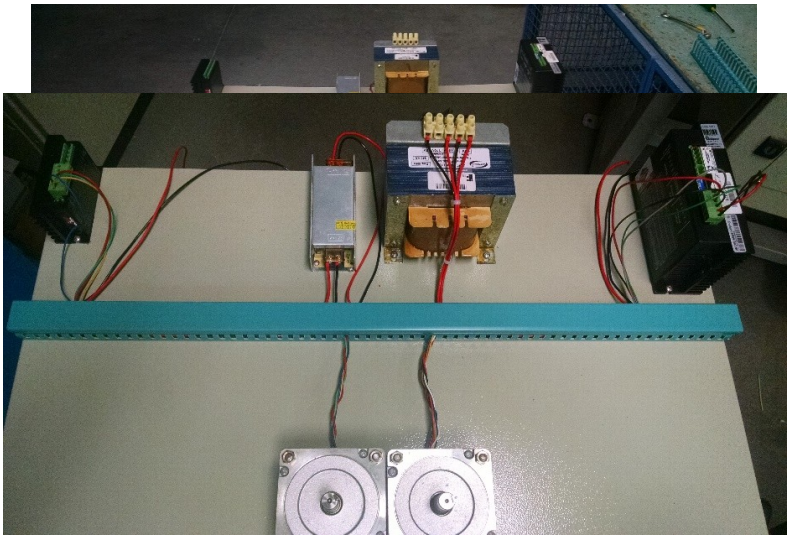


Figura 48 – Base robô – disposição da canaleta.
Fonte: Autores

Para facilitar a distribuição do cabeamento dos componentes, foi aderido ao uso de uma canaleta de cabos, muito utilizada em painéis elétricos, Figura 48 e 49.

Figura 49 – Base robô – canaleta azul petróleo 30x30mm.
Fonte: Autores

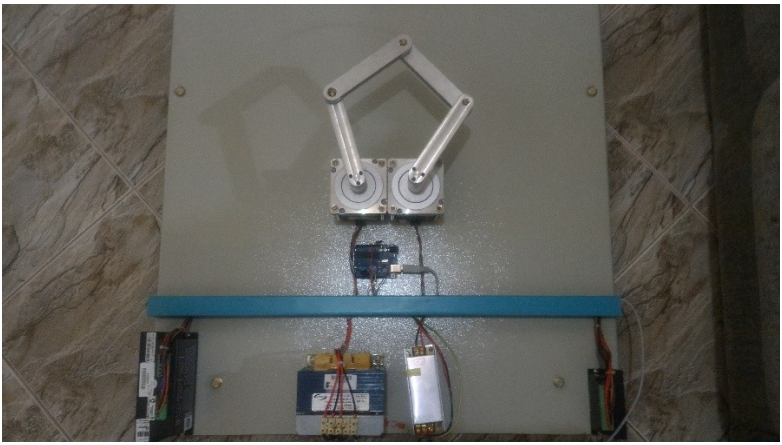


Figura 50 – Vista superior do robô.
Fonte: Autores

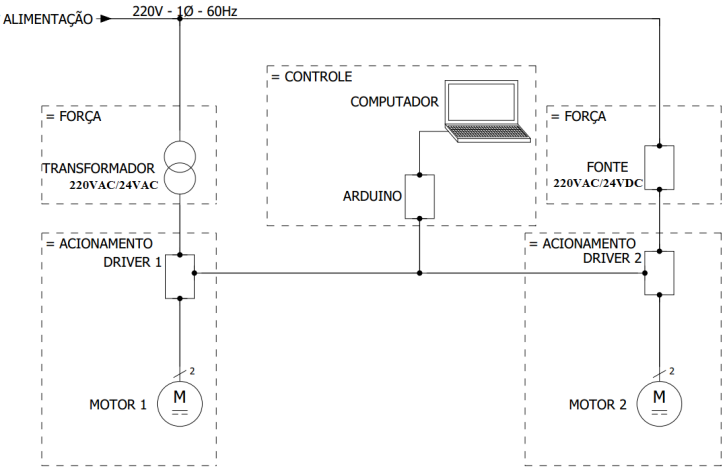


Figura 51 – Diagrama de construção do robô.
Fonte: Autores

4 CONCLUSÃO

4.1 Analise geral

A intenção desse projeto foi buscar uma abordagem didática aos leitores, trazendo a solução de um problema que envolvesse a maioria dos conhecimentos adquiridos durante o curso de tecnologia em mecatrônica industrial.

A escolha de desenvolver o protótipo de um robô planar didático atingiu essa expectativa, pois interage com os campos de elétrica, eletrônica, mecânica e programação.

4.2 Resultados obtidos

Na concepção deste protótipo os resultados obtidos foram predominantemente positivos, tendo apenas algumas alterações com intuito de otimizar o custo do projeto, não influenciando assim na capacidade ou funcionamento do mesmo.

A construção mecânica do robô planar na composição do mecanismo de cinco barras “RRRRR”, apresentou resistência e estabilidade. Isso foi possível por conta das tolerâncias utilizadas na usinagem das peças, também pela escolha e dimensão dos materiais utilizados, sendo o alumínio um material leve e resistente que atendeu as expectativas, possibilitando que os testes fossem realizados de maneira eficaz.

Os desafios após a montagem do protótipo se tornaram mais reais, pois evidenciaram a complexidade dos pontos de singularidade deste mecanismo, exigindo muito estudo e dedicação.

No quesito *hardwares,* esse protótipo foi equipado com *drivers* de diferentes tamanhos para cada motor, prezando o custo benefício estes foram reutilizados e apesar da diferença de potência, não apresentaram diferença de rendimento, visto que o esforço no eixo dos motores é muito leve, uma característica deste tipo de estrutura. A placa de controle Arduino Uno mostrou-se suficiente ao funcionar como interface de transmissão de dados entre o computador e os drivers de controle dos motores, já que os cálculos da cinemática direta e inversa são efetuados no computador e somente são transmitidos os dados necessários para a movimentação do mecanismo.

As linguagens de programação (Visual C# e C++), apesar de não serem de baixo nível, exigem um conhecimento mínimo de suas bibliotecas e possibilidades que ambas oferecem ao programador.

O protótipo apresentou certa oscilação no mecanismo ao passar perto dos pontos de singularidade, isso acontece principalmente por que

não está sendo controlado velocidade e aceleração dos motores, dessa forma ao chegar perto desses pontos o mecanismo começa a oscilar.

4.3 Objetivos alcançados

O conhecimento teórico é algo que se torna mais perceptivo quando aplicado na prática. Montar o protótipo do robô possibilitou que o conhecimento da área de cinemática fosse explorado com mais propriedade.

A fabricação das peças do robô foi realizada de acordo com o projeto. Durante a montagem da estrutura não houve a necessidade de ajustes mecânicos, dessa forma pode-se afirmar que o projeto foi bem elaborado.

A programação aplicada a esse protótipo foi desenvolvida nos softwares IDE Arduino e Visual Basic do Windows. Os dois softwares apresentam muitas ferramentas que possibilitam a realização do controle do mecanismo do robô. Através da placa de controle Arduino e dos drivers, os motores foram controlados com boa precisão.

Os testes com o protótipo foram realizados primeiramente utilizando a cinemática direta, movendo os eixos de cada motor de forma independente. Durante os testes foram encontradas as limitações da área de trabalho de acordo com a estrutura e situações de singularidade as quais devem ser evitadas. O teste da cinemática inversa foi a prova real das análises feitas durante os testes da cinemática direta, com o objetivo principal da ideia desse estudo, mover o ponto de interesse até o ponto desejado. Apesar de ser possível identificar colisões entre as peças móveis e as singularidades a resolução destes pontos fica fora do escopo deste trabalho.

Nos testes de trajetória o robô perde um pouco de sua estabilidade ao passar perto dos pontos de singularidade, que no caso acontecem

Figura 52 – Exemplo de singularidade em que o ângulo entre as barras é de

quando as duas barras que compartilham a mesma junta de rotação ficam com valor de ângulo 0º ou 180º entre si.

4.4 Melhorias

O protótipo alcançou os objetivos esperados no início do TCC, porém existem possibilidades de melhoria que podem ser executadas em trabalhos futuros, ampliando o conhecimento aplicado neste trabalho.

Abaixo seguem algumas sugestões de melhorias:

- Desenvolver um atuador para o deslocamento no eixo (z), trabalhando assim o terceiro grau de liberdade, com opção de poder simular um manipulador de operações *pick-and-place* e até mesmo realizar a ação de fazer o deslocamento de uma peça.
- Colocar sensores que monitorem o posicionamento do robô até o ponto "zero" inicial, incluindo uma rotina no programa para que ao iniciar o robô busque este ponto zero.
- Fazer uma programação capaz de possibilitar a passagem do mecanismo pelos pontos de singularidade, aumentando assim a área de trabalho.
- Criar uma interface de programação para o usuário, que possibilite o registro de coordenadas e as execute automaticamente quando solicitado.
- Realizar o controle da velocidade e aceleração dos motores de passo.

5 REFERÊNCIAS

AGOSTINI, ***International Organization for Standardization***, 2012.

BRIOT, Sébastien; KHALIL, Wisama. ***Dynamics of Parallel Robots from Rigid Bodies to Flexible Elements***. Springer Ed., 2015.

CRAIG, John J. ***Introduction_To_Robotics: mechanics and control***. 2nd ed. Addison-Wesley, 1989.

HESS-COELHO, Vitor neves; HARTMANN, Tarcísio Antônio. ***Desenvolvimento de uma máquina fresadora de arquitetura paralela***. São Paulo: Biblioteca24horas Seven System Internacional LTDA, 2014.

LIMA, C. B.; VILLAÇA, M.V.M. ***Avr e arduino técnicas de projeto.*** Florianópolis: Sistemas de bibliotecas integradas do IFSC, 2012.

LENARCIC, Jadran; BAJD, Tadej; STANISIC, Michael M. ***Robot Mechanisms***. Springer Ed., 2013.

MERLET, Jean P. ***Advances in Robot Kinematics.*** Kluwer Academic Publishers, Ed., 2010.

NORTON, Robert L. ***Cinemática e Dinâmica dos Mecânismos.*** AMGH Editora Ltda Ed.,2011.

PATSKO, L. ***Tutorial, aplicações, funcionamento e utilização de sensores. Artigo de pesquisa e desenvolvimento***. Disponível em <http://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_aplicacoes_e_funcionamento_de_sensores.pdf > acesso em 21/03/2016.

TARTARI FILHO, Sylvio C. ***Modelagem e otimização de um robô de arquitetura paralela para aplicações industriais***. São Paulo: Universidade de São Paulo, 2006.

TOFÓLI, M. F.; HIGA, R.A.; MANCUSO, E. ***Estudo comparativo entre CLP e Microcontrolador em um elevador de baixa complexidade para carga.*** Garça: Revista e-f@tec, edição 2014.

WILLIAMS II, Robert L. ***Robot Mechanics.*** Ohio University, 2015.

6 APÊNDICES

6.1 APÊNDICE A: Detalhamento das peças do robô

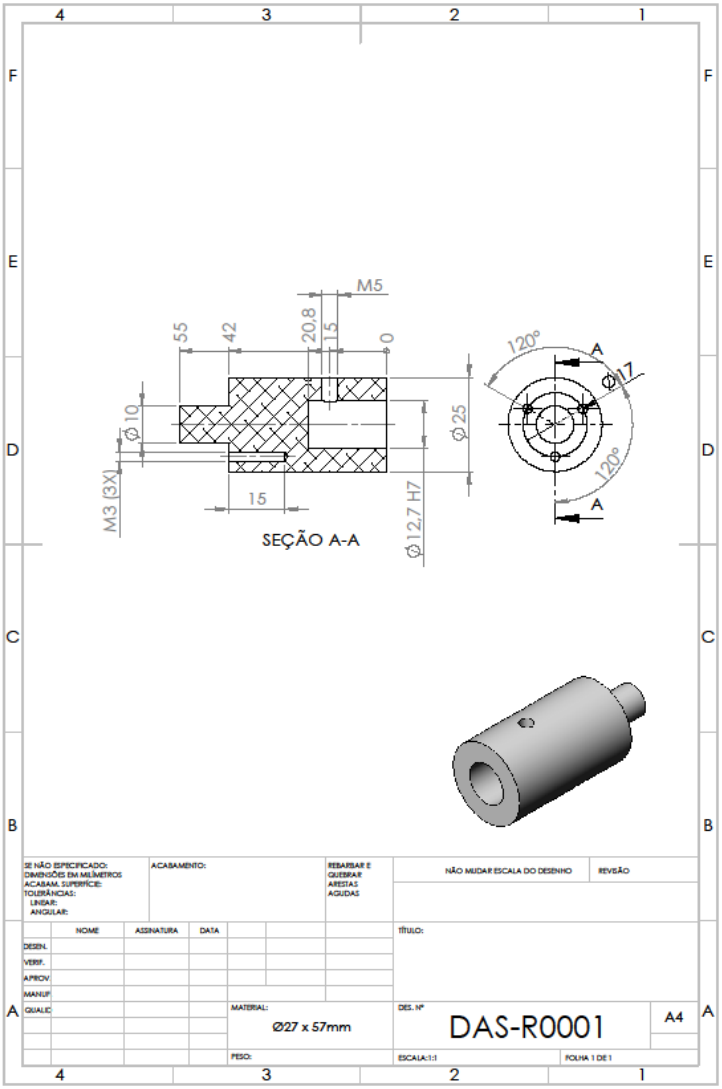


Figura 53 – DAS-R0001, bucha de fixação do motor e braços
Fonte: Autores

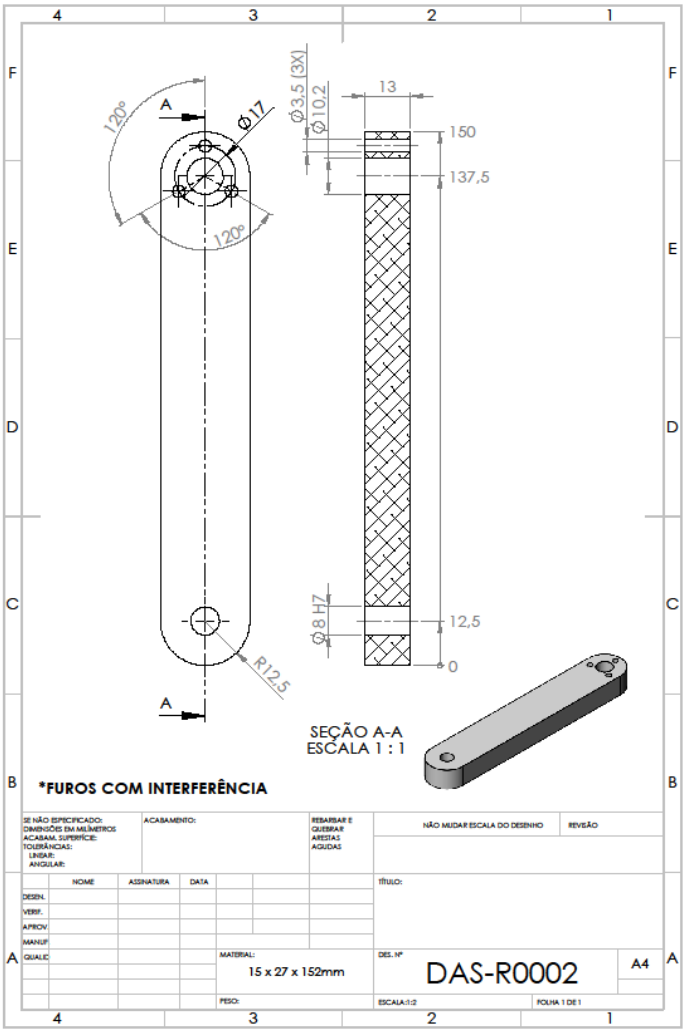


Figura 54 – DAS-R0002, barras $r2$ e $r5$
Fonte: Autores

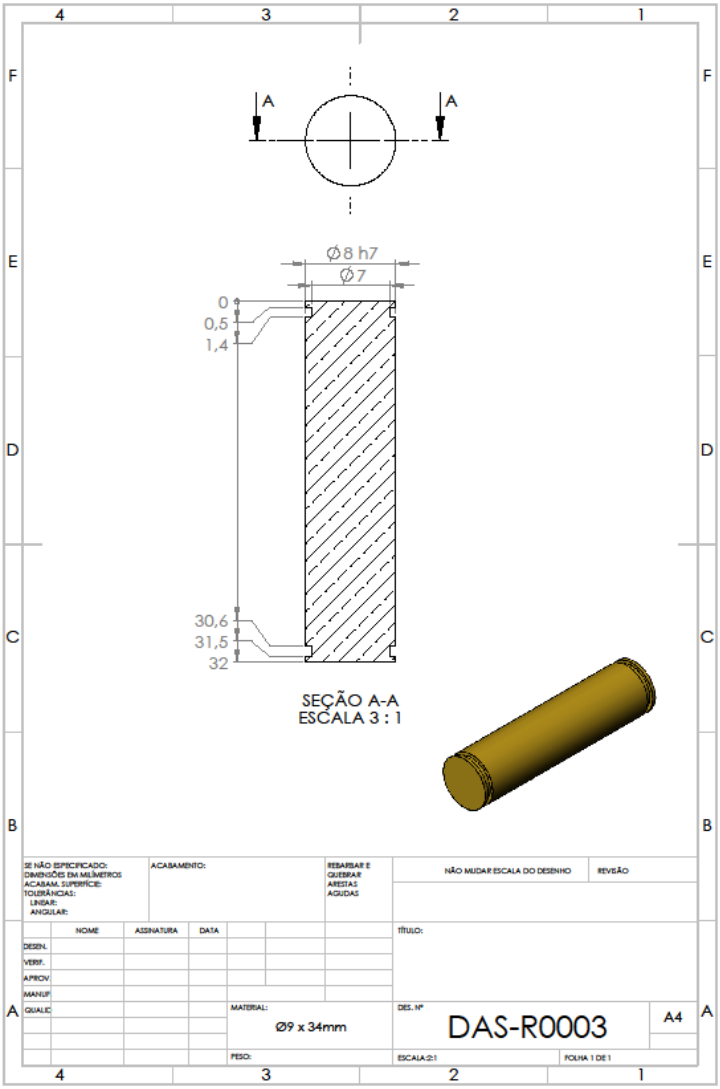


Figura 55 – DAS-R0003, pino das juntas θ_3 e θ_4
Fonte: Autores

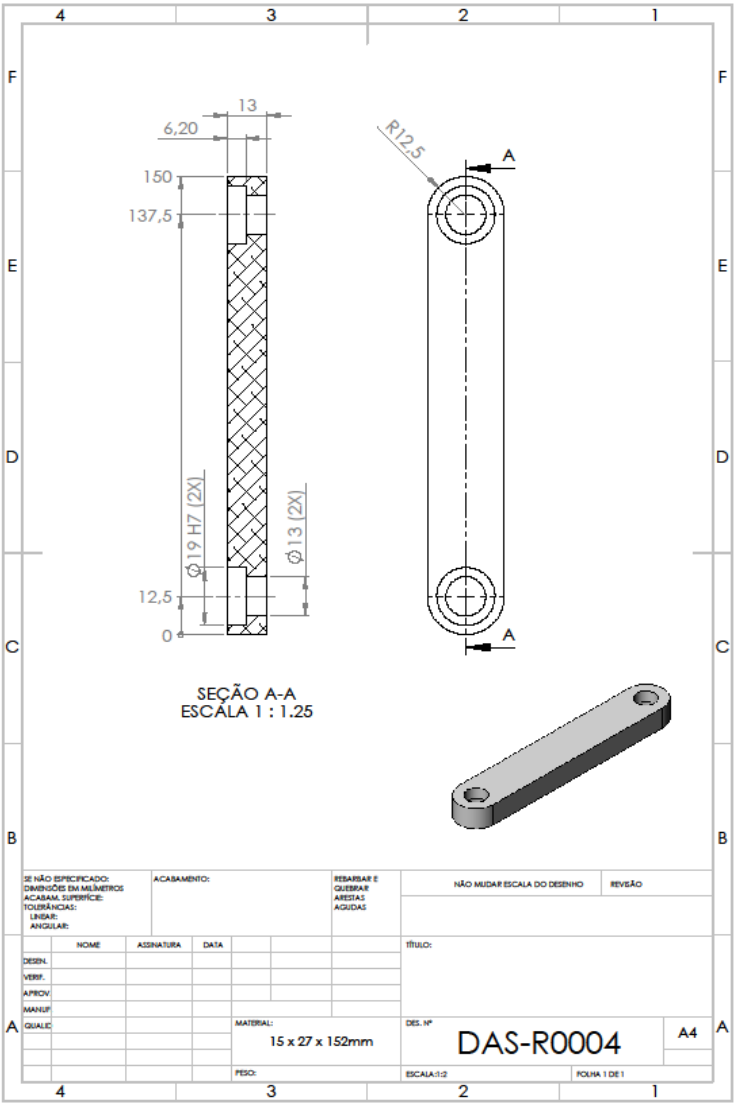


Figura 56 – DAS-R0004, barra $r\,4$
Fonte: Autores

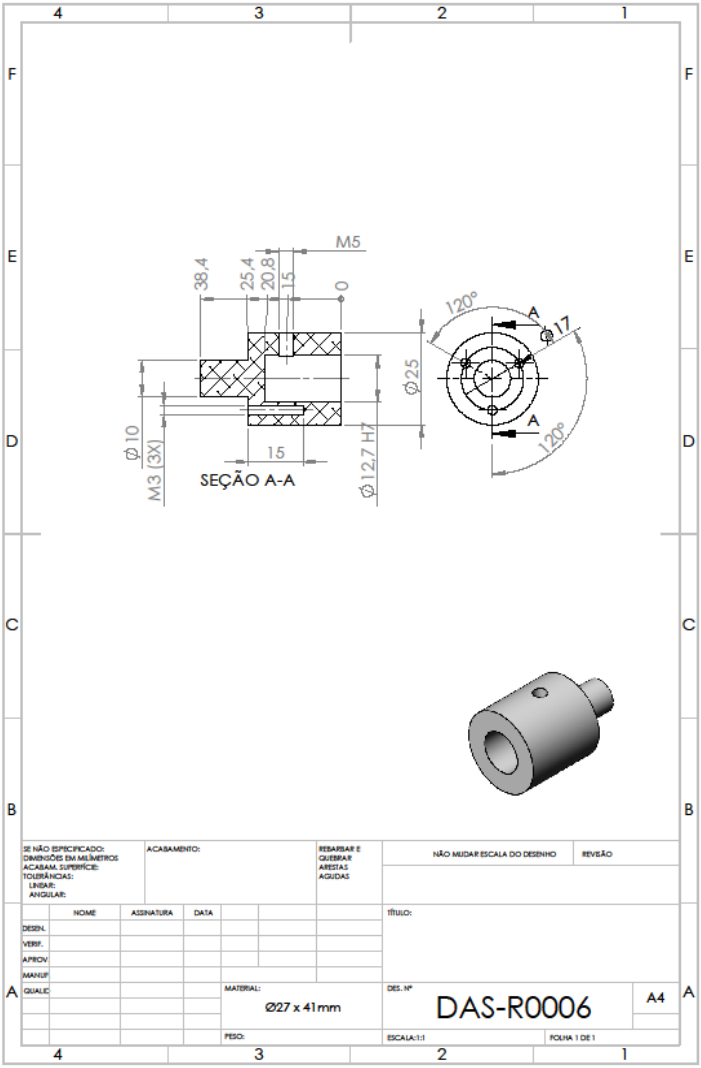


Figura 57 – DAS-R0006, bucha de fixação do motor e braços
Fonte: Autores

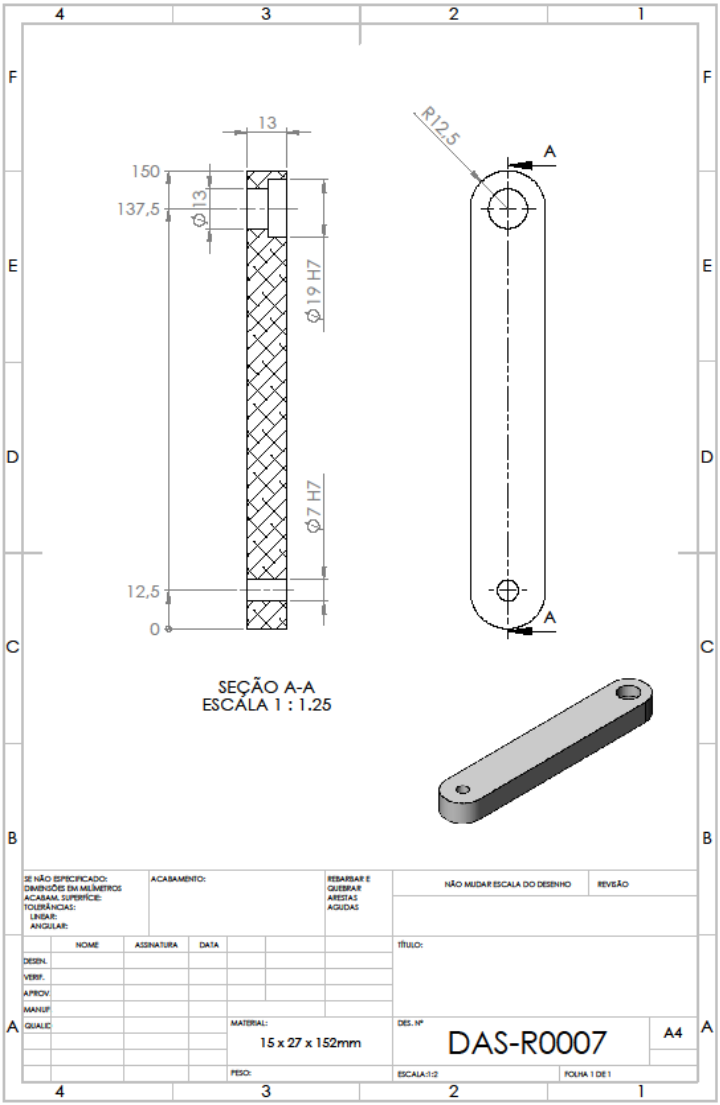


Figura 58 – DAS-R0007, barra $r3$
Fonte: Autores

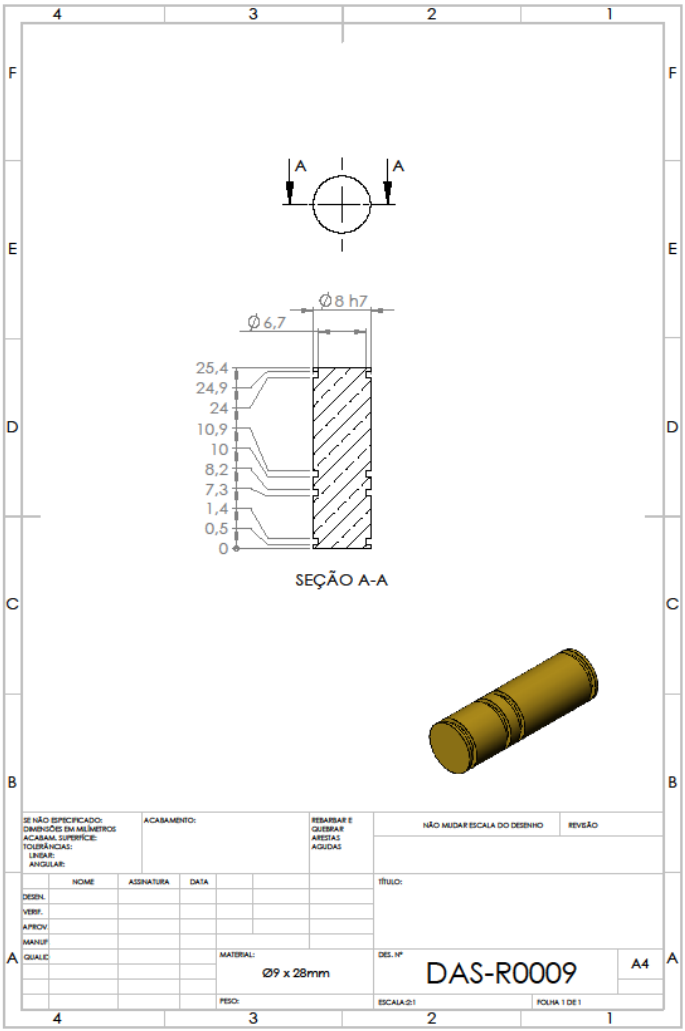


Figura 59 – DAS-R0009, pino de fixação das barras $r2$ e $r3$
Fonte: Autores

6.2 APÊNDICE B: Código utilizado no Arduino

```
// Biblioteca motor de passo
#include <AccelStepper.h>
//============

// Configuração motores
AccelStepper stepper1(1, 4, 3); // Pinos do arduino em que o
driver está conectado
AccelStepper stepper2(1, 7, 6); // Pinos do arduino em que o
driver está conectado
//============

// Configuração serial
const byte numCarac = 32;      //Máximo de caracteres
char caracRec[numCarac];       //Vetor que armazena valores rece-
bidos
char caracTemp[numCarac];      //Vetor temporário para análise

// Variáveis que armazenam os dados analisados
char tipoMens[numCarac] = {0};
char funcMens[numCarac] = {0};
int dadosMens = 0;

boolean novoDado = false;
//============

// Configuração do driver
const int ena1 = 2; // Pino enable do driver 1
const int dir1 = 3; // Pino direção do driver 1
const int pul1 = 4; // Pino pulso do driver 1
const int ena2 = 5; // Pino enable do driver 2
const int dir2 = 6; // Pino direção do driver 2
const int pul2 = 7; // Pino pulso do driver 2
//============

int i = 0;

String serial;
float n1, _n1;
float n2, _n2;

void setup() {
  stepper1.setMaxSpeed(640.0);      //Máxima velocidade em passos
por segundo
  stepper1.setAcceleration(640.0); //Máxima aceleração em passos
por segundo
  stepper2.setMaxSpeed(640.0);      //Máxima velocidade em passos
por segundo
  stepper2.setAcceleration(640.0); //Máxima aceleração em passos
por segundo
```

```
  Serial.begin(9600); //Taxa de transmissão da serial

  pinMode(ena1, OUTPUT);     //Habilita o pino enable como saída
  pinMode(dir1, OUTPUT);     //Habilita o direção enable como sa-
ída
  pinMode(pul1, OUTPUT);     //Habilita o pulso enable como saída

  pinMode(ena2, OUTPUT);     //Habilita o pino enable como saída
  pinMode(dir2, OUTPUT);     //Habilita o direção enable como sa-
ída
  pinMode(pul2, OUTPUT);     //Habilita o pulso enable como saída

  digitalWrite(ena1, HIGH); //Enable em nível alto
  digitalWrite(dir1, LOW);  //Direção em nível alto
  digitalWrite(pul1, LOW);  //Pulso em nível alto

  digitalWrite(ena2, HIGH); //Enable em nível alto
  digitalWrite(dir2, LOW);  //Direção em nível alto
  digitalWrite(pul2, LOW);  //Pulso em nível alto
}

void loop() {
  recComDelim();
  if (novoDado == true) {
    strcpy(caracTemp, caracRec);
    analisaDado();
    comando();
    novoDado = false;
  }
  stepper1.run();
  stepper2.run();
}

//============

void recComDelim() {
  static boolean recEmProg = false;
  static byte ndx = 0;
  char delimIni = '<';
  char delimFin = '>';
  char leSerial;

  while (Serial.available() > 0 && novoDado == false) {
    leSerial = Serial.read();

    if (recEmProg == true) {
      if (leSerial != delimFin) {
        caracRec[ndx] = leSerial;
        ndx++;
```

```
        if (ndx >= numCarac) {
          ndx = numCarac - 1;
        }
      }
      else {
        caracRec[ndx] = '\0'; // Termina a string
        recEmProg = false;
        ndx = 0;
        novoDado = true;
      }
    }
    else if (leSerial == delimIni) {
      recEmProg = true;
    }
  }
}

//============

void analisaDado() {      // Divide os dados em partes
  char * strtokIndx;      // Usado por strtok() como índice
  strtokIndx = strtok(caracTemp, ","); // Continua a partir da
última chamada
  strcpy(tipoMens, strtokIndx);
  strtokIndx = strtok(NULL, ",");
  strcpy(funcMens, strtokIndx);
  strtokIndx = strtok(NULL, ",");
  dadosMens = atoi(strtokIndx);
}

void comando() {
  n1 = stepper1.currentPosition();
  n2 = stepper2.currentPosition();
  if (strcmp(tipoMens, "00") == 0 && strcmp(funcMens, "00") == 0
&& dadosMens == 0000) {
    digitalWrite(ena1, LOW);
    Serial.println("Motor 1 ligado");
  }
  if (strcmp(tipoMens, "00") == 0 && strcmp(funcMens, "00") == 0
&& dadosMens == 0001) {
    digitalWrite(ena1, HIGH);
    Serial.println("Motor 1 desligado");
  }
  if (strcmp(tipoMens, "00") == 0 && strcmp(funcMens, "00") == 0
&& dadosMens == 0002) {
    stepper1.setCurrentPosition(0);
    Serial.println(stepper1.currentPosition());
  }
  if (strcmp(tipoMens, "01") == 0 && strcmp(funcMens, "00") == 0
&& dadosMens == 0000) {
    digitalWrite(ena2, LOW);
    Serial.println("Motor 2 ligado");
  }
```

```
  if (strcmp(tipoMens, "01") == 0 && strcmp(funcMens, "00") == 0
&& dadosMens == 0001) {
    digitalWrite(ena2, HIGH);
    Serial.println("Motor 2 desligado");
  }
  if (strcmp(tipoMens, "01") == 0 && strcmp(funcMens, "00") == 0
&& dadosMens == 0002) {
    stepper2.setCurrentPosition(0);
    Serial.println(stepper2.currentPosition());
  }
  if (strcmp(tipoMens, "00") == 0 && strcmp(funcMens, "01") ==
0) {
    _n1 = n1;
    n1 = dadosMens;
    stepper1.moveTo(_n1 + n1);
  }
  if (strcmp(tipoMens, "01") == 0 && strcmp(funcMens, "01") ==
0) {
    _n2 = n2;
    n2 = dadosMens;
    stepper2.moveTo(_n2 + n2);
  }
  if (strcmp(tipoMens, "00") == 0 && strcmp(funcMens, "09") ==
0) {
    stepper1.moveTo(dadosMens);
  }
  if (strcmp(tipoMens, "01") == 0 && strcmp(funcMens, "09") ==
0) {
    stepper2.moveTo(dadosMens);
  }
  if (strcmp(tipoMens, "00") == 0 && strcmp(funcMens, "10") ==
0) {
    stepper1.setCurrentPosition(dadosMens);
  }
  if (strcmp(tipoMens, "01") == 0 && strcmp(funcMens, "10") ==
0) {
    stepper2.setCurrentPosition(dadosMens);
  }
}
```

6.3 APÊNDICE C: Código utilizado no Visual C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;

namespace arduino
{
    public partial class Form1 : Form
    {
        float ang1 = 90;
        float ang2 = 90;
        string retorno;

        int r1 = 85, r2 = 125, r3 = 125, r4 = 125, r5 = 125,
            zM1, zM2,
            qq2, qq5;

        float angDes1, angDes2;
        double q1 = 0, q2, q3, q3x, q3y, q4, q5, q6, q7, q8,
               q2r, q5r,
               tq2, tq5;

        double a, b, e, f, g, t, x, y,
               e2, f2, g2, t2, Bx, By,
               e5, f5, g5, t5;

        private void btnEnviar_Click(object sender, EventArgs e)
        {
            inversa();
            angDes1 = Convert.ToInt32(q2 * 6400) / 360;
            angDes2 = Convert.ToInt32(q5 * 6400) / 360;
            conexao.Write("<00,09," + angDes1.ToString() + ">");
            conexao.Write("<01,09," + angDes2.ToString() + ">");
            q2r = angDes1 * 360 / 6400;
            q5r = angDes2 * 360 / 6400;
            nudAnguloM1.Value = (decimal)q2r;
            nudAnguloM2.Value = (decimal)q5r;
            rtbRetorno.Text = q2r + "\t" + angDes1 + "\n" + q5r + "\t"
+ angDes2 + "\n" + qq2 + "\n" + qq5;
            escreve();
        }
```

```
        private void numericUpDown1_ValueChanged(object sender, Even-
tArgs e)
        {
            angDes1 = ((float)nudAnguloM1.Value * 6400f) / 360f;
            conexao.Write("<00,09," + angDes1.ToString() + ">");
            ang1 = (float)nudAnguloM1.Value;
            direta();
            escreve();
        }

        private void numericUpDown2_ValueChanged(object sender, Even-
tArgs e)
        {
            angDes2 = ((float)nudAnguloM2.Value * 6400f) / 360f;
            conexao.Write("<01,09," + angDes2.ToString() + ">");
            ang2 = (float)nudAnguloM2.Value;
            direta();
            escreve();
        }

        private void Form1_FormClosed(object sender, FormClosedEven-
tArgs e)
        {
            conexao.Write("<00,00,0001>");
            retorno = conexao.ReadLine();
            rtbRetorno.Text += retorno + "\n";

            conexao.Write("<01,00,0001>");
            retorno = conexao.ReadLine();
            rtbRetorno.Text += retorno + "\n";

            conexao.Close();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            conexao.Open();

            zM1 = Convert.ToInt32(txbPosAtual1.Text) * 6400 / 360;
            rtbRetorno.Text += zM1 + "\n";
            conexao.Write("<00,10," + zM1 + ">");
            nudAnguloM1.Value = zM1 * 360 / 6400;

            zM2 = Convert.ToInt32(txbPosAtual2.Text) * 6400 / 360;
            rtbRetorno.Text += zM2 + "\n";
```

```
            conexao.Write("<01,10," + zM2 + ">");
            nudAnguloM2.Value = zM2 * 360 / 6400;
        }

        private void btnDesligaM1_Click(object sender, EventArgs e)
        {
            conexao.Write("<00,00,0000>");
            retorno = conexao.ReadLine();
            rtbRetorno.Text += retorno + "\n";
            conexao.Write("<00,00,0001>");
            retorno = conexao.ReadLine();
            rtbRetorno.Text += retorno + "\n";
            conexao.Write("<01,00,0000>");
            retorno = conexao.ReadLine();
            rtbRetorno.Text += retorno + "\n";
        }

        private void btnLigaM2_Click(object sender, EventArgs e)
        {
            inversa();
            conexao.Write("<01,00,0000>");
            escreve();
            retorno = conexao.ReadLine();
            rtbRetorno.Text += retorno + "\n";
        }
    }
    SerialPort conexao = new SerialPort("COM3", 9600);

    private void btnDesligaM2_Click(object sender, EventArgs e)
    public Form1()
    {
        conexao.Write("<01,00,0001>");
        InitializeComponent();
        retorno = conexao.ReadLine();
        rtbRetorno.Text += retorno + "\n";
    }
    private void btnZeraM1_Click(object sender, EventArgs e)
    {
    public void direta() {
        zM1 = Convert.ToInt32(txbPosAtual1.Text) * 6400 / 360;
        q2 = ang1;
        q5 = ang2;
        conexao.Write("<00,10," + zM1 + ">");
        a = r1 * Math.Cos(Math.PI * q1 / 180) + r5 * Math.Cos(Ma-
th.PI * q5 / 180);
        nudAnguloM1.Value = zM1 * 360 / 6400;
        b = r1 * Math.Sin(Math.PI * q1 / 180) + r5 * Math.Sin(Ma-
th.PI * q5 / 180);
        escreve();
        e = 2 * r4 * (a - r2 * Math.Cos(Math.PI * q2 / 180));
    private void btnZeraM2_Click(object sender, EventArgs e)
        f = 2 * r4 * (b - r2 * Math.Sin(Math.PI * q2 / 180));
    {
        g = Math.Pow(a, 2) + Math.Pow(b, 2) + Math.Pow(r2, 2) -
Math.Pow(r3, 2) + Math.Pow(r4, 2) - 2 * r2 * (a * Math.Cos(Math.PI *
        zM2 = Convert.ToInt32(txbPosAtual2.Text) * 6400 / 360;
q2 / 180) + b * Math.Sin(Math.PI * q2 / 180));
        rtbRetorno.Text += zM2 + "\n";
        conexao.Write("<01,10," + zM2 + ">");
        t = (-1 * Math.Sqrt(Math.Pow(e, 2) + Math.Pow(f, 2) - Ma-
th.Pow(g, 2)) / (g - e));
        nudAnguloM2.Value = zM2 * 360 / 6400;
        escreve();
        q4 = 2 * Math.Atan(t) * 180 / Math.PI;
        if (q4 > -180 && q4 < 0)
        {
    private void btnLigaM1_Click(object sender, EventArgs e)
            q4 = 360 + q4;
        }
    {
        q3x = r1 * Math.Cos(Math.PI * q1 / 180) + r5 * Math.-
        conexao.Write("<00,00,0000>");
Cos(Math.PI * q5 / 180) + r4 * Math.Cos(Math.PI * q4 / 180) - r2 *
        retorno = conexao.ReadLine();
Math.Cos(Math.PI * q2 / 180);
        rtbRetorno.Text += retorno + "\n";
        q3y = r1 * Math.Sin(Math.PI * q1 / 180) + r5 *
```

```
Math.Sin(Math.PI * q5 / 180) + r4 * Math.Sin(Math.PI * q4 / 180) - r2
* Math.Sin(Math.PI * q2 / 180); ;
            q3 = Math.Atan2(q3y, q3x) * 180 / Math.PI;
            if (q3 > -180 && q3 < 0) {
                q3 = 360 + q3;
            }
            y = r2 * Math.Sin(Math.PI * q2 / 180) + r3 * Math.Sin(Ma-
th.PI * q3 / 180);
            x = r2 * Math.Cos(Math.PI * q2 / 180) + r3 * Math.Cos(Ma-
th.PI * q3 / 180);

            q6 = (180 - 90 - q3) + (180 - 90 - (180 - q4));
            q7 = 180 - (90 + q2 - 90) + q3;
            q8 = 180 - (90 + 90 - q5) + 180 - q4;
        }

        public void inversa() {
            Bx = Convert.ToDouble(txbBx.Text);
            By = Convert.ToDouble(txbBy.Text);
            e2 = -2 * r2 * Bx;
            f2 = -2 * r2 * By;
            g2 = Math.Pow(r2, 2) - Math.Pow(r3, 2) + Math.Pow(Bx, 2) +
Math.Pow(By, 2);
            t2 = (-f2 + Math.Sqrt(Math.Pow(e2, 2) + Math.Pow(f2, 2) -
Math.Pow(g2, 2))) / (g2 - e2);
            e5 = 2 * r5 * (-Bx + r1 * Math.Cos(Math.PI * q1 / 180));
            f5 = 2 * r5 * (-By + r1 * Math.Sin(Math.PI * q1 / 180));
            g5 = Math.Pow(r1, 2) + Math.Pow(r5, 2) - Math.Pow(r4, 2) +
Math.Pow(Bx, 2) + Math.Pow(By, 2) - 2 * r1 * (Bx * Math.Cos(Math.PI *
q1 / 180) + By * Math.Sin(Math.PI * q1 / 180));
            t5 = (-f5 - Math.Sqrt(Math.Pow(e5, 2) + Math.Pow(f5, 2) -
Math.Pow(g5, 2))) / (g5 - e5);

            q2 = 2 * Math.Atan(t2) * 180 / Math.PI;
            tq2 = q2;
            //Se menor que zero muda
            if (q2 < 0) q2 = 360 + q2;
            //Verifica quadrante q2
            if (0 < q2 && q2 < 90)    qq2 = 1;
            if (90 < q2 && q2 < 180)  qq2 = 2;
            if (180 < q2 && q2 < 270) qq2 = 3;
            if (270 < q2 && q2 < 360) qq2 = 4;
            q3 = Math.Atan2(By - r2 * Math.Sin(Math.PI * q2 / 180), Bx
- r2 * Math.Cos(Math.PI * q2 / 180)) * 180 / Math.PI;
            q5 = 2 * Math.Atan(t5) * 180 / Math.PI;
```

```
            tq5 = q5;
            //Se menor que zero muda
            if (q5 < 0) q5 = 360 + q5;
            //Verifica quadrante q5
            if (0 < q5 && q5 < 90) qq5 = 1;
            if (90 < q5 && q5 < 180) qq5 = 2;
            if (180 < q5 && q5 < 270) qq5 = 3;
            if (270 < q5 && q5 < 360) qq5 = 4;
            //Ângulo conforme combinação de quadrantes
            if (qq2 == 1 && qq5 == 3) q5 = tq5;
            if (qq2 == 1 && qq5 == 4) q5 = tq5;
            if (qq2 == 1 && qq5 == 1) q5 = tq5;
            if (qq2 == 1 && qq5 == 2) q5 = tq5;
            if (qq2 == 2 && qq5 == 1) q5 = tq5;
            if (qq2 == 2 && qq5 == 2) q5 = tq5;
            if (qq2 == 3 && qq5 == 2) q5 = tq5;
            //if (qq2 == 3 && qq5 == 3) q5 = q5;
            if (qq2 == 4 && qq5 == 2) q5 = tq5;
            //if (qq2 == 4 && qq5 == 3) q5 = q5;
            //if (qq2 == 4 && qq5 == 4) q5 = q5;
            q4 = Math.Atan2(By - r1 * Math.Sin(Math.PI * q1 / 180) -
r5 * Math.Sin(Math.PI * q5 / 180), Bx - r1 * Math.Cos(Math.PI * q1 /
180) - r5 * Math.Cos(Math.PI * q5 / 180)) * 180 / Math.PI;
        }

        public void escreve() {
            txbX.Text = x.ToString("0.00");
            txbY.Text = y.ToString("0.00");
            txbTheta1.Text = q1.ToString("0.00");
            txbTheta2.Text = q2.ToString("0.00");
            txbTheta3.Text = q3.ToString("0.00");
            txbTheta4.Text = q4.ToString("0.00");
            txbTheta5.Text = q5.ToString("0.00");
            txbTheta6.Text = q6.ToString("0.00");
            txbTheta7.Text = q7.ToString("0.00");
            txbTheta8.Text = q8.ToString("0.00");
        }
    }
}
```