ENGENHARIA DE COMPUTAÇÃO

Luís Felipe dos Santos Oliveira

DESENVOLVIMENTO DE UM ROBÔ CARTESIANO ACIONADO POR CNC

Santa Cruz do Sul

2016

Luís Felipe Dos Santos Oliveira

DESENVOLVIMENTO DE UM ROBÔ CARTESIANO ACIONADO POR CNC

Trabalho de Conclusão II apresentado ao Curso de Engenharia de Computação da Universidade de Santa Cruz do Sul, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Me. Taiser Tadeu Teixeira Barros

Santa Cruz do Sul

2016

Aos meus pais, irmão, e minha esposa Marinês, e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Ao professor Taiser Tadeu, pela paciência durante a orientação e pelo incentivo que tornaram possível a conclusão deste trabalho.

AGRADECIMENTOS

Agradeço aos meus familiares pelo incentivo, em especial à minha esposa pela dedicação junto a mim. Agradeço a todos, por serem compreensivos diante das horas que não pude estar presente em suas vidas. Agradeço à equipe Baja de Galpão por proporcionar um ambiente para pôr em prática os conhecimentos adquiridos. Agradeço também ao professor Mestre Leonel Pablo Tedesco pelo apoio e colaboração durante o desenvolvimento do trabalho de conclusão.

Agradeço a todos direta ou indiretamente que fizeram parte da minha formação, obrigado.

“Se você pensa que pode ou se pensa que não pode, de qualquer forma você está certo. ”
-Henry Ford

RESUMO

O uso de ferramentas computacionais - *Computer Aided Manufacturing* (CAM) - na indústria acelerou o processo de evolução dos recursos utilizados, neste contexto, máquinas controladas por comando numérico - *Computer Numeric Control* (CNC) - realizam trabalhos de forma mais ágil, rápida e precisa. Uma máquina com controle do tipo CNC pode realizar diversos tipos de trabalhos, como cortar, desbastar, furar, aplainar, retificar, entre outros, e são compostas de dois ou mais eixos móveis programáveis controlados tipicamente por um computador. Este trabalho aborda máquinas CNC caracterizadas como um robô cartesiano de 2 e 3 graus de liberdade e controle computacional. A construção deste tipo de máquina envolve áreas distintas, como mecânica, eletrônica e computacional, que juntas podem garantir a precisão e agilidade de tais máquinas. Este trabalho exibe conteúdo bibliográfico sobre os assuntos pertinentes para a construção de um robô cartesiano, bem como: detalhes de projeção, desenvolvimento de *hardware* e *software*, construção física, levantamento de dados e avaliações dos resultados obtidos.

Palavras-chave: Cartesiano, Robô, CNC, Fresadora, Controle.

ABSTRACT

The use of computational tools - Computer Aided Manufacturing (CAM) - the industry accelerated the development of the resources used in this context, machines controlled by CNC - Computer Numeric Control (CNC) - perform jobs more agile, fast and accurate. A machine with CNC control type can perform various types of work, such as cutting, chopping, drilling, planning, grinding, among others, and are composed of two or more programmable axes furniture typically controlled by a computer. This work will address CNC machines characterized as a Cartesian robot 2 and 3 degrees of freedom and computer control. The construction of this type of machinery involves different areas, such as mechanics, electronics and computing, which together can ensure the accuracy and speed of such machines. This academic work presents bibliographic content on the subjects pertinent to the construction of a cartesian robot, as well as: details of projection, development of hardware and software, physical construction, data collection and evaluation of results obtained.

Keywords: Cartesian Robot, CNC, Milling Machine, Control.

LISTA DE IMAGENS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

bps	*Bauds per second*
CAD	*Computer Aided Design*
CAM	*Computer Aided Manufacturing*
CI	*Circuito Integrado*
CNA	*Comando Numérico Adaptativo*
CNC	*Computer Numeric Control*
CTS	*Clear To Send*
DC	*Direct Current*
EIA	*Electronic Industries Association*
EUA	*Estados Unidos da Améric*
IDE	*Integrated Development Environment*
JVM	*Java Virtual Machine*
KHz	*"quilohertz"*
MHz	*"megahertz"*
MCU	*Machine Control Unit*
mm	*milímetro*
MIT	*Massachusetts Institute of Technology*
ms	*milisegundo*
NC	*Numeric Control*
PC	*Personal Computer*
PIC	*Placa de Circuito Impresso*
PNP	*Transistor de lógica negativa*
PPP	*Prismático Prismático Prismático*
RTS	*Request To Send*
Rx	*Reception*
TTL	*Transistor Transistor Logic*
Tx	*Transmission*
UC	*Unit Control*
VM	*Virtual Machine*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

Com a expansão no uso de máquinas na indústria, principalmente após a revolução industrial, o aumento da capacidade de produção continua a expandir junto com a inovação tecnológica. Para Vitor Ferreira Romano (2002) as máquinas são capazes de ajudar a automatizar o processo de manufatura, e desta forma, representam boa parte deste "aumento" de produção da indústria. Máquinas como as fresas são capazes de em alguns casos tornarem autônomo o processo de manufatura de algum produto.

Inicialmente uma alternativa para o controle manual das máquinas em geral era feita por uma sequência de números, os quais representavam a ordem e movimentos que a máquina deveria realizar, sem o advento da informática. Os comandos numéricos NC eram armazenados em uma fita de papel perfurado, a máquina possuía um dispositivo que era capaz de "ler" tal fita e reproduzir os comandos numéricos ali gravados, Aryoldo Machado (1994).

Com a evolução da computação e do surgimento de ferramentas auxiliadas por computador (*Computer Aided Design* - CAD), o uso de computadores para desenvolvimento de produtos e a manufatura dos mesmos foi inevitável. Frente às limitações do papel perfurado para o controle das máquinas, a aplicação de um computador para tal função se mostrou eficiente e até hoje é utilizada, assim, surgiu o comando numérico computadorizado CNC, Adriano F. de Souza e Cristiano B. L. Ulbrich (2009).

Analisando a definição de robô industrial: "manipulador multifuncional reprogramável, projetado para movimentar materiais, partes, ferramentas ou peças especiais, através de diversos movimentos programados, para o desempenho de uma variedade de tarefas" (RIVIN, 1988, p.3), pode-se classificar uma fresadora como um dispositivo similar a este tipo de robô.

Segundo Gonçalves (2006) uma máquina CNC é composta de uma unidade de controle, na qual é armazenado o *software* que processa todos os cálculos necessários do sistema, a máquina propriamente dita, parte física do robô junto com os sistemas responsáveis pelo movimento. Para que uma máquina CNC possa funcionar, é necessário estabelecer uma conexão, diálogo, entre o programador e a máquina, comunicação esta feita por códigos ou símbolos padronizados, normalmente utilizando programação em código G.

Atualmente estuda-se muito acerca de máquinas deste tipo, principalmente por desenvolvedores independentes como: entusiastas, amadores, engenheiros, *hackers* e artistas comprometidos com a criativamente, concepção e construção de objetos e/ou materiais para ambas as extremidades lúdicas e úteis. Segundo Lee Martin (2015), o Movimento *Maker* tem

crescido entre os educadores, ao fato que estes estão trazendo o movimento para a educação, oportunizando aos alunos o envolvimento em práticas de engenharia, design, ciência, tecnologia entre outras. Algumas atividades já faziam parte dentro do currículo nas escolas como: marcenaria, costura, eletrônica. E, estas práticas já desenvolvidas foram revigoradas nos últimos anos com o advento das ferramentas de fabricação digital e a internet, o que tornou muito mais fácil criticar, desenvolver e compartilhar informações e projetos. Martin (2015) apud Montessori (1912) reitera a ideia ao afirmar que as crianças e os jovens podem aprender brincando e construindo com ferramentas e materiais, podendo ser autoras de seu próprio aprendizado, e seguindo nessa linha, nos dias atuais tem-se o uso de mais recursos. Como afirma o autor Martin, "Entre os materiais mais comuns utilizados nestes projetos é possível citar Arduino, Raspberry e outros microcontroladores"(MARTIN, 2015, p.34).

Este trabalho propõe-se a juntar estas duas partes, ser uma forma de ligação entre máquinas industriais e ferramenta didática. Considerando a preferência atual de desenvolvedores por máquinas do tipo, este busca ser um projeto simples, mas que contemple os elementos e características que sirvam de porta de entrada para o estudo de robôs industriais.

1.1 Objetivo geral

O objetivo geral deste trabalho é criar um robô cartesiano simplificado com acionamento por CNC para servir como porta de entrada para estudos sobre robôs industriais, sendo capaz de servir como referência didática para práticas de laboratório com baixa complexidade e contribuindo com material para o Movimento *Maker*.

1.1.1 Objetivos específicos

Os objetivos específicos podem ser divididos em:

- Comando numérico, compreender a concepção e fundamentos considerados no desenvolvimento do sistema de controle robôs por meio numérico;
- Comando numérico computadorizado, entender a diferença entre o comando numérico e o comando numérico computadorizado bem como as alterações necessárias e padrões utilizados em sua aplicação;
- Robótica industrial, esta pesquisa busca por detalhes de alguns robôs industriais como forma de atuação e aspectos físicos, busca também elucidar a importância e aplicabilidade do robô na indústria;

- G-CODE e M-CODE, definir com clareza sua importância para a atuação de robôs industriais e compreender seu funcionamento a fim de poder ser utilizado;
- Unidades de controle UC, buscar por modelos existentes e diagramas e literaturas que auxiliem na projeção e construção de uma UC para este trabalho;
- Sistemas de comunicação, definir por meio de pesquisa qual sistema de comunicação é largamente utilizado e porquê, desta forma, definido qual o sistema utilizado no trabalho;
- *Softwares* CAD e CAM, elucidar a importâncias destes *softwares* para o meio industrial bem como suas aplicações e importância para o projeto;
- Elaborar *software* para comunicação, baseado nas pesquisas realizadas este *software* promove a comunicação PC ⇔ Máquina deste trabalho;
- Construir uma UC, *hardware* e *software*, esta unidade de controle se baseia nos aspectos estudados e utilização das características importantes encontradas na pesquisa;
- Construção física do robô, esta etapa se destina à construção física do robô baseado nas pesquisas realizadas com a utilização de características observadas e definições encontradas nas pesquisas.

1.2 Estrutura

Este trabalho é formulado a partir de pesquisa bibliográfica e exploratória na área de robótica, automação industrial e computacional. A pesquisa bibliográfica consiste principalmente no conhecimento teórico que segundo Gil (2008, p. 50), "[...] é desenvolvida a partir de material já elaborado, constituído principalmente de livros e artigos científicos".

Desta forma, este trabalho está disposto da seguinte maneira, no capítulo 1 é apresentado o referencial teórico. Em seguida, no capítulo 2 são apresentados alguns trabalhos relacionados.

Após, no capítulo 3 as definições do projeto a ser realizado. A seguir, no capítulo 4 está o processo de desenvolvimento dos *softwares* da UC (*firmware*) e de comunicação, após, o capítulo 5 é formado pela construção do robô. O capítulo 6 compreende de coleta e análise de dados, no capítulo 7 são apresentadas as avaliações e por fim, o capitulo 8, o qual expõe as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a base para este trabalho, pois todos os sistemas a serem desenvolvidos dependem de uma boa compreensão destes aspectos fundamentados. O capítulo aborda robôs industriais, comando numérico, conceitos de G-CODE e M-CODE, unidades de controle UC, sistemas de comunicação mais utilizados e ferramentas CAM e CAD.

2.1 Robôs industriais

A expansão do uso da robótica na indústria é impulsionada pela necessidade de atender o mercado obtendo um sistema de produção mais automatizado e dinâmico, como exemplifica Romano (2002), devido a características como flexibilidade e adaptação aos sistemas de manufatura, o robô industrial tornou-se um elemento importante.

Vale (2011, p. 22) apud UNECE (2004) complementa que nas últimas décadas, a expansão do mercado aliada a globalização da economia tem levado a indústria a um novo padrão de concorrência, fazendo com que as formas até então empregadas de gestão e produção não sejam mais suficientes para manter a lucratividade e permanência no mercado. Com o objetivo de aumentar a produtividade e garantir a padronização e qualidade dos produtos, a indústria tem aderido à automatização de sua produção, basicamente utilizando robôs manipuladores para realizar tarefas repetitivas. Neste cenário, robôs manipuladores têm sido crescentemente mais utilizados em atividades que envolvem precisão e velocidade.

Vale (2011) detalha que robôs manipuladores sobretudo são máquinas de posicionamento e que podem possuir inúmeros graus de liberdade, do qual a extremidade está fixada a ferramenta ou efetuador, mecanismo cuja finalidade é realizar o trabalho. Desta forma, elos ou juntas que compõem o robô determinam o grau de liberdade do conjunto, podendo elas serem de movimentos de translação (prismático) ou rotação (rotacional), sendo assim, um robô pode ser classificado pelo sentido dos movimentos e por sua alimentação. A autora ainda complementa que:

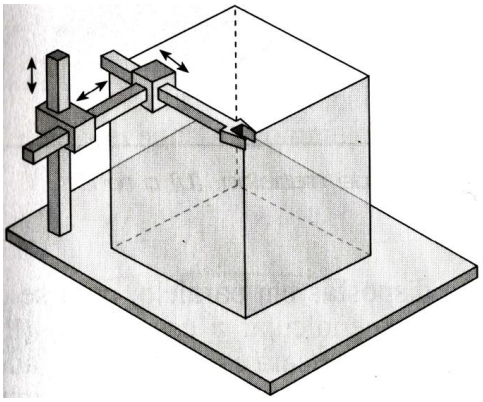
> Quanto à configuração física, um robô pode ser de coordenadas cartesianas, coordenadas cilíndricas, coordenadas esféricas, SCARA (*Seletive Compliance Assembly Robot Arm*), entre outros. Quanto à fonte de alimentação requerida, podem ser hidráulicos, pneumáticos, hidro-pneumáticos, elétricos, eletro-hidráulicos, eletro-pneumáticos, eletromagnéticos, entre outros (VALE, 2011, p. 22).

Romano (2002) esclarece que efetuador é um componente responsável por promover uma interação entre extremidade do manipulador e o objeto a ser manipulado, que podem ser divididos em dois grandes grupos: ferramentas especiais e garras mecânicas. Ferramentas

especiais tem como função realizar trabalhos ou ações sobre uma peça. As garras mecânicas são destinadas à preensão (agarramento) dos objetos para movimentação ou manipulação.

Na indústria, um robô muito popular definido como PPP composto por todas as suas juntas prismáticas, é a fresadora, "[...] máquina cuja ferramenta está animada de movimento de rotação e arranca o material em excesso em forma de cavacos [...]" (FREIRE, 1983, p.1), capaz de produzir peças a partir da matéria prima ao manipular ferramentas predefinidas, uma característica importante é a capacidade de multi ferramentas, não ficando limitada a somente cortar ou furar, Freire (1983).

Imagem 1- Robô Cartesiano PPP



Fonte: ROMANO. *Robótica Industrial*: Aplicação na indústria de manufatura e de processos. (2002, p. 7).

Craig (1989) ainda afirma que manipuladores cartesianos são possivelmente a configuração mais simples de robôs, compostos de 1 a 3 juntas prismáticas, mutuamente ortogonais que correspondem aos vetores X, Y e Z do plano cartesiano, o que garante que sua cinemática inversa seja trivial. Robôs deste tipo podem possuir uma estrutura muito forte, o que os permite serem muito grandes e ainda manipular desde ferramentas pequenas para manufatura até carros ou aeronaves inteiras. Haja vista que, outra vantagem é o fato de suas juntas serem independentes, o que os torna mais simples de projetar e não produz quaisquer singularidades em sua cinemática.

2.2 Comando Numérico

A definição do comando numérico deixa claro sua importância e funcionalidade ao comando de máquinas operatrizes. Segundo Machado,

> O comando numérico NC é um equipamento eletrônico capaz de receber informações por meio de entrada própria, compilar estas informações e transmiti-las em forma de comando à máquina operatriz, de modo que esta, sem a intervenção do operador, realize as operações na sequência programada (1990, p.21).

O comando numérico já vinha sendo pesquisado desde 1945, como esclarece, MACHADO (1990), porém o primeiro esforço em aplicar o NC em máquinas operatrizes começou em 1949, no Laboratório de Servo Mecanismo do Instituto de Tecnologia de Massachusetts (M.I.T) associado a U.S. *Air Force e Parsons Corporation of Traverse City*, de Michigan. Foi escolhida uma fresa de 3 eixos (X, Y e Z) para viabilizar as experiências, a máquina foi modificada a fim de receber todos os equipamentos necessários para viabilizar o comando numérico, o resultado foi uma excelente demonstração de praticidade em março de 1952.

No final da década de 50, aprimoramentos foram realizados gerando mais versatilidade às máquinas com NC, o que tornou a adaptação de máquinas convencionais em NC mais trabalhosas, tal fato fez com que construções de máquinas já desenvolvidas para operar com NC ganhasse espaço. Em 1962 todos os maiores fabricantes de máquinas ferramentais estavam empenhados no comando numérico.

O armazenamento das instruções em comando numérico na época em que foi desenvolvido era feito por meio de cartões de papel perfurados, o qual a UC da máquina era capaz de ler e interpretar os comandos ali contidos. Com o aperfeiçoamento das máquinas, o detalhamento de seus movimentos passou a requerer mais informações, o que forçou a evolução do papel perfurado para uma fita de papel perfurado, depois para fitas magnéticas, disquetes e para volumes maiores de informações um computador próprio para tal finalidade começou a ser usado e, em 1970 surgiu o comando numérico com computador ou comando numérico computadorizado CNC, Aryoldo Machado (1994).

Ainda que outras variações tenham nascido na mesma época, como o DNC comando numérico distribuído, o CNA comando numérico adaptativo entre outros, a absorção do CNC foi tão forte que encontrar outros sistemas em operação é muito raro.

Para cada método de armazenamento que foi usado ao longo da evolução do NC uma linguagem para escrita dos códigos era desenvolvida, com a popularização do CNC após 1970, se fez necessária uma padronização do formato que as instruções deviam ter para serem enviados para a máquina, para isto foi desenvolvido então entre outros o G-CODE e o M-CODE, que são utilizados largamente hoje em dia como forma de comunicação padrão para máquinas CNC.

Máquinas controladas numericamente por computador podem ser encontradas em quase todos os lugares, desde pequenas lojas em comunidades rurais e fortemente na indústria. Hoje, não há praticamente uma etapa de fabricação que não é de alguma forma conduzida por máquinas deste tipo, como afirma, Lynch (1998).

2.3 G-CODE e M-CODE

Estes termos G e M *codes* definem de forma genérica a programação que será enviada para a máquina, como afirmam Kramer, Proctor e Messina (2000). A linguagem NGC RS274 é baseada em linhas de códigos, estas linhas chamadas de "sentença" que, por sua vez, incluem "palavras" que são compostas por uma letra seguida de um número, uma palavra pode ser um comando, ou fornecer argumentos para um comando. A maioria das "palavras" da linguagem NGC começam com G ou M, "palavras" que representam estes comandos são conhecidos como códigos G e códigos M.

Uma sentença pode conter uma ou mais "palavras", sendo ou não iniciada pelo caractere N, o sinal que determina o fim de uma sentença pode ser LF, EOB ou /n/r (quebra de linha). Uma palavra é formada por uma letra que representa um endereço e um número que indica a grandeza deste endereço, sendo um componente individual da sentença e que pode ou não conter mais palavras, como afirma Cassaniga (2005).

É importante também lembrar o que diz Cassaniga (2005, p.148): "No momento de salvar o arquivo este seja feito com formato texto". Entende-se então que a programação em código G é a representação do comando numérico em uma linguagem definida, NGC RS274, com comandos e instruções escritas em formato G-code e M-code em um arquivo de texto.

Comandos de trajetória servem para definir os modos de posicionamento, interpretação de valores e as correções que devem ser feitas nas ferramentas, representados pela letra G seguida de 2 dígitos. A letra M representa as condições adicionais, também seguida de 2 dígitos representados de forma codificada para a máquina, Freire (1983).

As palavras mais utilizadas dos grupos M e G, junto com algumas palavras não pertencentes aos grupos G e M estão disponíveis nas tabelas a seguir.

Tabela 1-G-*CODES* mais utilizados.

Palavra	Descrição
G00	Posicionamento rápido
G01	Interpolação linear
G02	Interpolação circular/helicoidal horário
G03	Interpolação circular/helicoidal anti-horário
G04	Permanência
G10	Coordenada de configuração de origem
G17	Seleção do plano XY
G18	Seleção do plano XZ
G19	Seleção do plano YZ
G20	Sistema em polegadas
G21	Sistema em milímetros
G28	Retorno ao *home*
G30	Retorno ao *home* secundário
G53	Movimentação por sistema de coordenadas
G90	Distância absoluta
G91	Distância incremental

Fonte: Adaptado de Machado (1994 p. 390).

Tabela 2-M-CODES mais utilizados.

Palavra	Descrição
M00	Parada do programa
M01	Parada opcional do programa
M02	Fim do programa
M03	Rotação da ferramenta no sentido horário
M04	Rotação da ferramenta no sentido anti-horário
M05	Parar ferramenta
M06	Carregar ferramenta
M07	Refrigeração por névoa
M08	Refrigeração por inundação
M09	Parar Refrigeração
M30	Fim do programa e redefinição
M48	Permite mudança da velocidade de avanço
M49	Proíbe mudança da velocidade de avanço
M60	Transporte e parada do programa

Fonte: Adaptado de Machado (1994 p. 349).

Tabela 3-Palavras não pertencentes aos grupos G e M.

Palavra	Descrição
A	Eixo A da máquina
B	Eixo B da máquina
C	Eixo C da máquina
D	Compensação de raio
F	Tacha de avanço
G	Funções G (código G)
H	Índice de comprimento da ferramenta
I	Eixo de deslocamento para arco seno X
J	Eixo de deslocamento para arco seno Y
K	Eixo de deslocamento para arco seno Z
L	Número de repetições por ciclo
M	Função auxiliar
N	Número da linha
P	Habilita tempos fixos de permanência
Q	Incremento no alimentador (*feeder*)
R	Raio do arco em ciclo plano (radiano)
S	Velocidade do eixo
T	Seleção de ferramenta
X	Eixo X da máquina
Y	Eixo Y da máquina
Z	Eixo Z da máquina

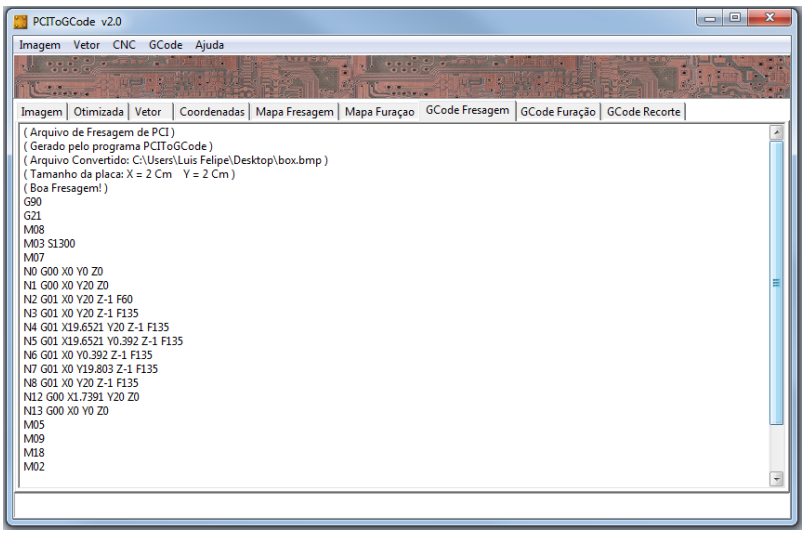
Fonte: Adaptado de Machado (1994 p. 391).

Então, entende-se que um código escrito em linguagem NGC RS274 é comumente chamado de código G por possuir em sua maioria "palavras" do grupo G.

A imagem 2 representa um quadrado com 2 cm de lado em código G, o código foi gerado pelo o *software open source* PCIToGCode [1] desenvolvido por Rubens Bernardi (Eng. Automação Industrial), que permite o salvamento do código gerado em formato texto com extensão .nc, *software* este proposto para utilização no presente trabalho como ferramenta CAM.

[1] PCITOGCODE: conversão de imagem em código G, voltado para placas de circuito impresso. São Paulo: Rubens Bernardi, 2013. Disponível em: <https://sourceforge.net/projects/pcitogcode/>. Acesso em: 12 dez. 2016.

Imagem 2- Representação de um quadrado em código G.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

É possível observar e compreender cada "sentença" do código gerado pelo *software* seguindo as definições citadas com auxílio das tabelas 1, 2 e 3, compreendendo a função de cada "palavra" contida no código.

2.4 Unidade de controle UC ou MCU

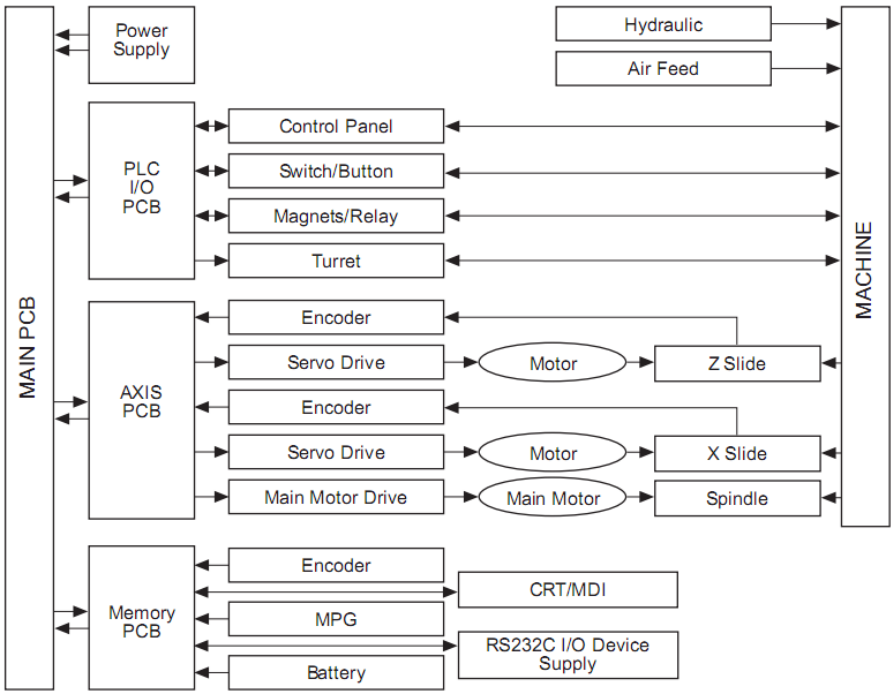
Uma UC pode ser descrita como a definição de Krar, Gill e Smid (2001), A MCU é uma parte intermediária de todo o processo de NC, sua função é armazenar um programa capaz de traduzir as informações recebidas em uma linguagem que possa ser entendida e executada pela máquina para produzir a peça desejada, utilizando as funções disponíveis, funções estas que podem, por exemplo, ser o controle de relés, solenoides, controle dos movimentos por servos mecânicos, elétricos ou hidráulicos.

No início dos anos 1950, as primeiras MCUs começaram a ser desenvolvidas na época a válvulas, hoje com tecnologias mais recentes são empregadas como microprocessador e microcomputadores. Até o início de 1970 todas as funções de uma MCU como: reconhecimento de código, posição absoluta ou incremental e leitura dos dados, eram definidos de forma eletrônica, no *hardware*, este tipo de MCU foi chamado de "*hard-wire*", porque nenhuma função da gama de funções que existiam poderia ser alterada depois da MCU construída. Em meados de 1970 começou a se desenvolver o conceito de "*soft-wire*", mais flexível e mais barata que a proposta anterior, para este, microcomputadores se tornaram parte da MCU, as funções que até então eram atreladas ao *hardware* passaram a ser responsabilidade de um *software*, com

esta nova abordagem o custo foi reduzido e ao mesmo tempo uma mesma MCU pode ser programada para uma variedade de funções diferentes sempre que necessário, Krar, Gill e Smid (2001).

Em seu trabalho sobre falhas em sistemas CNC, Wang, Junyi Yu e Shangfeng Yi (1999) expressão em um diagrama de blocos uma generalização de como uma UC é composta, este diagrama de blocos pode ser visto na imagem 3. Ainda que as UCs utilizadas não são exatamente como o diagrama apresenta, esta generalização é muito representativa para entender os processos internos e subsistemas que compõem uma UC.

Imagem 3-Diagrama de blocos para sistemas CNC



Fonte: De Field failure data base of CNC lathes

Existem *hardwares* e *softwares* já desenvolvidos e prontos que se propõem a ser a UC de um robô cartesiano e que podem ser encontrados facilmente, dentre a variedade disponível é possível citar como exemplo a placa *shield* RAMPS atualmente a mais popular, placa esta com a finalidade de conectar a plataforma Arduino aos drives que acionam os motores, opera sobre o *firmware* Marlin, também existe a placa *shield* GRBL, com propósito semelhante à citada anteriormente e trabalha com *firmware* de mesmo nome, ambas UC mencionadas são de código aberto em *hardware* e *software*.

2.5 Comunicação

Este segmento aborda o sistema de comunicação PC <-> máquina, utilizado para a comunicação do CN mais utilizado, assim como expõe algumas dificuldades e vantagens encontradas neste sistema.

Uma forma para estabelecer uma comunicação entre dois dispositivos computacionais é utilizar a interface desenvolvida pela EIA (*Eletctronic Industries Association*) dos EUA nomeada de RS-232, esta interface permite a comunicação entre dois dispositivos como um computador e uma máquina CNC, Souza e Ulbrich (2009).

Romano (2002) afirma que a interface serial RS-232 é uma das interfaces mais utilizadas, especialmente em conexões ponto a ponto, como em ligações entre robôs industriais e computadores pessoais que rodam *softwares* de configuração e programação. Ainda, cita algumas outras interfaces de comunicação, como a RS-485 que permite maiores distâncias de comunicação com taxas maiores de dados e a interface GPIB, padronizada pela norma IEEE-488 que foi desenvolvida inicialmente para conexões de instrumentos laboratoriais, e por fim as interfaces paralelas que como a maioria dos equipamentos trabalha internamente com dados de forma paralela, evitando a necessidade de conversões, entre outras interfaces e protocolos.

Quando uma UC baseada em Arduino é utilizada, a comunicação ainda que seja conectada por um cabo USB é feita no padrão serial RS-232, característica essa nativa da plataforma Arduino.

2.6 CAM e CAD

Atualmente as ferramentas CAD e CAM são fundamentais em projetos de engenharia, já que a agilidade que as mesmas proporcionam ao desenvolvimento do projeto é fundamental para manter a velocidade de desenvolvimento que o mercado exige.

Por volta de 1950 as primeiras aplicações de computadores para auxiliar as etapas de engenharia foram empregadas, época em que MIT dos EUA havia iniciado discussões sobre tecnologias CAD. Na época os sistemas se limitavam a manipulação de desenhos e representação de entidades em duas dimensões em terminais gráficos monocromáticos, segundo Souza e Ulbrich (2009).

Nos dias atuais, um programa CAD é focado em desenhos de produtos e documentações das fases do projeto durante o processo de desenvolvimento. Os recursos CAD são largamente utilizados por diversos segmentos da engenharia como: Engenharia Elétrica, Mecânica, Civil,

Estruturais, Computação entre outras, evidentemente existem ferramentas CAD voltados para segmentos como os mencionados, o que faz com que a maioria dos *softwares* CAD não sejam concorrentes diretos, visto que podem focar em segmentos diferentes de mercado, Leão (2015).

Já as ferramentas CAM possuem finalidade diferente, na maioria dos casos, as ferramentas CAM são parte das ferramentas CAD e não mais um programa separado.

O CAM, manufatura assistida por computador, é o uso de um programa de computador para controlar a ferramenta utilizada pela máquina que irá realizar o processo de fabricação, sendo assim, não é considerado um sistema de programa de engenharia, mas sim voltado às máquinas na fabricação. Ainda, pode-se definir como CAM a utilização de um computador para ajudar na elaboração de uma planta de fabricação, seu objetivo principal é de criar um processo mais rápido e eficiente com dimensões mais precisas e consistência material, como afirma Leão (2015).

Souza e Ulbrich (2009), dizem que antes dos sistemas CAD/CAM a fabricação de produtos com formas geométricas complexas era normalmente feita de forma artesanal, um modelo de resina deveria ser feito seguindo a forma geométrica a ser usinada e com uma fresa-copiadora a superfície era copiada em outro material, processo este artesanal. No final da década de 1970 foi iniciada a implantação industrial dos sistemas CAM, o que permitiu a programação de máquinas CNC via *software* e assim integrando CAD-CAM ainda que com grandes limitações.

No começo dos anos 1980, quando a tecnologia se tornou mais popular, como exemplifica Souza e Ulbrich (2009), os sistemas CAD e CAM eram integrados e vendidos em um único *software*, o que forçava o comprador a adquirir o *software* completo, já no final dos anos 1990 algumas empresas começaram a desenvolver os *softwares* CAM e CAD em pacotes distintos, o que possibilitava a aquisição dos sistemas individualmente. Atualmente considera-se um erro mencionar CAD/CAM como uma única coisa, pois empresas distintas podem fabricar módulos diferentes.

Neste cenário, os robôs controlados por CNC são a ponta final do processo que se inicia na elaboração do produto em alguma ferramenta CAD, após seu término de desenvolvimento com o auxílio de uma ferramenta CAM os movimentos do robô são definidos a fim de produzir o produto projetado de maneira eficiente, movimentos estes programados em código G que na etapa final será lido pelo conjunto CNC, que lê e executa o programa em código G sobre o robô controlado que, por sua vez, efetivamente faz a manufatura do produto.

2.7 Movimento *Maker*

A crescente excitação em torno do Movimento Maker é compreensível,

> uma das características mais aparentes do Movimento [...] é a celebração e o uso de ferramentas digitais novas e recentemente acessíveis. Como estas ferramentas fornecem novas formas de interagir com materiais físicos, eles também oferecem novas oportunidades de aprendizagem (MARTIN, 2015, p.32).

Walkowiak (2005) diz que tradicionalmente o ensino utiliza diferentes formas como: palestras, aulas práticas, laboratório e projetos, no entanto dentro das universidades as técnicas de laboratórios são a alma do ensino, principalmente nas áreas de ciências e engenharia. Durante aulas de laboratórios os estudantes testam seus conhecimentos teóricos obtidos em palestras.

Lee Martin (2015) afirma que os alunos quando desenvolvem seus projetos sentem-se orgulhosos de desenvolver ferramentas controladas por computador, entre as ferramentas mais populares desenvolvidas pelos estudantes e entusiastas estão às impressoras 3D, CNCs Routers (fresadoras acionadas por CNC), cortadores lasers entre outras máquinas controladas por computador, que são capazes de utilizar um arquivo digital gerado por uma ferramenta CAD para criar objetos e materiais de forma muito mais fácil e ágil do que de forma artesanal. Embora essas ferramentas já existam há algum tempo, os custos atualmente caíram o suficiente para torná-los acessíveis a amadores, fábricas de pequeno porte, escolas e universidades.

Nos dias atuais estuda-se muito acerca de máquinas deste tipo, principalmente por desenvolvedores independentes, como reafirma Sueiro (2015) que no evento SXSW[2] realizado em Austin no Texas, o trabalho apresentado sobre o Estado do Movimento *Maker* no Brasil cita alguns trabalhos realizados que chamaram a atenção dentro do Movimento *Maker* no BRasil, entre estes trabalhos estão uma "CNC", fresadora acionada por CNC, e uma impressora 3D. Sueiro (2015) complementa que "As pessoas, conforme conhecem as ferramentas *open source*, criam seus projetos e os compartilham, aumentando essa onda que promete ser a inspiração de novos negócios e inovação em geral" (2015, p.1), ou seja, existir materiais disponíveis para usar de referência, bem como de inspiração é de grande valia.

Blikstein (2013) justifica a ideia do quanto aulas práticas em laboratório tem se mostrado eficientes. Ele expõe que um dos primeiros e mais marcantes resultados do *workshop* digital que propunha para suas aulas foi o relato dos alunos sobre seu entusiasmo em seguir o plano da aula, que em laboratório, os alunos deveriam primeiramente projetar suas criações no computador com medições e cálculos, para então construir fisicamente o projeto, ou seja, não

[2] LEMOS, Manoel. The State of the Maker Movement in Brazil. In: SXSW, 2015, Austin, 2015. Disponível em: < http://schedule.sxsw.com/2015/events/event_IAP35251>. Acesso em: 12 dez. 2016.

só a teoria, mas tudo fundamentado em duas práticas, sendo elas computação e matemática, o que não gerou somente projetos mais refinados e sofisticados, mas também a capacitação e o aumento da autoestima. Contudo, isto provou um forte princípio freiriano para a construção de experiências através do "*digital fabrications lab*", *fablab*, que funde computação e práticas de engenharia, e tem o potencial para aumentar o conhecimento prático que os alunos já possuem, de modo que eles possam reconhecer suas próprias experiências anteriores e pôr em ação no laboratório.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta alguns trabalhos encontrados durante a pesquisa que abordam uma proposta semelhante a este trabalho, desenvolvimento de um robô e analise do mesmo.

João de Barro (2015) é o nome do projeto de uma fresadora "robô cartesiano" desenvolvida por alunos da Universidade Federal do Rio Grande do Sul - UFRGS e financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPS), com a finalidade de fabricar placas de circuito impresso (PCI) o robô foi desenvolvido pelos alunos utilizando materiais de fácil obtenção. Uma micro retífica foi utilizada como ferramenta para o robô, desta forma, é possível cortar e perfurar materiais. O acionamento do robô é realizado por uma UC composta por um Arduino Uno e uma placa RAMPS 1.4, placa esta pertencente a um projeto *open source* de desenvolvimento de controladores CNC, portanto, a placa de controle da UC e o *software* empregado, o *firmware* Marlin, não foram desenvolvidos pela equipe de, somente as partes mecânicas foram desenvolvidas. Ainda que o projeto João de barro se diferencie do trabalho proposto, as experiências mecânicas compartilhadas pelos desenvolvedores são de grande valia para a projeção e construção do robô cartesiano proposto.

Julio Lazzarim (2012) o objetivo de seu projeto apresentado em seu trabalho de conclusão é auxiliar práticas laboratoriais na área de robótica com a construção de um manipulador de fácil manutenção e baixo custo para auxílio didático. O autor descreve detalhadamente as fases de desenvolvimento e tomadas de decisões ao longo do projeto, escolhas como o uso de motores de passo e não servomotores, o *hardware* da UC ser comporto por um microcontrolador da família PIC e não Arduino são bem claros e pertinentes. A proposta de ser uma ferramenta didática é uma característica em comum com a proposta deste trabalho, desta forma, alguns aspectos criados e desenvolvidos pelo autor podem servir de inspiração para o desenvolvimento deste.

Carlos Perché (2013) desenvolve em seu projeto uma fresadora "robô cartesiano" automática para confecção de placas de circuito impresso, o *software* de comunicação e a UC também são desenvolvidos pelo autor. A UC desenvolvido conta com um microcontrolador da família PIC, o *software* desenvolvido possui a capacidade interpretar programações em código G e promover a comunicação com o robô, o autor ainda detalha o processo de construção e materiais envolvidos. Ainda que a proposto do trabalho citado seja a construção de um robô funcional com uma aplicação especifica, seu trabalho é de extrema relevância para o projeto

deste, que visa construir um robô semelhante, porem com o menor nível de complexidade possível para que possa ser utilizado como ferramenta didática.

Artigo escrito por Laureto. et al. (2016) e publicado em julho de 2016 apresenta um projeto *open source* de um sistema de solda de multicamadas de polímeros por laser. O projeto se caracteriza por detalhar os componentes utilizados e técnicas empregadas para que o objetivo seja alcançado, o soldador desenvolvido é constituído de um robô cartesiano planar "eixos X e Y" com o efetuador "laser" atuando de forma binária. A UC utilizada no robô não é detalhada no projeto, porém um ponto importante a ser explorado são os dados capturados e exposto no trabalho, dados como velocidades de deslocamento dos eixos, altura do laser em relação ao material e demais características. Ainda que o trabalho citado tenha uma proposta diferente ele se baseia em um robô cartesiano com acionamento binário para o eixo Z e explora a coleta de dados, o que para este trabalho proposto é de grande valia, visto que quando utilizado como ferramenta didática, o cruzamento de dados é uma forma de avaliação muito relevante.

Ao analisar os trabalhos relacionados foi possível fazer algumas comparações com o trabalho desenvolvido e posteriormente algumas considerações. A Tabela 4 apresenta as principais diferenças e semelhanças encontradas nos trabalhos mencionados.

Tabela 4-Comparativo entre trabalhos relacionados e o trabalho proposto.

Trabalho	Robô cartesiano	UC desenvolvida	Eixo Z binário	Softwares desenvolvidos	Analise de resultados
João de barro	**x**				
Julio Lazzarim		**x**		**x**	
Carlos Perché	**x**	**x**		**x**	
Laureto et al.	**x**		**x**		**x**
Este Trabalho	**x**	**x**	**x**	**x**	**x**

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

i) Os trabalhos apresentados apresentam o desenvolvimento de algum tipo de robô, porém sem grande detalhamento de todas as etapas do desenvolvimento.

ii) Os trabalhos analisados não exploram resultados pertinentes ao robô desenvolvido ou utilizado, este trabalho apresenta dados reais do robô criado.

iii) O desenvolvimento completo de um robô cartesiano juntamente com sua UC, *software* de comunicação e analise de resultados é apresentado apenas neste trabalho.

4 DESENVOLVIMENTO

Com base na pesquisa realizada anteriormente, todas as etapas de formulação das propostas para o robô e seus sistemas de controle estão descritos neste capítulo, todas as escolhas tomadas são baseadas na pesquisa já realizada ou em pesquisas pontuais para algum aspecto em específico.

4.1 Definições

Foi adotado como ponto de partida para o desenvolvimento do robô proposto os requisitos que ele deve cumprir, os aspectos mecânicos, eletrônicos e recursos computacionais.

O robô deve ser capaz de manipular algum objeto para comunicação com o mundo, para fins didáticos, quando utilizado como exemplo de ferramenta de ensino e para facilitar a visualização dos processos envolvidos, ao qual foi adotada a capacidade de manipular uma caneta esferográfica comum, desta forma, o robô será capaz de desenhar objetos programados, para tal tarefa será necessário dois eixos deslizantes a fim de cobrir o plano X e Y com área mínima semelhante ao padrão de folhas A4 630 cm^2, além disso, um efetuador deve ter a capacidade de acionar o uso da caneta quando requisitado, novamente para fins didáticos, o efetuador será controlado de forma binária, com apenas 2 níveis lógicos, considerando que o efetuador controlará a ferramenta (caneta) somente em movimentos no eixo Z, será atribuído o eixo Z para controle do efetuador.

O efetuador, representado pelo eixo Z, tem seu funcionamento como um deslocamento linear definido, desta forma, seu controle binário controla a existência ou não deste deslocamento considerando a ferramenta a ser manipulada, este deslocamento pode ser bem reduzido, cobrir pequenos deslocamentos com controle binário podendo ser feito por um solenoide, que basicamente é um eletroímã que desloca um êmbolo em um deslocamento conhecido, mantendo a simplicidade de controle, que é a finalidade do projeto.

Para promover o deslocamento nos eixos X e Y um atuador deve ser empregado, estes atuadores podem ser de diversos tipos, como cita Bardelli, (2005) diferentes tipos de motores podem ser usados em máquinas controladas por CNC, motores lineares, servomotores, motores de corrente contínua com *encoder* e motores de passos.

Segundo Brites e Santos (2008), os motores de passo são empregados em trabalhos em que a precisão de movimentos é necessária. Seu ponto forte não está em sua velocidade ou força

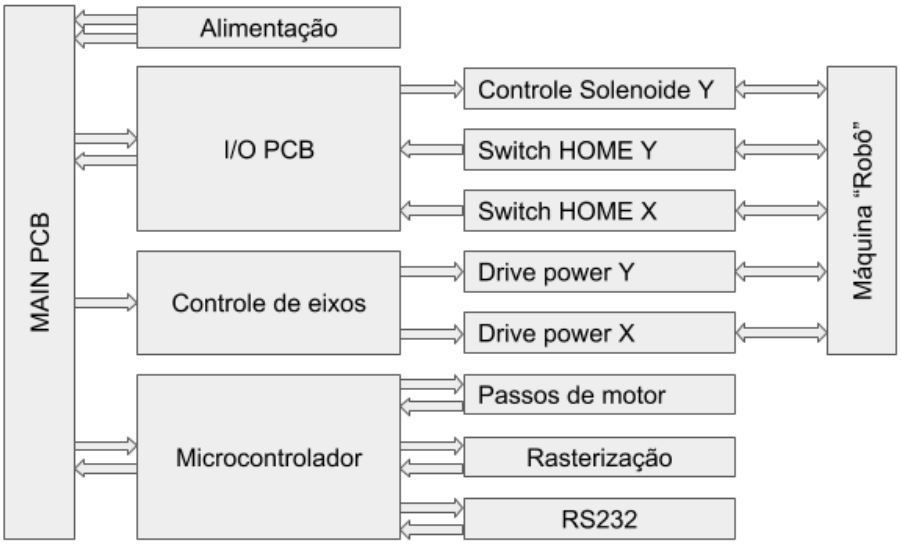
de rotação (torque), mas sim na capacidade com que se pode controlar seus movimentos, é por tal motivo que estes motores são amplamente utilizados em impressoras, robôs, automação industrial, dentre outros dispositivos que requerem precisão.

Utilizar motores de passo pode então dispensar o uso de *encoders* para aplicações menos críticas, onde alguns erros de deslocamento são toleráveis, e ainda garantindo relativa precisão, sendo assim, para realizar o deslocamento dos eixos X e Y optou-se por motores de passo, os quais podem ser encontrados facilmente em sucatas, principalmente de impressoras, o que pode baixar o custo do projeto drasticamente.

Para a comunicação com o foco na simplicidade do projeto o padrão de comunicação RS-232 foi considerado pois é utilizado em máquinas CNC comerciais e em projetos *open sources* Paiotti (2003) reforça afirmando que por ser muito simples e versátil se tornou padrão para aplicações de curtas distâncias, principalmente em comunicações entre máquinas, sensores, atuadores e também os computadores que os controlam. Sendo assim, a comunicação entre o computador e a UC deve ser feita por uma porta serial no padrão RS-232.

O robô deve conter UC capaz de gerenciar os comandos recebidos pelo computador e executa as tarefas necessárias para que sejam efetuados tais comandos, a tarefa de calcular a rasterização deve ser feita na UC que deve também ser capaz de gerenciar os motores de passo, o efetuador e os sensores de *home*. Com base no diagrama de Wang, Junyi Yu e Shangfeng Yi (1999), o diagrama da imagem 4 foi criado para servir de referência para a UC a ser desenvolvida, ele representa em diagrama de blocos os subsistemas que compõem a UC.

Imagem 4-Diagrama de blocos definido para a UC

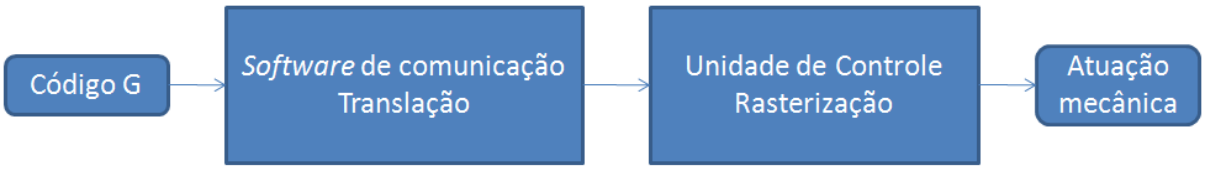


Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Para este desenvolvimento, foi decidido que o código G seja interpretado por um *software* no computador e enviado somente os deslocamentos de cada motor para a UC, haja vista que, a posição atual do efetuador no plano não é conhecida pela UC e sim pelo *software* no computador que é encarregado de calcular as translações, este inspirado nos *softwares* comerciais e *open source* como o Grbl de comunicação com máquinas CNC, capaz de ler arquivos de código G gerado por alguma outra ferramenta, ao mesmo tempo deve exibir para o operador funções básicas como a posição do efetuador e andamento do processo, além de possuir controles básicos como ajuste de velocidade, chamada do efetuador para a posição de *home* e a possibilidade de enviar comandos manualmente.

O diagrama da imagem 5 exibe a fluxo de processamento geral que deve ser feito para o robô funcione corretamente conforme o especificado.

Imagem 5-Fluxo de cálculos para definido



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Ainda, todas as etapas do desenvolvimento foram possíveis de serem realizadas sem o auxílio dos laboratórios da universidade, ou seja, foi possível o seu desenvolvimento em casa, seguindo a ideia que o Movimento *Maker* estimula.

4.2 Mecânica

Nesta seção é destinada ao desenvolvimento da parte mecânica do robô, ao qual propõem-se soluções para alguns problemas além de definições do seu formato físico.

Quanto à etapa de transmissão do deslocamento, Bardelli (2005) sugere que o deslocamento gerado pelo motor deve ser transmitido aos demais elementos do robô a fim de movê-los. Esta transmissão de deslocamento atualmente é feita largamente por fuso de esfera de alta precisão, por correias planas ou dentadas, barras roscadas padrão ‘M’ ou cremalheira.

Entre as maneiras citadas para promover o movimento, a barra roscada simples é o item mais barato e fácil de encontrar, lojas de ferragens possuem este item com diversos tamanhos e passos (deslocamento por revolução), apesar de ser um pouco diferente do fuso para esta

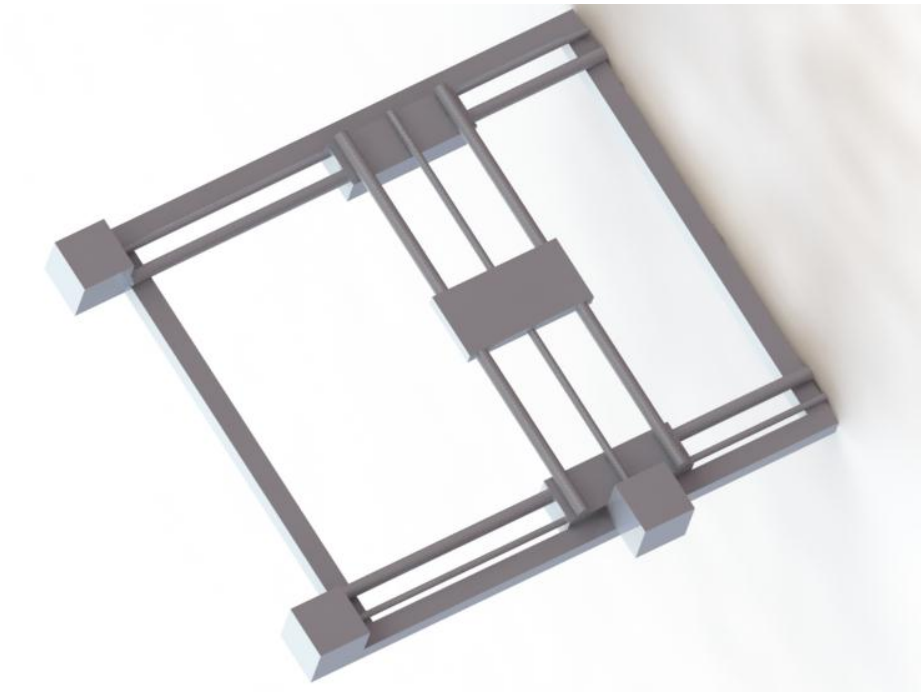
aplicação, que possui mais precisão e deslocamento por passo, é uma peça fácil de encontrar no mercado.

Para uma máquina CNC que utiliza barras roscadas, como explica Ardielli (2005), é indicado a utilização de acoplamentos elásticos, pois permitem que exista compensação de possíveis erros no alinhamento dos eixos durante o deslocamento.

Seguindo a recomendação do autor supracitado, os acoplamentos entre as barras roscadas e os motores devem poder se ajustar a pequenas falhas de alinhamento. Os acoplamentos elásticos promovem esse feito, contudo, considerando a falta de padrão nos componentes utilizados por serem de origem de sucata, encontrar um acoplamento elástico comercial que se adapte às diferenças dos componentes é muito difícil, para solucionar este problema, uma solução é o uso de um pedaço de tubo de silicone, assim o movimento pode ser transmitido pelo tubo e o mesmo pode se deformar para ajustar as pequenas imperfeições que podem existir no alinhamento entre o motor e a barra roscada, solução esta muito utilizada por hobistas em projetos de pequeno porte.

Para as juntas prismáticas, uma possível solução é o uso de barras lineares de impressoras, junto com as buchas de bronze que compõem a cabeça de impressão, ainda que retiradas de sucata existem barras semelhantes à venda, o que permite replicar o projeto com o uso de componentes novos.

Imagem 6-Esboço em 3D do robô cartesiano.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A imagem 6 ilustra os aspectos físicos e mecânicos do robô cartesiano construído, a modelagem em 3D foi feita no *software* CAD SolidWorks produzido por Dassault Systemes S. A., e representa a característica anteriormente definidas para o robô, a posição e identificação de cada componente e suas dimensões.

4.3 Desenvolvimento

Como a proposta do trabalho não é voltada para ferramentas CAN/CAD e sim para o controle de robôs por sistema CNC, propõe-se utilizar algum *software* capaz de gerar o código G e exportá-lo em um arquivo .nc para ser utilizado no projeto. Como já mencionado, diversas ferramentas CAM/CAD fazem tal procedimento, porém para minimizar tarefas fora do projeto foi utilizado o *software open source* PCIToGcode, o *software* é voltado para confecção de placas de circuito impresso com CNC, o *software* é capaz de ler uma imagem monocromática e transformá-la em sua representação vetorial e por fim em um código G que representa as bordas da imagem, o programa gera o código G no padrão correto para ser utilizado em qualquer CNC, desta forma, seu arquivo está adequado ao projeto proposto.

4.3.1 *Software* da UC

Seguindo a linha de manter a simplicidade geral da máquina desenvolvida, o *software* utilizado na UC deve ter o mínimo de processamento possível para o funcionamento, desta maneira menos recursos são necessários e menos *hardware* deve ser empregado.

Esta etapa de desenvolvimento da UC compreende a parte de controle de atuadores, sensores e do efetuador além da comunicação para tal, o desenvolvimento mecânico anteriormente detalhado define quantos sensores e atuadores deverão ser controlados, neste caso, são 2 atuadores compostos de motores de passo, 1 atuador composto por solenóide e 2 sensores para o *home*, sendo eles um micro interruptor para cada eixo, X e Y.

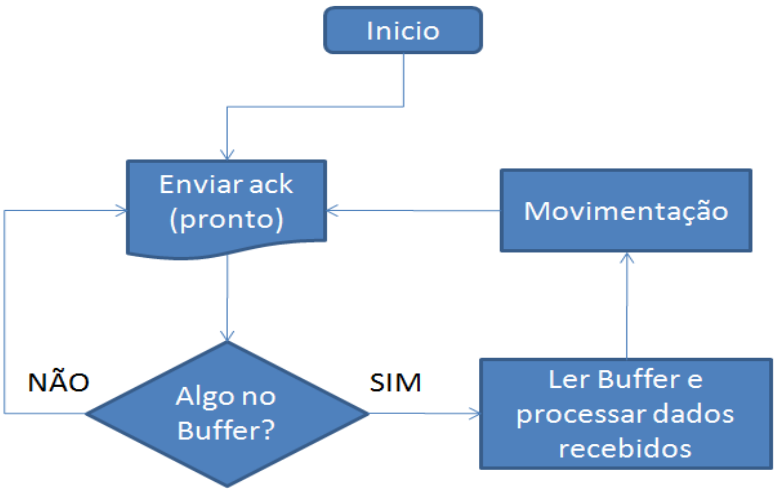
Para o desenvolvimento do *software* da UC, com base no microcontrolador escolhido, PIC16F628A, o ambiente de programação utilizado no desenvolvimento foi o CCS *Compiler* desenvolvido pela CCS Inc., que oferece um IDE e compilador para programação na linguagem C. Para auxílio na programação neste ambiente o livro ‘Microcontroladores PIC: Programação em C’, do autor Fábio Pereira voltado para compiladores CCS foi importante no processo de desenvolvimento.

4.3.1.1 Aquisição de dados e *feedback*

Como definido, os dados são transportados via serial RS-232, desta forma, a UC deve se comunicar ao computador para receber os dados que irá processar e informar ao computador que o trabalho foi realizado, e que uma nova instrução deve ser transmitida, arbitrariamente foi definido que o *feedback* enviado da UC para o computador será o caractere '2' somente, isto para que o mínimo possível de dados seja transmitido, e assim diminuindo o tempo perdido na comunicação de dados.

O fluxograma da imagem 7 mostra como a captura de dados e *feedback* é realizado.

Imagem 7-Fluxograma do sistema de captura de dados e feedback.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Usando o recurso de interrupção presente no microcontrolador proposto para uso, a UC fica constantemente enviando o caractere '2' que informa que está pronta para receber dados, ao detectar a presença de dados no *buffer* de sua porta serial, a interrupção faz a captura destes dados e desabilita a interrupção para iniciar o processamento dos dados, uma vez que os dados sejam processados e executados, e a interrupção volta a ser habilitada e a transmissão do *feedback* é iniciada até que novos dados sejam recebidos.

4.3.1.2 Tratamento de dados

Com os dados já recebidos e armazenados no *buffer*, esta etapa tem a finalidade de interpretar cada valor contido nos dados recebidos e encaminhá-los para os métodos corretos.

Considerando a limitação de processamento e de armazenamento de instruções do microcontrolador escolhido, o método a seguir foi criado para consumir o mínimo de recursos possíveis para fazer as conversões necessárias.

O *buffer* com os dados recebidos consiste de um vetor de inteiros com o número de campos correspondentes ao número de caracteres recebidos, cada campo armazena um valor inteiro correspondente ao valor decimal do caractere da tabela ASCII, desta forma, o caractere ‘0’ (zero) tem o valor inteiro 48, o ‘1’ (um) tem o valor 49 e assim sucessivamente, sabendo disso, basta uma subtração da constante inteira 48 para ter o valor individual de cada campo do *buffer*. Para caracteres não numéricos, neste caso (x, y, z, v e +) são identificados por seu decimal correspondente da tabela ASCII respectivamente 120, 121, 122, 118 e 43.

A primeira etapa deste método é testar se os dados recebidos que estão no *buffer* com um caractere passado como parâmetro, caso tal caractere exista na sequência contida no *buffer* o valor referente a ele é capturado e atribuído ao seu devido lugar, o método é chamado repetidamente para cada caractere de interesse, sendo eles: ‘+’ para o comando de *home*, ‘v’ para atribuição de velocidade e (X, Y e Z) respectivamente para cada coordenada.

Como a movimentação do efetuador se dá por passos do motor, cada variável de cada coordenada contém o número de passos que cada motor deve girar para promover o deslocamento desejado, o sentido da rotação dos motores é definido por valores positivos e negativos, sendo positivos para deslocamentos crescentes e negativos para deslocamentos decrescentes com referência ao ponto atual do efetuador.

O método *home* tem a finalidade de levar o efetuador até o ponto inicial de onde todas as atividades a serem realizadas devem partir, este ponto representa os valores de X e Y = 0, este método é chamado sempre que a UC for iniciada e quando solicitado pelo computador de controle. Este método faz a leitura dos interruptores que indicam o zero de cada coordenada, enquanto algum desses sensores permanecer aberto, o método de movimentação é chamado passando como parâmetro a movimentação de 1 passo no sentido decrescente, ou seja -1, até que o interruptor seja acionado, isso para cada coordenada, quando os 2 interruptores forem pressionados o método é encerrado e o sistema de *feedback* volta a operar.

4.3.1.3 *Hardware* da UC

Para atender os requisitos impostos pelas definições detalhadas anteriormente o *hardware* responsável pela parte central da UC deve ser capaz de supri-los, para isso, um número mínimo

de portas I/O devem existir e certa capacidade de processamento e armazenamento de código, além de possuir comunicação serial RS-232 ou TTL.

Para isso foi escolhido o uso de microcontroladores programáveis, a tabela 5 mostra os requisitos básicos que um microcontrolador deve ter para atender os requisitos existentes.

Tabela 5-Requisitos mínimos de *hardware*.

Requisito	Quantidade
Porta Serial RS-232/TTL	1
Terminais para motor X	4
Terminais para motor Y	4
Terminais para efetuador	1
Terminais para sensores *home*	2

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Os microcontroladores encontrados que atendem os requisitos de projeto, além de ter capacidade de memória para futuras expansões/alterações do *firmware* são os modelos PIC16F628A da Microchip Inc. e o AT90S2313 da ATMEL Corp.. Apesar de serem semelhantes o PIC16f628A possui custo relativamente menor, além de possuir extenso material didático para consulta.

Tabela 6-Características básicas.

Recursos	**PIC16F628A**	**AT90S2313**
I/O Pins	16	17
Memória Flash	3 Kbytes	2 Kbytes
Memória RAM	224 bytes	128 bytes
Clock máximo	20 MHz	10 MHz
Arquitetura	8 bits	8 bits
Serial TTL	1	1

Fonte: Adaptado dos *datasheets* PIC16F628A e AT90S2313.

O uso da plataforma Arduino também foi cogitado para uso na UC, porém como a proposta é a construção geral da UC do robô, o Arduino seria uma solução superdimensionada com muito mais recursos do que o necessário, além de ser uma plataforma de desenvolvimento e que não é voltada para aplicação final, apesar de possuir vasto material, inclusive *softwares* prontos já destinados a esta aplicação, não se encaixa a proposta.

Por estes fatores, foi escolhido o microcontrolador PIC16F628A para compor a UC a ser desenvolvida. Além do microcontrolador a UC deve conter reguladores de tensão para os motores e outros componentes, conversor de TTL para RS-232, controle elétrico do efetuador solenoide e terminais de conexões de cabos e dados, o detalhamento eletrônico está detalhado nos demais capítulos.

4.3.1.4 Motor de passo

Athani (2005) apresenta o motor de passo como sendo um motor de corrente contínua sem escovas, no qual o rotor gira a medida que os enrolamentos do estator são energizados, a rotação ocorre devida às interações magnéticas entre os pólos do rotor e a sequência com a qual o estator está sendo energizado, o giro do rotor se dá em incrementos angulares discretos.

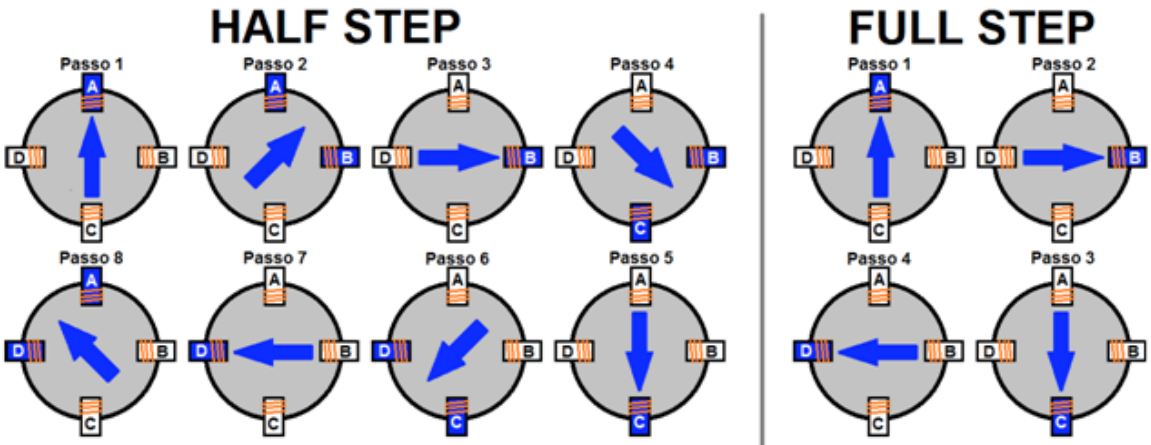
Os motores de passo são construídos de forma que sua rotação se faz por passos definidos e mensurados em graus, existem motores com diversos graus de deslocamento, o módulo de controle do motor deve conhecer estes valores e estar programado para trabalhar corretamente de acordo com a aplicação desejada.

Em virtude de sua precisão no deslocamento e facilidade de controle foi definido como o atuador responsável pelos deslocamentos das juntas do robô, acionando as barras roscadas.

Um motor de passo pode, entre outras maneiras, Segundo Brites e Santos (2008) trabalhar em *half step* e *full step*, cada pulso de controle equivale meio passo quando controlado em *half step*, sendo assim, garante mais precisão e maior torque, no entanto ocasiona maior dissipação de calor pelo motor e menor velocidade. No *full step*, cada pulso de controle equivale a exatamente um passo do motor, e é a maneira mais fácil e intuitiva de ser controlado, contudo o torque é menor e a velocidade é superior ao *half step*.

A imagem 8 representa os dois métodos de controle citados e uma matriz binária correspondente à alimentação em cada pino para um motor unipolar.

Imagem 8-Padrões de movimentação para controle half e full step.



Fios	Passos							
	1	2	3	4	5	6	7	8
A	1	1	0	0	0	0	0	1
B	0	1	1	1	0	0	0	0
C	0	0	0	1	1	1	0	0
D	0	0	0	0	0	1	1	1

Fios	Passos			
	1	2	3	4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Seguindo o padrão de controle *full step*, a matriz correspondente a este controle foi na UC, pois não haverá uso de *drive* para controle externo dos motores, a lógica para controle do motor é, neste caso, toda implementada via *software* na UC, outro aspecto importante é o tempo entre passos, existem limitações mecânicas que limitam o tempo mínimo do intervalo entre os passos do motor, tempo menor representa um torque menor e pode levar a perda de um passo devido a resistências mecânicas, que também deve ser controlado via *software*.

O trecho de código mostrada na imagem 9 representa a forma de controle *full step* implementada na UC para o controle dos motores.

Imagem 9-Métodos de controle de passos dos motores.

```
signed int posx = 0, posy = 0;    //posição dos motores
int sentidox, sentidoy;           // variaveis de sentido de giro dos motores
unsigned int m[4][4]={1,0,0,0,
                      0,1,0,0,
                      0,0,1,0,
                      0,0,0,1};
```

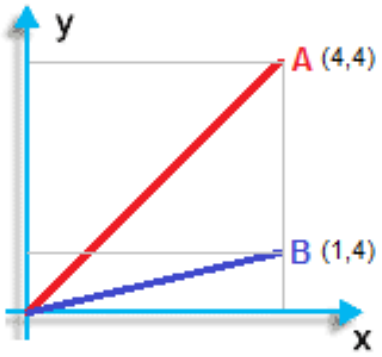
```
void movex() //movimento de 1 passo motor x
{
   if (sentidox == 1)
   {
      posx++;
      if (posx > 3)
         posx = 0;
   }
   else
   {
      posx--;
      if (posx <= 0)
        posx = 3;
   }
   output_bit( B1, m[3][posx]); //motor X
   output_bit( B2, m[2][posx]);
   output_bit( B3, m[1][posx]);
   output_bit( B4, m[0][posx]);
}
```

```
void movey() //movimento de 1 passo motor y
{
   if (sentidoy == 1)
   {
      posy++;
      if (posy > 3)
         posy = 0;
   }
   else
   {
      posy--;
      if (posy <= 0)
        posy = 3;
   }
   output_bit( B5, m[3][posy]); //motor Y
   output_bit( B6, m[2][posy]);
   output_bit( B7, m[1][posy]);
   output_bit( B8, m[0][posy]);
}
```

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Como são dois motores deste tipo a serem controlados, e durante sua operação tais motores devem operar em sentidos opostos ou em velocidades diferentes, deve haver uma relação entre o número de passos de cada motor para que a posição alvo seja alcançada, por exemplo:

Imagem 10-Diferença de deslocamentos no plano.

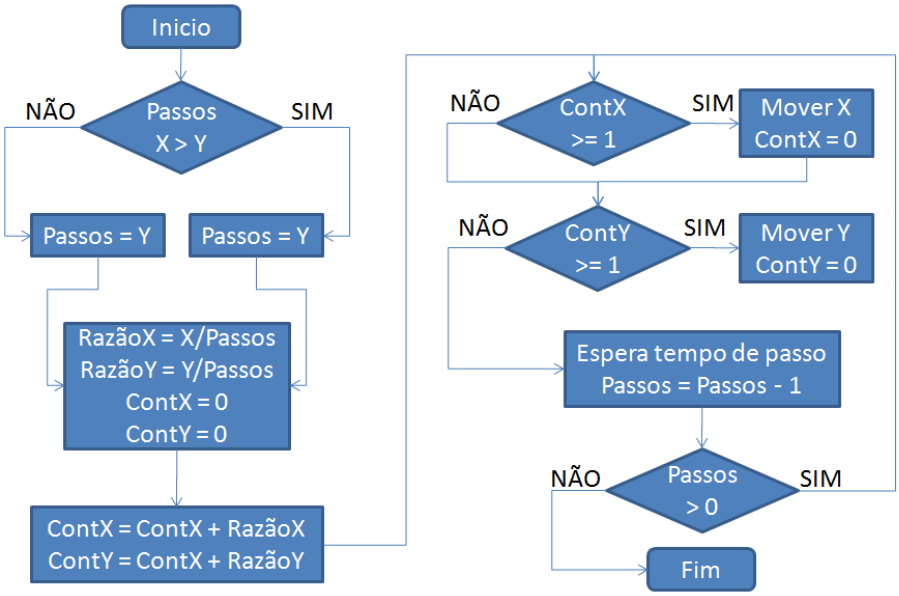


Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Na condição A da imagem 10, os 2 motores devem deslocar o mesmo número de passos para um deslocamento em 45°, o perímetro percorrido varia em função de número de passos, o passo da barra roscada e número de graus dos passos do motor. Na condição B, um dos motores deve ter seu número de passos reduzido para que um ângulo fora dos ângulos possíveis à 45° (0/360°, 45°, 90°, 135°, 180°, 225°, 270°, 315°) seja alcançado, a relação neste caso será feita com a divisão simples do número de passos do motor que terá menor número com a do motor de maior número, desta forma, um contador é incrementado com essa razão junto ao controle

de passos, esse incremento sempre que alcançar 1 ou maior que 1 (cont >= 1), representa um passo no motor de menor números de passos, cada incremento do controle de passos corresponde ao passo do motor de maior número de passos.

Imagem 11-Fluxograma controle de passos.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Seguindo o proposto no fluxograma representado na imagem 11 foi desenvolvido uma função para a programação deste na UC desenvolvida, esta função é composta de 2 etapas, uma responsável pelo controle da razão e do tempo mínimo entre passos, a outra etapa é responsável pela chamada do método secundário, este responsável por controlar a sentido da rotação e a alimentação correta do motor com base na matriz proposta para o padrão *full step*.

Esta função de controle dos motores para garantir a rasterização é a representação em linguagem C para esta aplicação do algoritmo incremental básico, segundo Lopes (2004) para melhorar o desempenha da rasterização de segmentos de reta é possível reduzindo o número de operações envolvidas, desta forma, o algoritmo imediato cuja expressão pode ser vista a seguir, pode ser reduzida para o algoritmo incremental básico:

Equação 1-Incremental básico

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$
$$b = y_1 - m\,x_1$$

$$\begin{aligned} y_{i+1} &= m\,x_{i+1} + b \\ &= m\,(x_i + 1) + b \\ &= m + m\,x_i + b \\ &= m + y_i \end{aligned}$$

Algoritmo Imediato

$$y = mx + b$$

Incremental Básico

$$y_{i+1} = m + y_i$$

Opções para conversão em inteiros

$$y = Round(m\,x + b) \quad y = Floor(0{,}5 + m\,x + b)$$

Fonte: LOPES, João Manuel Brisson. Rasterização. 2004.

Desta forma, o número de passos em y pode ser calculado para qualquer valor de x, porém em casos que o x é um valor menor que y o incremento dado por *m*, podem corresponder valores maiores que 1, que caracteriza, neste caso, mais de um passo de *y* para cada passo de *x*, como isto não é possível mecanicamente, a equação foi dividida em 2 casos, um quando $x > y$ e outro quando $y > x$, desta forma é possível garantir que o incremento *m* nunca será maior que 1.

As relações matemáticas implementadas podem ser expressas segundo a equação a seguir, vale ressaltar que o valor de destino para a coordenada calculada deve corresponder com o resultado do somatório da equação de rasterização utilizado.

Equação 2-Detalhamento incremental básico

Sendo a definido por:

$$a = \begin{cases} x\,, & x <= y \\ y\,, & senão \end{cases}$$

E k definido como a coordenada trabalhada, x ou y.

$$\sum_{i=1}^{n} \left(\frac{k}{a}\right).i$$

Fonte: LOPES, João Manuel Brisson. Rasterização. 2004.

A imagem 12 demonstra um exemplo da aplicação da equação modelada para formação do rastro correspondente ao vetor informado. É importante lembrar que o *software* de comunicação irá fornecer os valores para o vetor sempre em número de passos e sempre partindo do pressuposto que o efetuador se encontra no ponto (0,0) do vetor a ser rasterizado.

Imagem 12-Aplicação de cálculo de rasterização.

Aplicando rasterização para o vetor (0,0),(8,3).

$$\sum_{i=1}^{n} \left(\frac{k}{a}\right).i$$

Considerando que:

$k = y \quad a = x \quad n = 8$

X	passo X	Rastro X	Y	passo Y	Rastro Y
0	0	0	0	0	0
1	1	1	0,375	0	0
2	1	2	0,75	0	0
3	1	3	1,125	1	1
4	1	4	1,5	0	1
5	1	5	1,875	0	1
6	1	6	2,25	1	2
7	1	7	2,625	0	2
8	1	8	3	1	3

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A imagem 13 exibe a implementação da equação de rasterização para *software* da UC para gerir os deslocamentos do efetuador baseado em passos de motor.

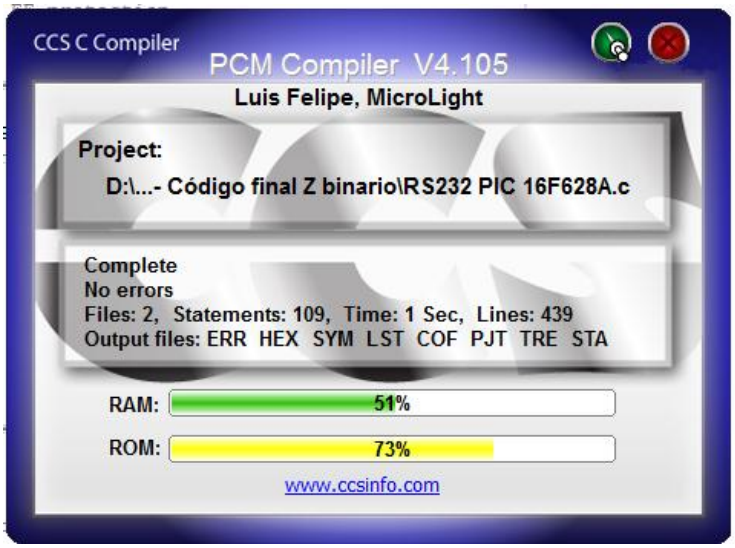
Imagem 13-Método de controle de passos.

```
void mover(unsigned int16 x,unsigned int16 y, unsigned int16 z)
{
     output_bit( AZ, z);
     unsigned int16 passos = 0;
     float quebrax = 0, contadorx = 0;
     float quebray = 0, contadory = 0;
     if      (x>=y)
        passos = x;
     else
        passos = y;
     quebrax = (float)x/passos;
     quebray = (float)y/passos;
     for(;passos>0;passos--)
     {
         contadorx = contadorx + quebrax;
         contadory = contadory + quebray;
         if (contadorx >= 1 ) //x
         {
            movex();
            contadorx = contadorx - 1;
         }
         if (contadory >= 1 ) //y
         {
            movey();
            contadory = contadory - 1;
         }
         delay_ms(velocidade);
     }
     limpar();
}
```

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Todo o código desenvolvido para a UC, que incluem todas as etapas, desde aquisição de dados, *feedback* e controle de motores e que deve ser gravado no microcontrolador, ocuparam 73% da memória disponível para código do microcontrolador, informação esta fornecida pelo IDE/compilador CCS.

Imagem 14-Consumo de memória do microcontrolador.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

4.3.2 *Software* de comunicação

Quando uma máquina controlada por CNC está conectada a um computador universal, *desktop*, a comunicação entre a máquina e o computador se faz, além do meio físico pelo *software* de comunicação que roda no computador.

Alguns *softwares* CAM/CAD já trazem embutido o sistema de comunicação, o programa Edgecam, por exemplo, possui seu sistema de comunicação direta por serial RS-232 localizado na aba Comms Setup dentro do programa, porém, existem *softwares* muito conhecidos, como o TurboCNC, Prorobotics, Mach3, ainda na linha de *softwares* comerciais, existem outros em que a UC deve rodar o mesmo código do fabricante do *software*, o que faz com que o usuário fique preso à empresa.

Voltando-se para ao lado *open source*, existem ferramentas como Linuxcnc uma ferramenta robusta e confiável, porém com limitação como rodar somente em ambiente Linux, e sua comunicação é voltada para dados paralelos e não seriais.

Como a ampliação em desenvolvimento neste trabalho possui um formato de dados diferente do código G para a comunicação com o computador, nenhum destes *softwares* será capaz de estabelecer a comunicação. Para este problema, o desenvolvimento de um *software* exclusivo para esta comunicação é uma solução, no qual deve ser capaz de ler um arquivo contendo o código G (.nc) gerado por alguma ferramenta CAM.

Apesar de o *software* necessitar fazer esta conversão do formato código G para número de passos de motor antes de enviar para a UC, todas as demais funcionalidades básicas devem existir, como controle direto de deslocamento, acionamento do *home*, capacidade de criar um falso *home*, alterar a velocidade e parâmetros de deslocamentos do robô e exibir a posição do efetuador no plano.

Para esta aplicação, quem detém a posição atual do efetuador é este *software* e não a UC como normalmente é feito. Para tal, o *software* tem a capacidade de ler os arquivos .nc gerados pelas ferramentas CAM/CAD que contém o código G que representa a movimentação da máquina pelo plano.

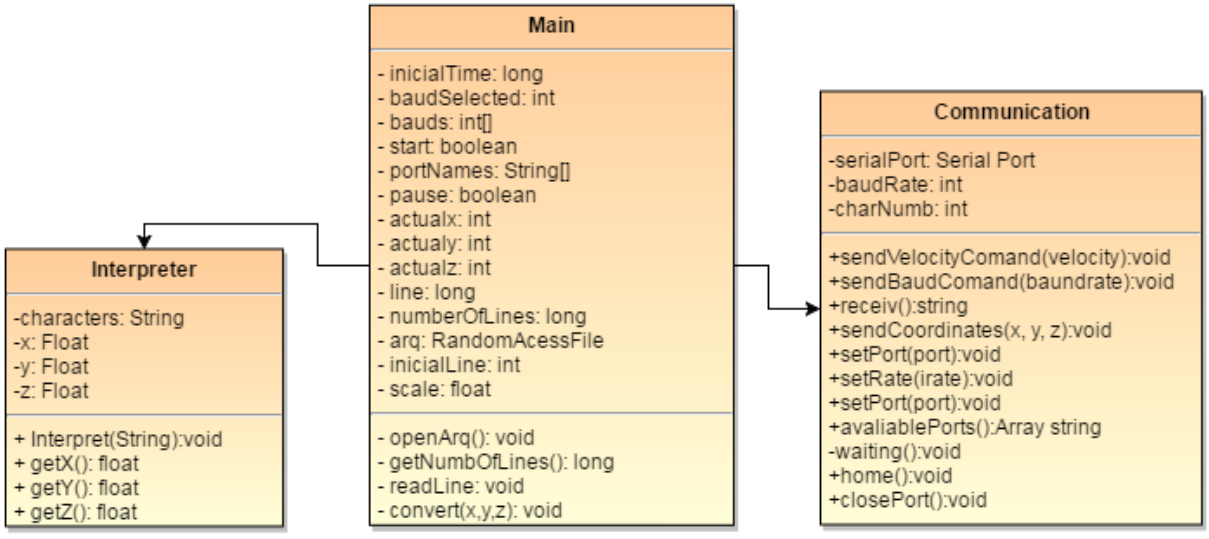
Com o objetivo de tornar o mais simples possível seu desenvolvimento, a linguagem Java foi escolhida por ser uma linguagem de fácil aprendizagem para quem já conhece C ou C++ e domina a orientação a objetos, além de tornar possível o conceito *write once* (escreva uma vez) e *run anywhere* (execute em qualquer lugar), como afirma Mendes (2009).

Sobre o conceito *run anywhere*, Mendes (2009) explica que a portabilidade de um programa Java é garantida pela (Java Virtual Machine - JVM), que é responsável por executar o *bytecode* gerado pelo compilador sobre o código escrito, uma vez gerado este *bytecode* ele pode ser executado em uma instância de qualquer JVM. Com a disponibilidade de JVM para diversas plataformas, um *bytecode* pode ser executado em diferentes sistemas operacionais, como: Windows, Solaris, Linux ou Mac OS.

Como definido, a porta COM no formato RS-232 para a comunicação na linguagem Java é necessária uma biblioteca externa para que a porta COM seja acessada. A biblioteca escolhida foi a JSSC, por ser de código aberto e possuir suporte a diversos sistemas operacionais. A classe Communication é responsável por manipular a biblioteca de comunicação, os métodos necessários que envolvem o uso da biblioteca estão contidos nesta classe.

Desta forma, o *software* de comunicação foi dividido em 23 classes, a classe "*Main*" que manipula a interface e realiza trabalhos sobre os objetos das demais classes, "*Interpreter*" responsável por fazer as interpretações das linhas em código G e classe "*Communication*" que faz a manipulação da porta de comunicação serial além de implementar alguns comandos fixos, como *home*, *set* e configurações de taxa de dados. O diagrama de classes apresenta na imagem 15 ilustra as classes descritas o relacionamento entre elas.

Imagem 15- Digrama de classes



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O método de conversão "*interpret*" é responsável por ler as linhas do código individualmente no arquivo, como cada linha representa uma instrução a ser executada, o programa então faz a leitura da linha, converte o formato de dados de texto para os formatos de ponto flutuante correspondente a cada coordenada em milímetros. Cada nova linha que

contenha valores para as coordenadas X e Y devem ser tratadas como uma translação do efetuador sobre sua posição atual. A equação 3 apresenta a equação de translação para planos, 2D, que se apropria para robôs cartesianos, como Craig (1989) indica:

Equação 3-Translação matemática

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \cdot \left[\begin{array}{cc|c} 1 & 0 & Tx \\ 0 & 1 & Ty \\ \hline 0 & 0 & 1 \end{array}\right]$$

ou

$$\begin{cases} x' = x + Tx \\ y' = y + Ty \end{cases}$$

Fonte: Adaptado de Craig. Introduction to robotics: Mechanics and control. 1989.

Um ponto importante é que os valores de X e Y representam a posição atual do efetuador e os valores de X’ e Y’ é a posição final após a translação definida por Tx e Ty, rém os valores de Tx e Ty são os valores a serem calculados, pois a posição atual é armazenada em variáveis internas do *software* e a posição final é fornecida pelo código G, desta forma, a equação a ser aplicada pode ser escrita como:

Equação 4-Dedução de Translação.

$$\begin{bmatrix} Tx \\ Ty \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \cdot \left[\begin{array}{cc|c} 1 & 0 & -x' \\ 0 & 1 & -y' \\ \hline 0 & 0 & 1 \end{array}\right]$$

ou

$$\begin{cases} Tx = x - x' \\ Ty = y - y' \end{cases}$$

Fonte: Adaptado de Craig. Introduction to robotics: Mechanics and control. 1989.

Assim, é possível conhecer em milímetros o deslocamento que o efetuador deve realizar para chegar ao ponto informado pelo código G, falta apenas converter este valor em milímetros para a representação de passos de motor, o valor referente à este fator é um dado informado na interface do *software*, visto que pode variar em função dos motores utilizados e das barras roscadas. Para adquirir este valor, basta fazer uma multiplicação simples pelo valor referente ao número de passos por mm, desta forma a equação final ficou definida como:

Equação 5-Detalhamento de translação

$$\begin{cases} Px = (x - x')\,Kx \\ Py = (y - y')\,Ky \end{cases}$$

Px = Passos no motor x Py = Passos no motor y
x = Posição x atual y = Posição y atual
Kx = Fatar de passos por milimtro em x
Ky = Fatar de passos por milimtro em y

Fonte: Adaptado de Craig. Introduction to robotics: Mechanics and control. 1989.

Uma vez que as informações de passos de motores sejam calculadas, estas podem ser enviadas para a UC do robô se encarregar da rasterização, durante esta etapa da UC o *software* fica bloqueado até que o *feedback* enviado pela UC seja recebido, a partir daí uma nova linha é lida, isto se repete até que seja alcançado o final do arquivo. As linhas de código G que carregam informações não implementadas pela máquina são descartadas, linhas com instruções do tipo troca de ferramenta, troca de velocidade entre outras são exemplos de instruções não implementadas e que serão descartadas.

Nos *softwares* utilizados para gerar o código G foi observado uma dificuldade em definir um deslocamento binário para o eixo Z, como o robô que está sendo controlado por código G foi desenvolvido com o eixo Z binário (0 ou 1) essa dificuldade foi tratada neste *software*, o tratamento então é feito considerando as variações dos valores do eixo Z quando arredondado para um número inteiro, desta forma, valores muito próximo de 0 são tratados como 0 e valores maiores, quando arredondados para 1 ou valores maiores que 1 o estado de Z então é definido como 1, desse modo, controlando a ativação ou não do eixo.

4.3.2.1 Detalhes do código fonte

Uma vez iniciada a interpretação de um arquivo .nc, o programa deve operar sozinho durante a leitura de cada linha do arquivo até que o final seja alcançado, porém isso faria a interface gráfica do *software* ficar “congelada” até o final da operação, desta maneira, não seria possível observar a posição atual do efetuador ou pausar a operação, para que a interface não fique bloqueada foi necessário a implementação de *threads*, dividindo as funções do programa em segmentos que podem ser executados de forma concorrente, sendo assim, foram necessárias mais duas *threads* além da padrão de um *software* qualquer.

A primeira delas é responsável por manter o *looping* de leitura do arquivo, chamar o método responsável pela conversão dos dados lidos do arquivo, chamar o método de envio que, por sua vez executa o método de espera por *feedback*, ao final de cada envio com o *feedback* da máquina, os dados da tela são atualizados.

Imagem 16-Thread de trabalho principal.

```
Thread t;
public void work(){ //rotina de leitura e processamento de cada linha G-code
        t = new Thread() {
            @Override
            public void run()
            {
                jImprimir.setEnabled(false);
                for(line=0;line<numberOfLines;line++)    //leitura individual de linhas
                    readLine();
                long milisegundos = System.currentTimeMillis();
                int segundos = (int) ((milisegundos - inicialTime )/1000);
                int minutos = segundos/60;
                segundos = segundos - minutos*60;
                milisegundos = milisegundos - ((segundos * 1000) + (minutos * 60 * 1000) + inicialTime);
                jImprimir.setEnabled(rootPaneCheckingEnabled);
                JOptionPane.showMessageDialog(null,"Fim do trabalho\nTempo gasto: "+
                        String.valueOf(minutos)+
                        ":"+
                        String.valueOf(segundos)+
                        ":"+
                        String.valueOf(milisegundos)+
                        " m,s,ms","Fim",
                    JOptionPane.INFORMATION_MESSAGE);
                inicialTime = 0;
                pause = false;
                jTlinha.setText("0");
                jPause.setForeground(Color.blue);
                jPause.setText("Pronto");
                try {
                      arq.seek(0);
                      } catch (IOException ex) {
                     Logger.getLogger(TelaPrincipal.class.getName()).log(Level.SEVERE, null, ex);
                }
                numberOfLines = 0;
            }
        };
        t.start();
}
```

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A outra *thread* é responsável por enviar os comandos de movimentação manual, ela captura os valores digitados na interface, converte para os parâmetros corretos e envia para a máquina.

Imagem 17-Thread de envio manual.

```
Thread e;
public void send(final int control){ //envio manual, direcionais ou coordenada especifica
        e = new Thread() {
            @Override
            public void run()
            {
                int x = 0, y = 0, z = 0, tempx, tempy;
                switch(control)
                {
                    case 0: x = round(Float.valueOf(Textx.getText())*scale); //manual
                            y = round(Float.valueOf(Texty.getText())*scale);
                            if (jCBstatez.isSelected())
                                z = 1;
                            else
                                z = 0;
                            break;
                    case 1: x = round(Float.valueOf(jTdeslocamento.getText()) * scale)+actualx; //left
                            y = actualy;
                            z = actualz;
                            break;
                    case 2: x = round(-Float.valueOf(jTdeslocamento.getText()) * scale)+actualx; //rigth
                            y = actualy;
                            z = actualz;
                            break;
                    case 3: y = round(Float.valueOf(jTdeslocamento.getText()) * scale)+actualy; //up
                            x = actualx;
                            z = actualz;
                            break;
                    case 4: y = round(-Float.valueOf(jTdeslocamento.getText()) * scale)+actualy;  //down
                            x = actualx;
                            z = actualz;
                            break;
                    case 5: y = actualy; //acionar ferramenta
                            x = actualx;
                            if (actualz == 1)
                                z = 0;
                            else
                                z = 1;
                            break;
                }
                tempx = x;
                tempy = y;
                x = x-actualx;
                y = y-actualy;
                actualz = z;
                actualx = tempx;
                actualy = tempy;
                send(x,y,z);
            }
        };
        e.start();
}
```

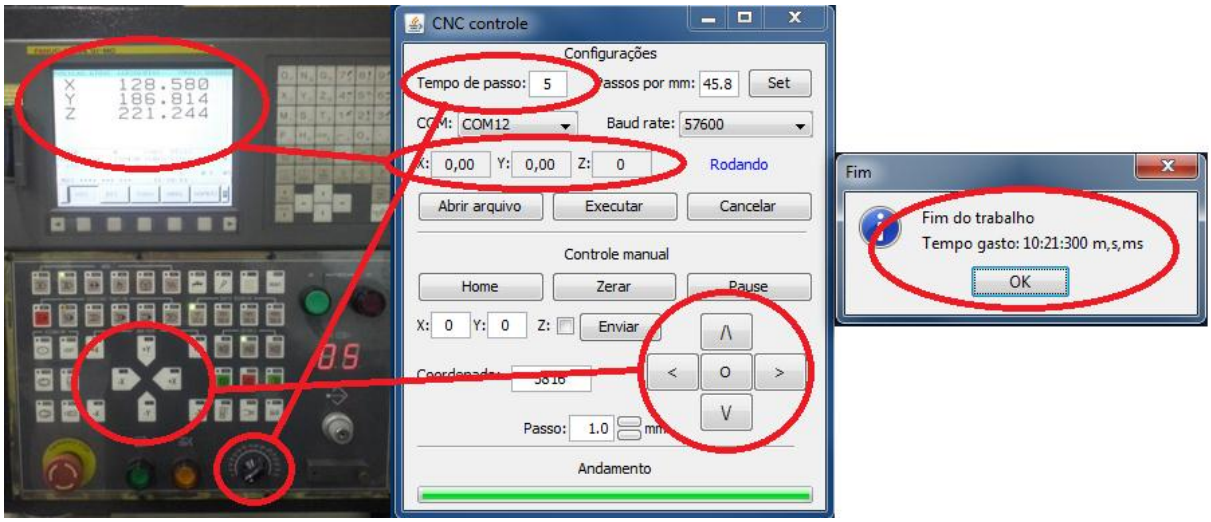
Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A interface final do *software* de comunicação pode ser vista na imagem 19, os requisitos de chamada do efetuador para o *home* foi implementado no botão “Início”, o recurso de falso *home* está presente no botão “Zerar” que executa um *reset* nas variáveis que detêm a posição

atual do efetuador, desta forma, os cálculos internos reconhecem o ponto atual como o *home*, o botão "*Set*" faz a configuração de velocidade e escala, que é a relação de revolução do motor pelo passo da barra roscada, no qual se mostrou importante um controle ajustável e não fixo, visto que podem haver motores e barras roscadas com propriedades diferentes, o botão "Enviar" envia os comandos manuais digitados nos campos respectivos, todos os deslocamentos devem ser informados em mm, ainda no comando manual é possível deslocar o efetuador ao clicar nas seta direcionais e o valor de "passo" corresponde ao deslocamento de cada interação no direcional, o centro do direcional corresponde ao acionamento do solenoide, e por fim o botão "Executar" que inicia a leitura do arquivo .nc contendo a programação em código G e transmissão para o robô. Adicionalmente o botão "*Pause*" foi implementado, este tem a função de pausar a transmissão das instruções do arquivo .nc e retomar quando clicado novamente, e a opção de "Coordenada" corresponde à linha lida atualmente no arquivo .nc, com este recurso é possível informar uma linha específica para o início da "impressão".

Ao fim de cada trabalho realizado é exibido o tempo decorrido durante o processo, este tempo é contabilizado com o armazenamento do tempo atual do computador no momento do clique em "Executar" e novamente ao final da leitura do arquivos, a diferença nestes dois valores corresponde ao tempo total envolvido no processo, este valor de tempo é contabilizado em milissegundos e corresponde ao tempo de processamento e execução dos comando pelo computador e pelo próprio robô, desta forma, o processamento do computador pode interferir no tempo total de execução.

Imagem 18-Interface do *software* de controle desenvolvido.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

4.4 Eletrônica da UC

Considerando limitações existentes em cada componente eletrônico alguns ajustes no circuito devem ser feitos para garantir o funcionamento correto de todos os componentes.

Como o responsável pelo controle de motores e do solenoide do efetuador é o microcontrolador PIC16F628A, é importante conhecer seus limites a serem respeitados.

Tabela 7-Propriedades do microcontrolador.

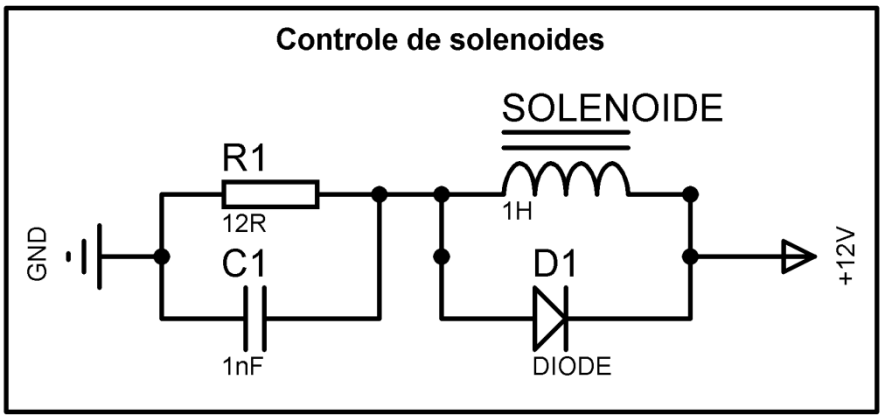
Descrição	Limite
Tensão de alimentação	2.0V até 5.5V
Corrente máxima nos pinos I/O	25 mA
RS-232	Nível TTL
Consumo máximo	250 mA

Fonte: Adaptado do *Datasheet* PIC16F628A. 2007.

Com o limite de corrente de 25 mA por pino do microcontrolador, os motores e o solenoide não podem ser ligados diretamente ao microcontrolador, para este problema as soluções a seguir são propostas. Mussoi (2005) explica que um solenoide é constituído de uma longa bobina de fio isolado e enrolado em espiras lado a lado, como um indutor, quando energizada o campo magnético criado em cada uma das espiras é somado, o resultado é um campo magnético igual ao de um imã permanente. Como um solenoide nada mais é que um indutor, suas propriedades devem ser consideradas no desenvolvimento da placa.

Em virtude destas propriedades, Rako (2013) propõe que o circuito de controle de um solenoide deve acionar o mesmo com uma tensão elevada para que ele possa movimentar o êmbolo, e em seguida a tensão deve ser reduzida, junto com a corrente para evitar que o solenoide se aqueça em demasia.

Imagem 19-Circuito proposto por Rako para controle simples de solenoide.



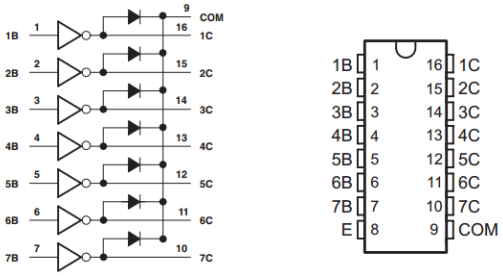
Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Como o solenoide a ser usado é oriundo de sucata, seus dados de indutância L e resistência interna RL não são conhecidos, a definição dos componentes R1 e C1 se deu na fase de desenvolvimento mecânico quando testes práticos puderam ser realizados.

Para o controle dos motores de passo, como dito anteriormente, não podem ser ligados diretamente aos terminais I/O do microcontrolador, uma solução é a aplicação de transistores para fazer o chaveamento da corrente com o acionamento do microcontrolador, dessa forma, a corrente flui pelo transistor e não pelo microcontrolador.

O CI ULN2003 como diz o fabricante em seu *datasheet* é composto de uma matriz de transistores de diodos *Darlington* para alta corrente e tensão, a matriz é composta de sete pares de transistores NPN *Darlington* com catodo comum e diodos para comutação de cargas indutivas.

Imagem 20-Ilustração do CI ULN2003 do fabricante.



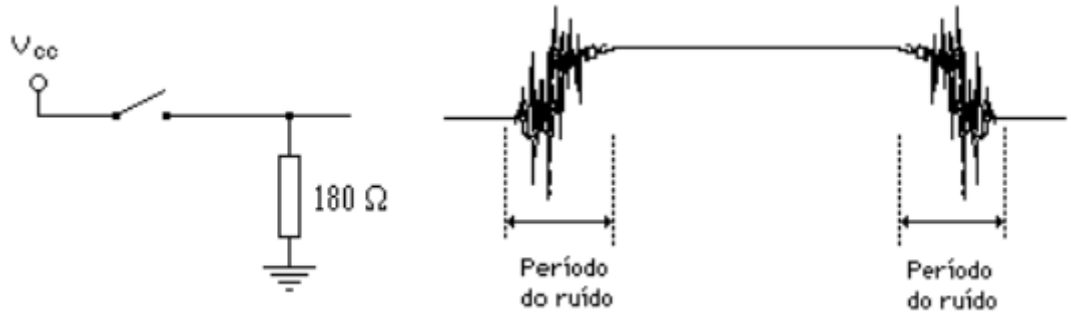
Fonte: Adaptado do *datasheet* CI ULN2003.2016.

O *datasheet* do CI ainda sugere seu uso para controle de motores, relés, lâmpadas, entre outras aplicações que o chaveamento por transistor em até 500 mA seja aplicada, desta forma, este CI foi escolhido para o chaveamento dos motores e do solenoide do motor.

Os interruptores de *home* do robô também precisam de uma atenção quanto à sua eletrônica, o Prof. Luiz Pinto [2008?] explica que lógicas TTL não reconhecem com segurança os níveis lógicos "1" ou "0" quando os terminais estiverem abertos, para isso é necessário o uso de um resistor *pull-up* para garantir o nível lógico desejado.

Com esta afirmação, um resistor então deve ser ligado de forma a manter o nível lógico baixo "0" no terminal do microcontrolador quando o interruptor estiver aberto, quando o seu estado mudar, o interruptor fornecerá tensão para o pino do microcontrolador e assim garantirá o outro nível, "1".

Imagem 21-Representação Pull-up.



Fonte: PINTO, Luiz. Técnicas com Sistemas Digitais. [2008?].

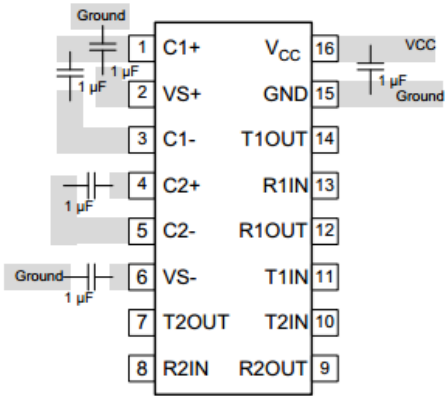
Quanto à comunicação, o microcontrolador possui sua porta de comunicação serial em nível TTL que, como explicado por Orlandini (2011), possui níveis lógicos de 0V a 0,8V para "1" e 2,4V a 5V para "0", já o padrão RS-232 possui seu nível "0' com tensões entre 3V e 25V e seu nível "1" com tensões entre -25V e -3V.

Para poder se comunicar com um computador comum, o microcontrolador precisa que seu nível lógico seja padronizado, Orlandini (2011) propõe em seu trabalho que transceptores, ou *transceivers* tem a responsabilidade de converter padrões diferentes de comunicação serial para o nível utilizado no circuito, por exemplo, 5V para alto e 0V para nível baixo, com isso, um dispositivo TTL conectado no adaptador de comunicação serial não precisa de conversão, porém quando conectado em uma porta com níveis diferentes, o uso de transceptores é inevitável.

Orlandini (2011) ainda detalha alguns transceptores que podem ser usados para esta aplicação, como o CIs FT232R, ST485 e MAX232, apesar de semelhantes, o CI MAX232 consegue converter sinais Rx, Tx, CTS e RTS em RS232 ou TLL, em ambas das direções de dados a partir de uma fonte simples de 5 volts.

Com isto, o uso do CI MAX232 foi escolhido para fazer a conversão necessária para promover a comunicação do microcontrolador com o computador.

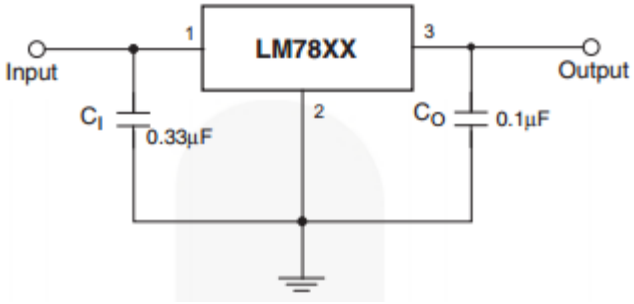
Imagem 22-Layout de montagem do CI MAX232.



Fonte: Adaptado do *datasheet* CI MAX232.2014.

Por fim, a alimentação de todo o sistema deve ser controlada, motores e solenoides trabalham em tensões diferentes e diferentes também dos CIs, para isso foi escolhido usar o regulador de tensão LM7805 para garantir a tensão de 5 V para os CIs, quanto aos motores e solenoides, a alimentação será diretamente da fonte principal de 12 V.

Imagem 23-Layout de montagem típico para reguladores LM78XX.

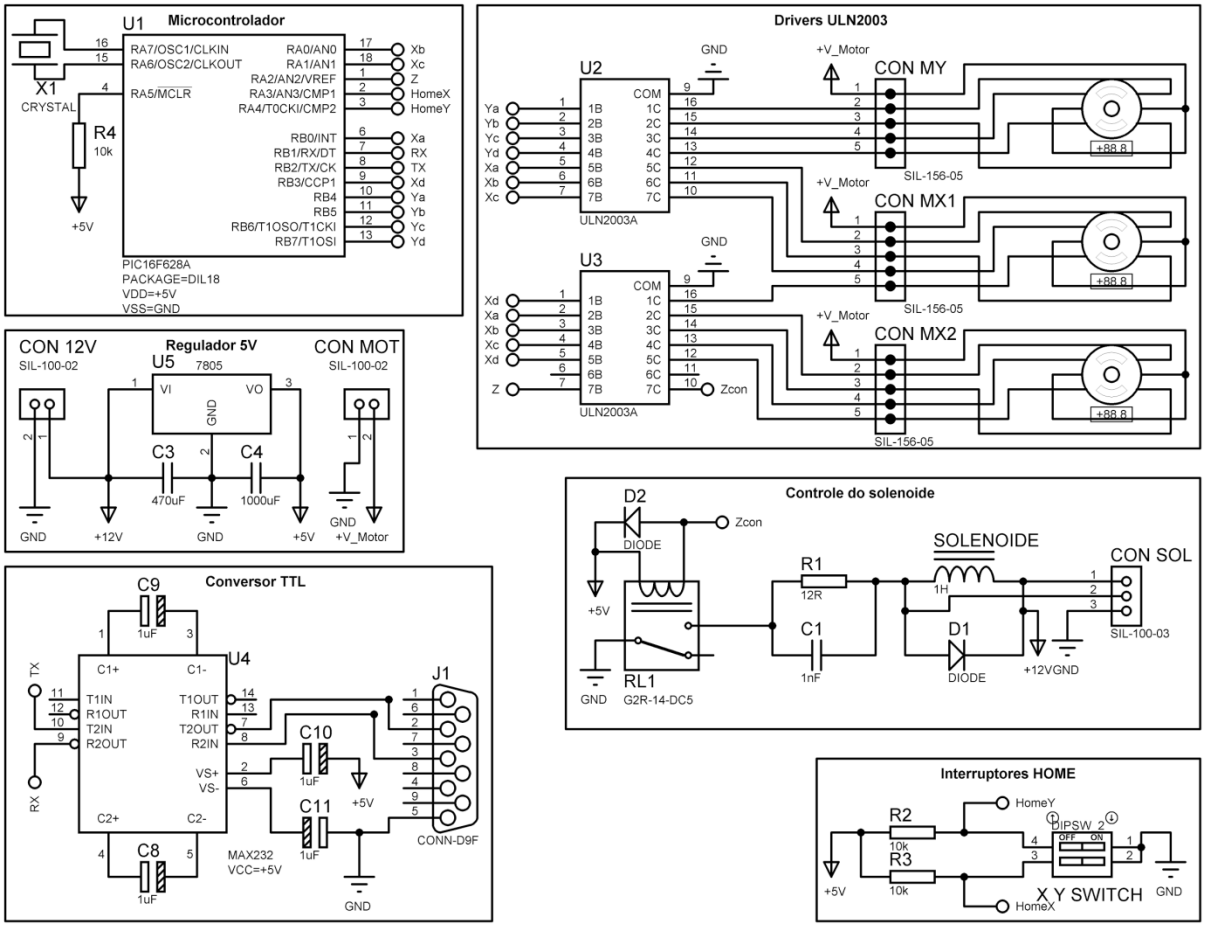


Fonte: Adaptado do *datasheet* CI LM78XX.2014.

A imagem 25 apresenta o circuito eletrônico utilizado na construção da UC, ele foi construído utilizando o mínimo de componentes possível. A quantidade reduzida de

componentes e o fato de serem baratos e de fácil aquisição favorecem a utilização do circuito e de todo o projeto como base para estudos e práticas de laboratório.

Imagem 24-Circuito eletrônico da UC.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O funcionamento do circuito pôde ser testado em simulador antes de ser montado com componentes reais, desta forma o *firmware* criado pôde ser testado mais facilmente do que seria se fosse necessárias diversas gravações do mesmo no microcontrolador, o circuito apresentado se mantém fiel circuito real utilizado na UC do robô.

5 CONSTRUÇÃO

Este tópico detalha as etapas de construção do robô, a construção foi dividida em aquisição de material, montagem das partes mecânicas e eletrônicas, programação do microcontrolador, testes das partes montadas, e a calibração.

5.1 Materiais

Os materiais utilizados para a construção física do robô são em sua maioria oriundos de sucata de equipamentos eletrônicos, juntamente com os demais materiais, que tiveram de ser comprados estão listados na seção custos em avaliações sobre o projeto.

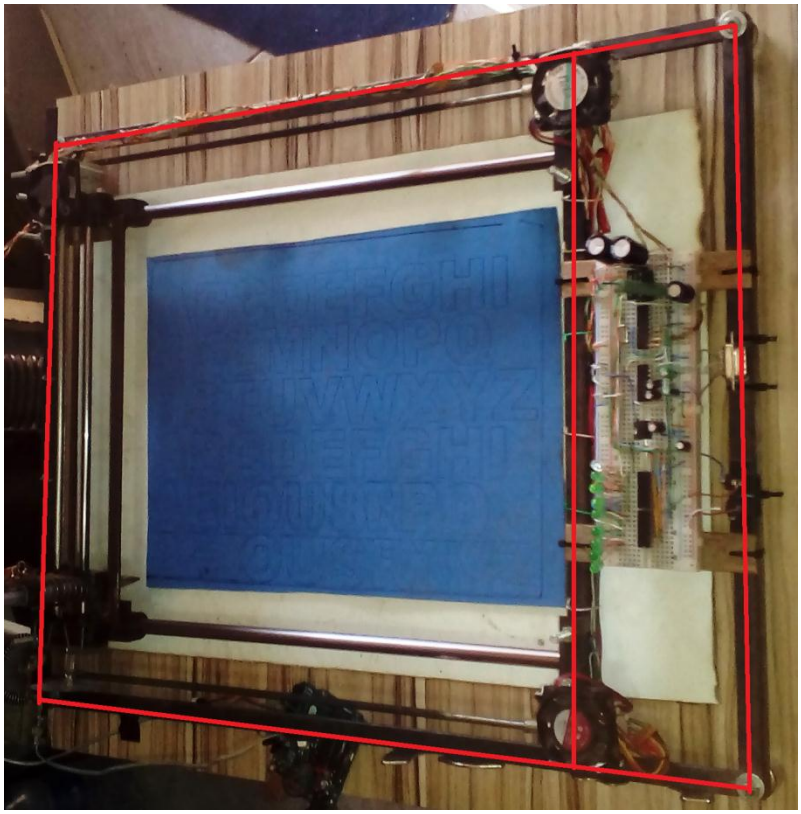
A utilização de componentes de fácil aquisição é parte considerada no projeto, visto que este se propõe a ser utilizado como referência para práticas didáticas ou mesmo até replicado por hobistas, desta forma o acesso à componentes semelhantes devem ser possíveis. A montagem e preparação dos materiais também é parte importante neste quesito, deve-se considerar que equipamentos específicos são de difícil acesso para a maioria dos desenvolvedores não profissionais, desta forma, o processo de montagem e construção se baseia em utilização apenas de ferramentas manuais e comuns, sendo a ferramenta menos comum utilizada o soldador de eletrodo revestido, que seu uso pode ser substituído por fixação por parafusos, o que dispensa soldas.

Uma consideração importante de ser mencionada sobre o material utilizado é sobre os motores de passo, os motores utilizados neste projeto são de origem de sucata, sendo assim, a tensão de alimentação dos motores pode variar em relação ao circuito, para isso o projeto eletrônico possui um conector nomeado de CON_MOT, este conector prove a fonte de energia aos motores, esta fonte é independente do resto do circuito pois a tensão de trabalho dos motores pode variar bastante, e considerando componentes retirados de sucata o circuito não pode ser "fixo" a tal ponto de permitir um único tipo de motor, sendo assim, os motores utilizados podem trabalhar em tensão diferente do resto do circuito. Os Motores utilizados são 3 motores da marca NMB modelo PM35L-048 que trabalham com tensões de 24 Volts, desta forma foram necessárias 2 fontes de 12V para alimentar todo o sistema, uma fonte12V alimenta o circuito pelo conector CON_12V e a outra fonte é ligada em série com a primeira fonte, o que resulta em 24 Volts, e esta tensão é ligada ao conector CON_MOT para a alimentação dos motores.

5.2 Montagem mecânica

Na etapa de construção da base do robô foram utilizados 220 Cm de cantoneira de ½ polegada. A união entre cada segmento utilizado foi feita com solda elétrica de eletrodo revestido, todos os ângulos são de 90º e as dimensões finais são de 500x400 mm na parte externa, a imagem 26 mostra a base do robô já soldada e as áreas de utilidade identificadas.

Imagem 25-Base soldada do robô.

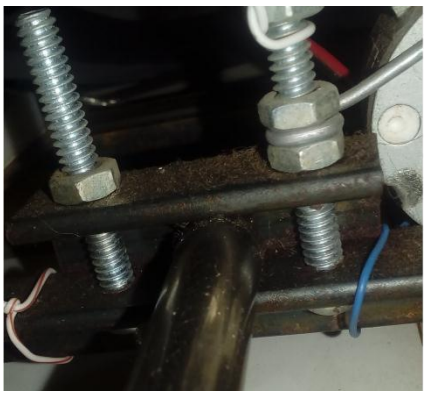


Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Com a base montada, a ordem de montagem definida foi de fixar as barras lineares para deslocamento do efetuador no eixo X, os motores do eixo X, as barras roscadas referentes ao mesmo eixo, o acoplamento e os rolamentos de folga axial.

As barras lineares foram fixadas com o uso de pedaço de 45 mm da mesma cantoneira já utilizada e dois parafusos em cada extremidade da barra, desta forma que ao afrouxar os parafusos a barra pode ser ajustada facilmente.

Imagem 26-Fixação das barras do eixo X.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Os rolamentos para compensar a folga axial foram fixados com uso de uma chapa de metal parafusada à estrutura, também é possível ajustar sua posição ao afrouxar os parafusos. Estes rolamentos são importantes para retirar a folga radial, no mesmo sentido do eixo, que pode ocorrer no momento que o motor acionar o efetuador, essa folga pode comprometer o resultado final.

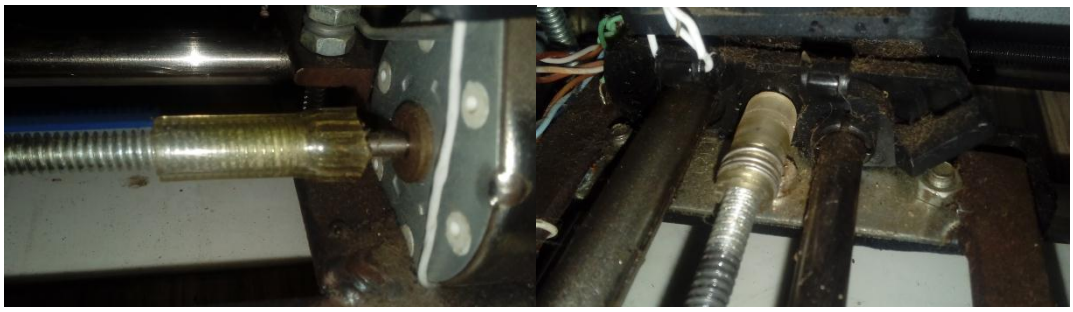
Imagem 27-Rolamento compensador para folga axial.

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

As barras roscadas que promovem a conversão do movimento circular do motor em um deslocamento linear para o efetuador, são fixadas ao rolamento por meio de porcas. Já ao motor por meio de um acoplamento elástico, no caso deste projeto em que os motores são retirados de sucata existe uma dificuldade em manter o padrão de algumas partes entre os motores, em vista disto, cada motor já possui uma engrenagem acoplada em seu eixo e removê-las de forma inadequada poderá danificar o motor, por esse motivo, encontrar algum acoplamento comercial que se adapte nesta condição é muito difícil. Diante disso, foi então optado por utilizar 2 segmentos de tubo de silicone com 25 mm de comprimento para fazer o acoplamento. O silicone é facilmente adaptável para esta aplicação, por sua flexibilidade é possível utilizar como uma

junta elástica, visto que pode se moldar facilmente à pequenas imperfeições e, mesmo assim conduzir o giro do motor para a barra roscada.

Imagem 28-Acoplamento dos motores.

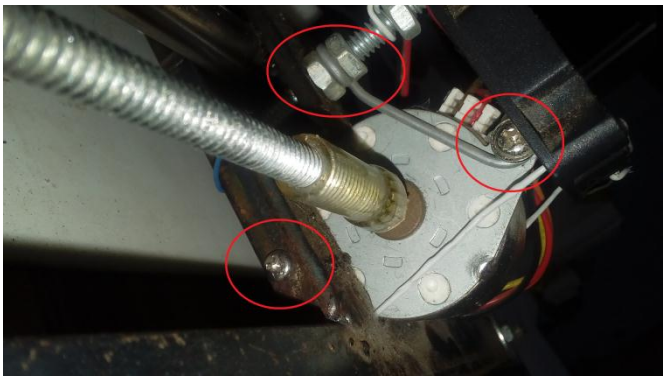


Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Para o eixo X foram utilizadas 2 barras roscadas com 420 mm de comprimento cada, juntamente com um rolamento, motor e junta elástica para cada.

Os motores foram fixados com um parafuso diretamente a estrutura e outro ligado a uma barra de metal que, por sua vez, é parafusada na estrutura, desta forma, também é possível ajustar a posição do motor se necessário.

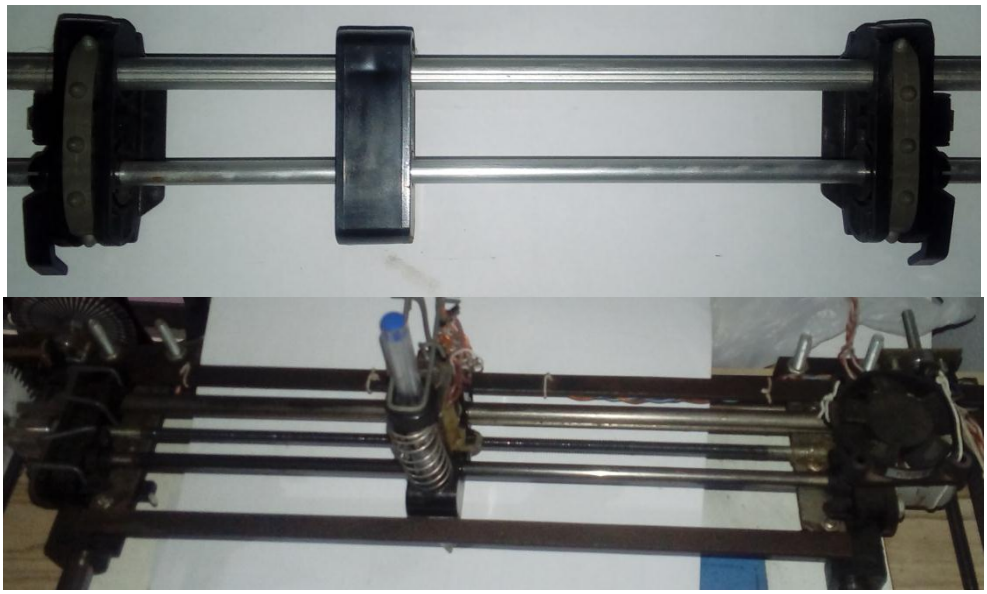
Imagem 29-Fixação dos motores.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A junta responsável pelo movimento do efetuador em Y é composta também por 2 barras lineares, rolamento para folga axial, barra roscada, junta elástica e motor, todos estes componentes são montados sobre o eixo X. Todos estes componentes foram montados em um bloco separado e ao final fixados às castanhas do eixo X.

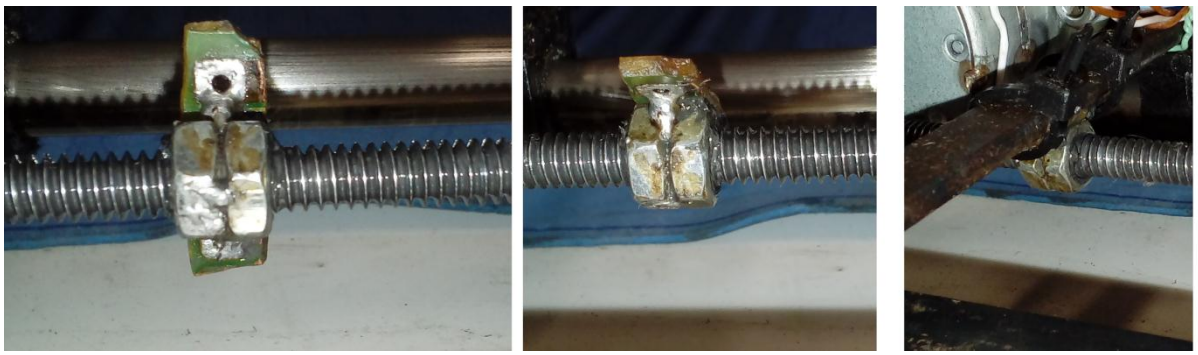
Imagem 30-Detalhamento da junta Y.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Em ambas as barras roscadas que compõem o robô foram utilizadas porcas para conduzir o movimento linear da barra para a junta, fazendo a função de castanhas, peça utilizada para tal finalidade, como uma única porca trabalha com certa folga, o que pode prejudicar a precisão do robô, uma alternativa encontrada foi unir 2 porcas com uma leve diferença de alinhamento entre elas, a união foi feita por soldagem com estanho, e junto com o processo de soldagem foi adicionado uma base para que possa ser fixada à junta correspondente.

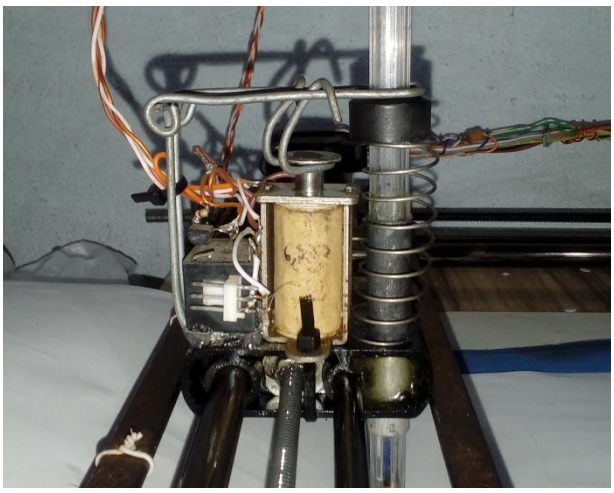
Imagem 31-Porcas operando como castanhas.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Por fim, no eixo Y é montada a última junta deslizante, na qual ficará posicionado o solenoide que compõe o efetuador. A base deslizante em que o efetuador é montado possui 54x20 mm, o solenoide é montado na face superior da junta de forma a seu acionamento provocar um deslocamento vertical na ferramenta controlada na base da junta, ao lado do solenoide, sendo assim foi adicionado um tubo de alumínio de 95 mm de diâmetro e 50 mm de comprimento, na qual a ferramenta irá se deslocar, por fim, uma mola tem a função de retornar a ferramenta para o estado inicial quando o solenoide estiver inativo.

Imagem 32-Efetuado, eixo Z binário.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Com a montagem dos eixos e do efetuador completos, a parte mecânica do robô está concluída e os testes podem ser realizados.

5.3 Testes individuais

Após a montagem dos componentes mecânicos é necessária a verificação do correto funcionamento, partes móveis podem emperrar ou oferecer resistência mecânica demasiada, por esse fato foram realizados testes individuais para fazer pequenas correções e ajustes a fim de garantir o perfeito funcionamento. Foram testados o deslizamento das juntas X e Y, o acionamento das mesmas pelos motores e o efetuador juntamente com o dimensionamento dos componentes para garantir o correto funcionamento do solenoide.

O deslocamento das juntas X e Y é efetuado pelo deslocamento proporcionado pela porca deslizando pela barra roscada, por sua vez a barra roscada é atuada pelo motor de passo correspondente e acoplados por uma junta elástica, para testar todo este conjunto em ambas as

juntas do robô um código básico foi gerado para a plataforma Arduino com a finalidade de manter um motor de passo rotacionando no sentido crescente, sentido em que o efetuador se afasta do *home*, juntamente com o Arduino é utilizado um CI ULN2003 para não exceder o limite de corrente das saídas do Arduino.

Como descrito anteriormente, a junta X do robô é movimentada por 2 motores de passo e não apenas 1 como na junta Y, este fato faz com que os 2 motores necessitam de um sincronismo em seus deslocamentos, mecanicamente isto é garantido com o uso de componentes de mesmas características, eletronicamente cada motor deve ter o seu drive de controle individual, pois, cada terminal do CI ULN2003 suporta uma determinada corrente e alimentar 2 motores simultaneamente poderá exceder este limite. Como já apresentado no circuito eletrônico, o número de terminais resultantes do uso de 2 CIs ULN2003 resulta em terminais suficientes para ligar todos os 2 motores e controlar o solenoide, então para o teste foi utilizado também 2 CIs para garantir o número de terminais suficiente para cada motor e para garantir o sincronismo, a entrada de controle dos CIs vinda do Arduino neste caso, é replicada de um motor para o outro, semelhante ao circuito da UC. O deslocamento de cada eixo foi testado individualmente acionando um motor de cada vez, estes testes foram importantes para garantir o funcionamento correto de cada junta do robô.

O efetuador também foi testado individualmente não só para garantir seu funcionamento, mas para dimensionar o circuito eletrônico para seu controle, como a peça é proveniente de sucata seu valor de indutância L é desconhecido considerando o uso de ferramentas básicas. Com o uso de um multímetro foi possível medir a resistência interna do solenoide que possui 6,2 ohms, desta forma, foi possível calcular a resistência necessária para manter a corrente em 750 mA utilizando a equação da lei de ohm:

$$V=I/R$$

V é a tensão em volts, I a corrente e R a resistência.

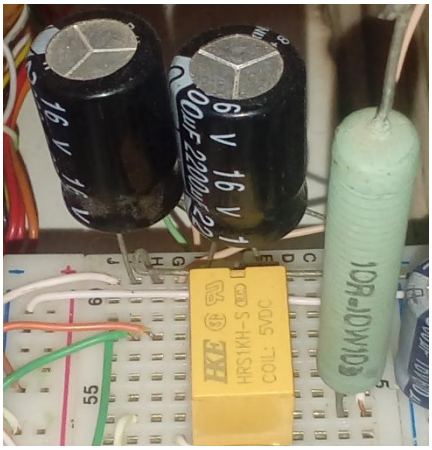
A resistência calculada foi de 16 ohms, subtraindo os 6,2 ohms da resistência interna do solenoide, foi encontrado um valor de 9,8 ohms, porém comercialmente existe um número limitado de resistências fabricadas, a mais próxima encontrada foi de 10 ohms. Outro ponto importante é a potência que esse resistor vai ter de suportar, neste caso com uma corrente de 750 mA e 12 volts foi possível calcular que o resistor deve suportar, a potência em watts pode ser calculada multiplicando a corrente total de 750mA ou 0,75A pela tensão de trabalho, 12V, desta forma encontrando o valor de 9Watts, novamente existem limitações comerciais para

estes valores. Um valor comercial de potência W facilmente encontrado no mercado é de 10W, o que atende a necessidade e a disposição do componente.

O capacitor C1 responsável pelo pico de corrente para o acionamento do solenoide foi definido por experimentação, pois as restrições mecânicas que ocorrem no efetuador como o deslocamento da ferramenta dentro do tubo de alumínio e a compressão da mola fazem variar a corrente de pico necessária, para isto então foi considerado o pior caso dos testes e, um capacitor com 4400 uF consegue atender a necessidade de corrente de pico para fazer o solenoide empurrar a ferramenta e fazer o efetuador funcionar corretamente.

A imagem 33 exibe os componentes calculados montados no circuito geral, o capacitor de 4400 uF é composto por uma associação em paralelo de dois capacitores de 2200uF por 16V, os demais componentes foram utilizados de forma normal, sem associações.

Imagem 33-Circuito do efetuador.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

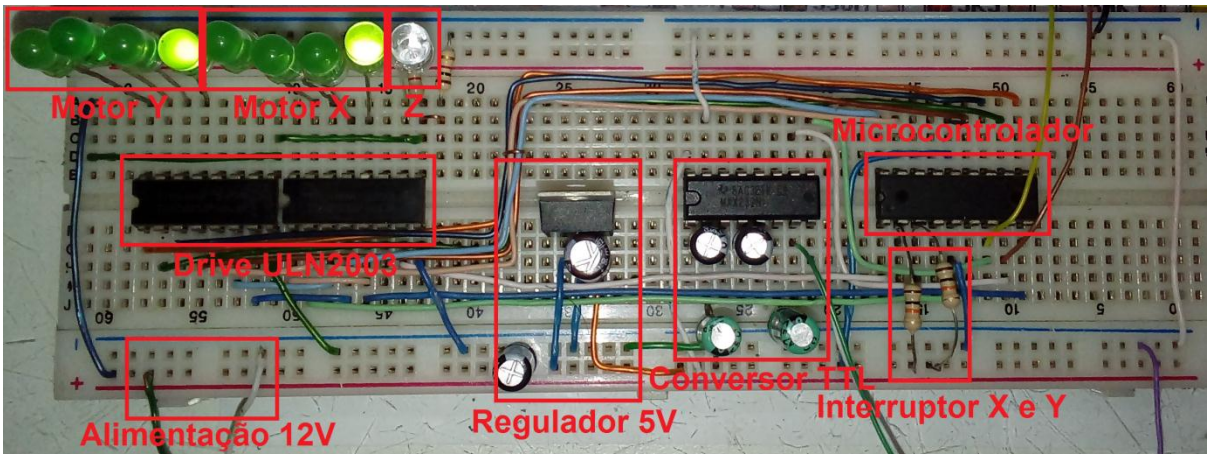
5.4 Eletrônica

A UC é a parte fundamental do funcionamento do robô, com isso a montagem eletrônica foi feita em uma *protoboard*, uma vez que, o uso deste projeto como auxiliar em práticas de laboratório é previsto, sendo assim a montagem em *protoboard* se torna muita mais prática. Ainda que o circuito eletrônico até então tenha sido testado apenas em ambiente de simulação, o circuito eletrônico montado segue o circuito da proposta eletrônica.

Os elementos que compõem o circuito eletrônico foram dispostos na *protoboard* de forma a ser o mais visível os grupos de componentes responsáveis por trabalhos específicos, desta forma a associação visual do circuito em simulador e o circuito real pode ser bem mais nítida quando apresentado à alunos.

A imagem 34 apresenta o circuito já montado na *protoboard* com identificação de cada grupo de componentes envolvidos.

Imagem 34-*Hardware* da UC montado na *protoboard*, fase de testes.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A imagem 34 mostra alguns LEDs - *Light Emiter Diode* nomeados como motores eixo Z, isso foi feito para que o circuito possa ser testado e seu funcionamento observado sem a utilização dos respetivos motores e dispositivos físicos. Através do comportamento dos LEDs durante a execução de um comando por parte da UC é possível observar se o comportamento é o esperado para tal comando. Lembrando que a UC é programada para sempre que iniciada movimentar 100 passos de cada motor e depois inverter o sentido de rotação para deslocar o efetuador em busca do *home*, que corresponde aos dois interruptores pressionados.

Quando a UC operar em condições normais, principalmente de velocidade de motores, o comportamento dos LEDs ainda que seja o esperado não poderá ser observado a olho nu em virtude do tempo muito pequeno entre cada mudança de estado, porém para esta etapa em que a UC trabalha sem os motores é possível configurar a velocidade entre passos de motores para o máximo de 200 ms via comando “v200”, ou no código antes de ser gravado no CI, desta forma, é perfeitamente observável as trocas de estados dos motores. Observar a troca de estados dos motores nesta etapa é importante para garantir a correta montagem do circuito, da forma com que está apresentado na imagem 34 cada led representa um terminal do motor, que em condições corretas de montagem ao se deslocarem devem acender sequentemente um após o outro no sentido informado.

Considerando as informações contidas na seção motor de passo, os LEDs demonstram a matriz de deslocamento correspondente para cada motor, desta forma então, cada LED que se

apaga quando seu vizinho se acende representa a troca entre as bobinas do motor e consequentemente a 1 passo de deslocamento do mesmo.

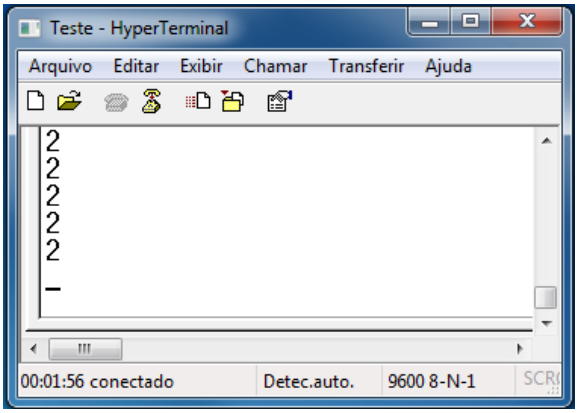
Contudo, apenas testar os motores não basta, um ponto crítico do sistema é a comunicação com o computador, sem essa comunicação o robô não irá receber comando algum do computador. Para testar a comunicação do robô é necessário que se possa observar os dados enviados pelo robô e que também possa enviar dados para o mesmo, considerando que o formato de dados transmitidos é baseado na tabela ASCII, os comandos a serem enviados e recebidos podem ser realizados em um terminal de texto simples.

Ferramentas que permitam a manipulação da porta serial por meio de uma interface de texto são bem comuns e fáceis de encontrar, alguns sistemas operacionais dispõem de tais ferramentas nativamente. Para este teste foi utilizado o *software HyperTerminal*, que é nativo do sistema operacional *Windows* e que permite a manipulação da porta serial por um terminal de texto puro.

Independentemente do *software* utilizado para manipular a porta serial por meio de um terminal, a configuração da porta de comunicação deve ser igual a configuração da porta serial da UC, que inicialmente sua taxa de dados é de sempre 9600 bps, ainda que possa ser alterada via comando, esta taxa de dados mostrada é sempre a inicial, as demais configurações da porta serial são: *Bits* de dados = 8, Paridade = nenhum, *Bits* de parada = 1 e Controle de fluxo = Xon/Xoff. Esta configuração de comunicação, com exceção da taxa de dados, é, segundo o fabricante do microcontrolador, a configuração padrão da porta quando nenhum valor específico é informado.

Após a UC ser ligada e os interruptores de *home* “pressionados”, espera-se observar no terminal da porta serial o caracter “2” aparecer repetidamente informado o estado “pronto” da UC. A imagem 35 mostra a tela do *software HyperTerminal* com o caracter 2 repetidamente recebido, comprovando o funcionamento adequado da UC para o controle de *feedback*.

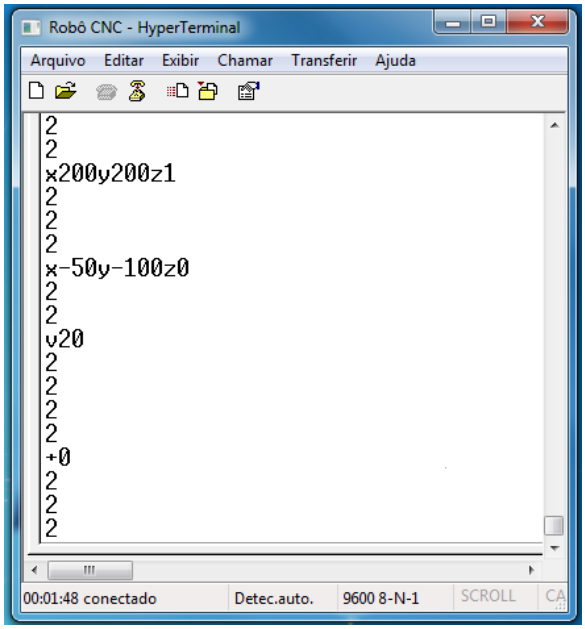
Imagem 35-Recepção do *feedback* no terminal da porta serial.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Com a inicialização da UC já testada, faltou testar apenas o envio de comandos e o comportamento utilizando os LEDs. Para testar o envio de dados alguns comandos foram digitados e enviados diretamente utilizando o *HyperTerminal*, os comandos enviados foram: “x200y200z1” que corresponde a avançar o efetuador nos dois eixos com o solenoide Z acionado, “x-50y-100z0” para retornar o efetuador em 50 passos no eixo X e 100 passos no eixo Y com o efetuador Z desligado, “+0” para retornar o efetuador ao *home* e por fim o comando “v20” para configurar a velocidade dos motores para 20 ms de intervalo entre cada passo. A imagem 36 exibe a tela do *HyperTerminal* após o envio dos comandos, nota-se que o *feedback* é enviado após cada comando, ainda que não se possa observar na imagem, o tempo entre o envio do comando e o retorno do *feedback* foi proporcional ao comando enviado e o comportamento observado pelos LEDs instalados.

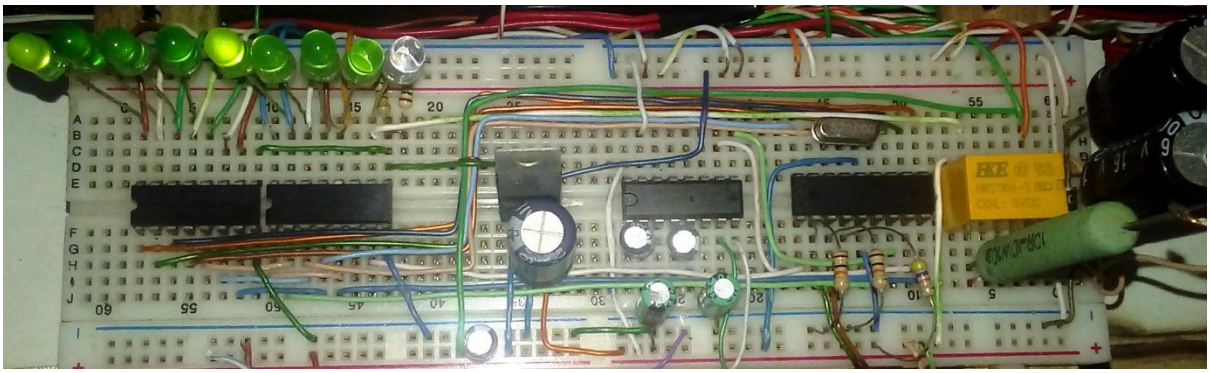
Imagem 36-Envio e recepção de informações no terminal da porta serial.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Com isso foi possível garantir o correto funcionamento da UC e correta montagem do circuito eletrônico completo, a partir deste ponto é possível então fazer a ligação dos motores ao circuito e iniciar as etapas de calibração e coleta de dados. A imagem 37 exibe o circuito completo montado na *protoboard*, o segmento de *hardware* que compunha o controle do solenoide está presente além do cristal gerador de *clock*, utilizado na etapa de coleta de dados.

Imagem 37-Circuito completo.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

5.4.1 Programação do microcontrolador

Somente gerar o arquivo compilado com o código programado para microcontrolador não basta, o microcontrolador deve receber este arquivo. Ao compilar o código escrito em C

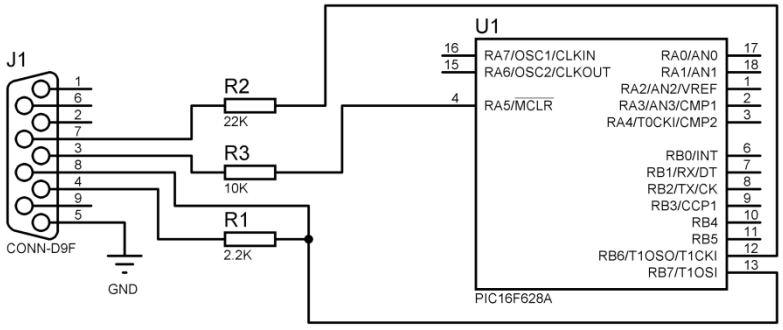
que descreve as rotinas de funcionamento da UC o compilador gera um arquivo no formato .hex, este arquivo contém todas as instruções geradas pelo compilador no formato hexadecimal e este arquivo deve ser gravado no microcontrolador.

Em pesquisas na internet sobre gravadores para o microcontrolador utilizado um *software* citado diversas vezes em fóruns, principalmente voltados para iniciantes em automação foi o WinPic. O WinPic é um *software freeware* que se propõem a gravar diversos microcontroladores da família PIC e entre eles o PIC16F628A.

Para tanto, é preciso um dispositivo que faça a interface entre o PC e o microcontrolador para que o arquivo .hex possa ser transferido (programador). No conteúdo da documentação do *software* WinPic estão listados os programadores suportados e também alguns circuitos eletrônicos para a construção de alguns programadores. Entre os circuitos de programadores disponíveis na documentação um deles chamou a atenção por sua simplicidade, composto apenas por 3 resistores e a utilização de uma porta serial como componentes. Este circuito é chamado pela documentação como *JDM Programmer*. Segundo o fabricante do microcontrolador em seu *datasheet*, do microcontrolador em questão é compatível com programadores do tipo JDM, desta maneira então, este gravador foi utilizado para transferir o arquivo .hex para o microcontrolador, ou seja, programar o CI.

A imagem 38 mostra o circuito eletrônico do programador JDM, o circuito foi modelado no *software* Proteus para que se pudesse gerar a imagem com maior clareza do que o presente na documentação do WinPic.

Imagem 38-Programado JDM

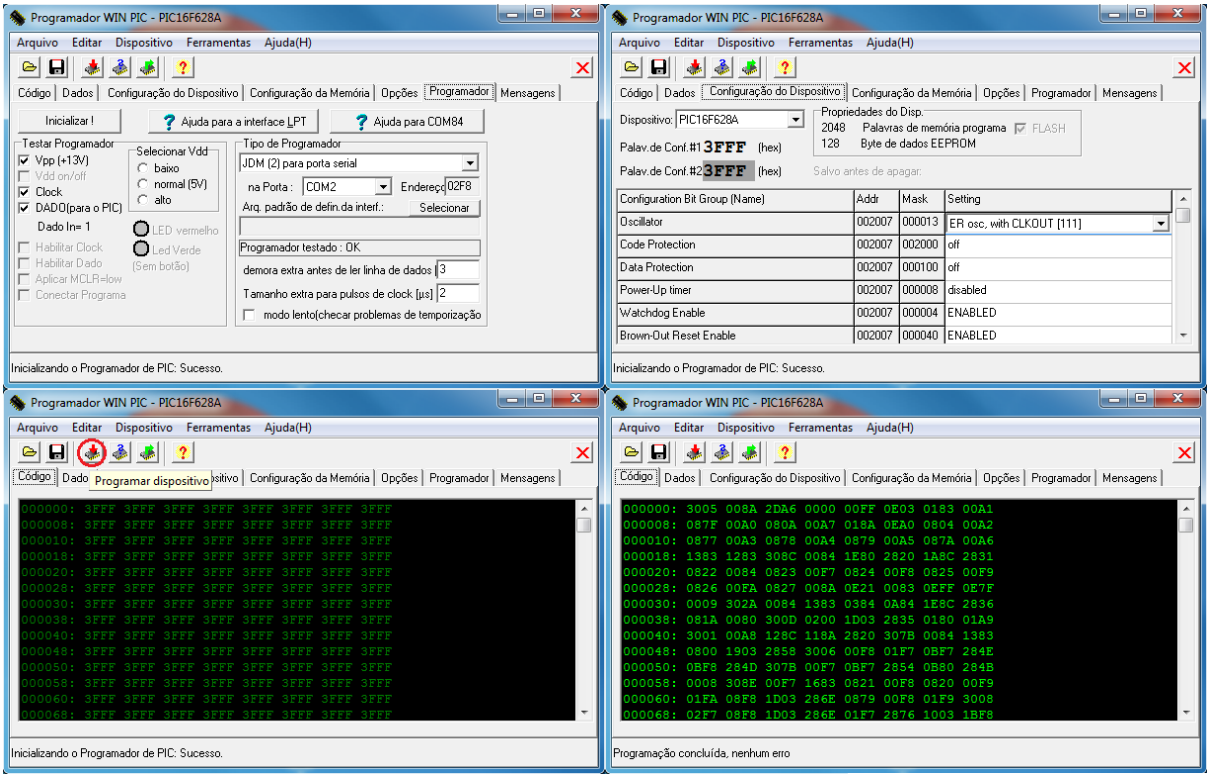


Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O circuito foi montado em uma *protoboard* para possibilitar que o chip possa ser retirado e transferido para o circuito da UC do robô quando necessário recolocado para outras gravações.

A imagem 39 apresenta a aba de configuração do WinPic com os ajustes necessários para a programação utilizando o *JDM Programmer* e a baixo, na metade inferior, a tela apresentada no início e ao final de uma programação do microcontrolador.

Imagem 39-Configurações básicas do *software* programado WIN PIC.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Vale lembrar que este programador não tem recursos como *debugger* e demais ferramentas, ele se propõe somente a transferência do arquivo gerado pelo compilador para o CI, informações estas contidas na documentação do WinPic.

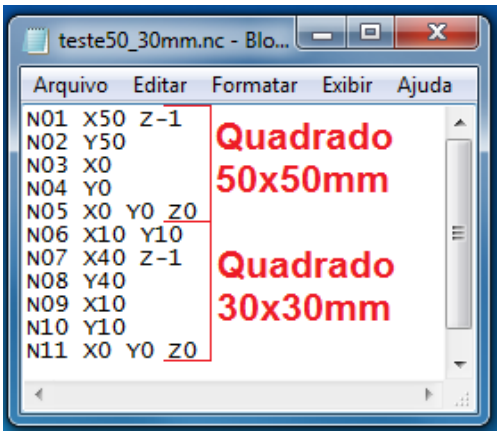
5.5 Testes do sistema

Uma vez que a montagem geral do robô está concluída, a fase de testes gerais é iniciada, estes testes têm a finalidade de buscar por erros e falhas que possam aparecer ao integrar todas as partes, além de ajustar alguns dados como o número de passos por milímetro deslocado, o motor deve dar uma velocidade razoável para os motores, que até então está configurada para intervalos entre passos de 10 ms.

Para o teste geral, um código G foi escrito manualmente com instruções para o robô acionar uma caneta em seu efetuador e desenhar 2 quadrados, um dentro do outro e centralizados com lados iguais a 50 mm e 30 mm.

A configuração inicial para do *software* de comunicação para os testes iniciais é de 46 passos por milímetro e velocidade de 10 ms entre passos. O código G escrito para os testes básicos pode ser visto na imagem 40.

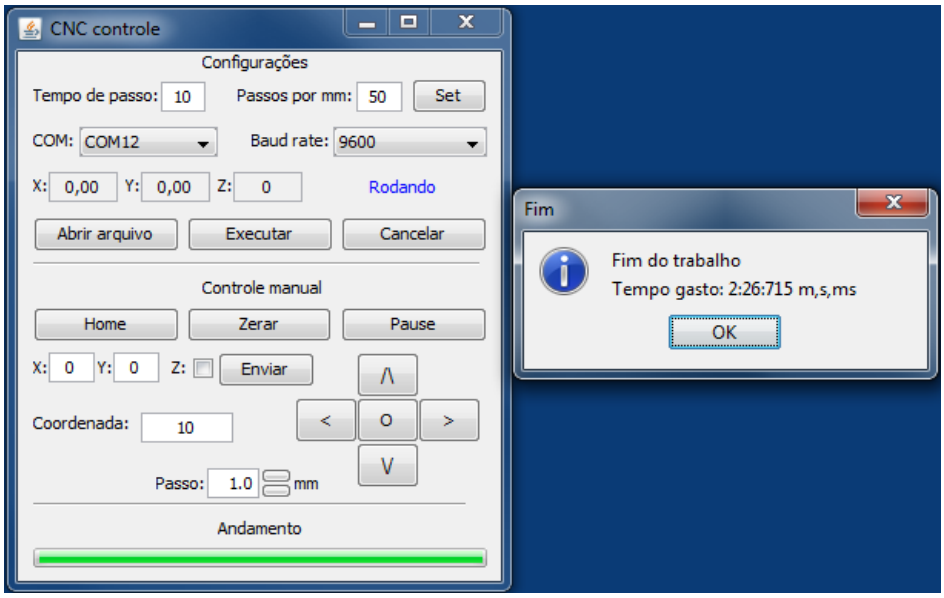
Imagem 40-Teste escrito em código G para realizar a calibração.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O código G foi escrito em um editor de texto puro e salvo no formato .nc para que o *software* de comunicação possa ser executado, conforme foi definido anteriormente. A imagem 41 mostra o resultado do código G executado pelo robô e a tela do *software* de comunicação ao final do processo.

Imagem 41-Trabalho concluído.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

5.5.1 Calibração

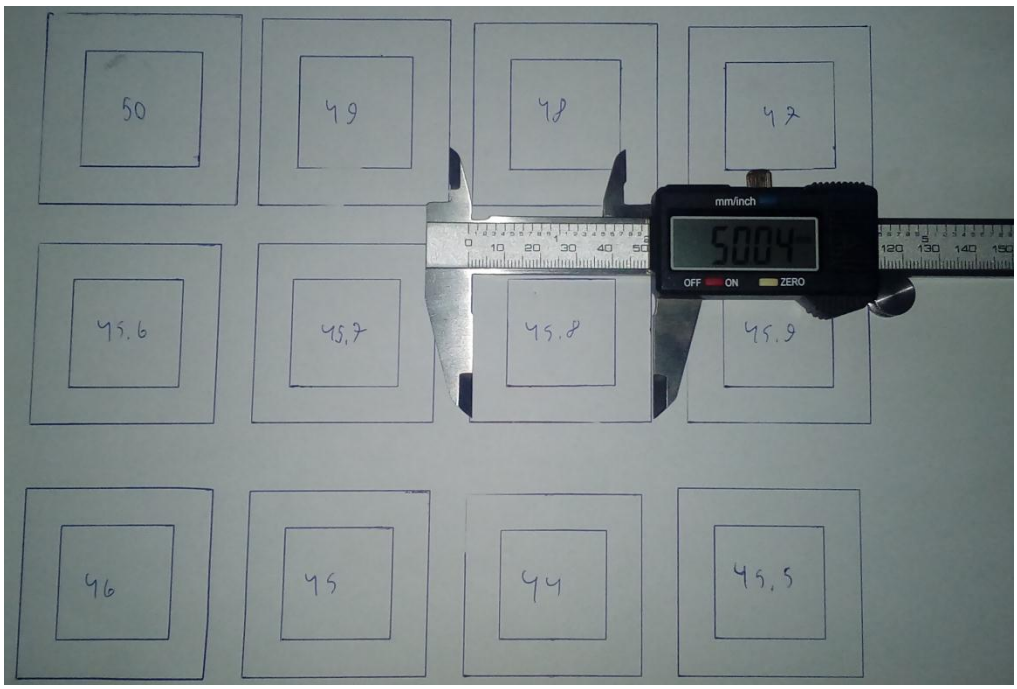
Após a montagem geral do robô e testes do sistema, é necessário calibrar o mesmo, como o código G que por ele será interpretado irá trabalhar com incremental em mm, a escala de trabalho do robô deve ser o mais fiel possível neste quesito. O deslocamento que um comando em código G inferir ao robô deve ser executado pelo mesmo corretamente para garantir a qualidade do trabalho, e para isso, ele deve estar ajustado para se comportar de maneira correta.

Considerando todos os sistemas já funcionando corretamente, é hora de calibrar o robô, deve-se levar em consideração que a maioria dos componentes empregados não são destinados para aplicações de precisão como o robô exige, componentes como a barra roscada utilizada e os motores não têm precisão garantida, então por meio de testes foram definidos valores ideais para o funcionamento do robô.

Inicialmente, as configurações calculadas foram considerando que a barra roscada possuía 1 mm de passo, ou seja, 1 mm de deslocamento por revolução, então se os motores têm 7,5° de ângulo por passo e para completar uma revolução é necessário o acúmulo de 360° o qual serão necessários 48 passos para uma revolução.

Utilizando o código G escrito anteriormente e fazendo alteração no número de passos por revolução, o código foi executado repetidas vezes até que as medidas reais dos desenhos se adequassem às do código, 50 e 30 mm, 43 apresenta as medidas de diversas situações variando entre 50 e 44 passos por milímetro, a configuração que alcançou as medidas corretas para os quadrados foi de 45.8 passos por milímetro, sendo então este valor utilizado como a relação de passos por milímetro para este robô.

Imagem 42-Calibração do robô.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Com o valor adequado encontrado foi possível então encerrar a etapa de calibração e partir para coleta de dados.

6 COLETA DE DADOS

Para que se possa definir alguns parâmetros de configuração e conhecer limites e capacidades do robô é necessário analisar suas capacidades em diversos aspectos. Considerando as etapas do algoritmo geral de funcionamento do robô, aspectos como tempo de processamento após receber as informações do *software* de controle, o tempo de comunicação entre o *software* e a UC e também capacidades mecânicas como velocidade de deslocamento dos eixos.

Para cada tipo de dado coletado foi necessária uma metodologia de captura de dados diferente, portanto, para cada característica avaliada são apresentadas também as metodologias utilizadas na captura dos dados e a validação do método de captura empregado em cada teste.

6.1 Tempos do sistema

A avaliação do tempo envolvido em cada etapa do trabalho realizado pela UC nesta fase do projeto serve para elucidar as implicações de variações de algumas características na configuração e programação da UC.

A taxa de dados utilizada na comunicação entre o PC e a UC interferem no tempo de trabalho geral do robô, assim como sua capacidade de processamento dos dados recebidos, para isso foi preciso então definir um método de aquisição destes dados. Considerando que a comunicação e a saída de comandos da UC são sinais elétricos e que pela estrutura da UC do robô e o formato de dados da comunicação utilizando estes sinais se caracterizam por variáveis binárias que quando dispostas no tempo geram ondas no formato quadrado.

O formato de onda gerado pela UC e recebido por ela consiste nos dados que devem ser analisados, as variações nestas ondas são reflexos de alterações na UC, segundo Newton Braga (2014) para analisar este tipo de corrente e formato de ondas de forma gráfica é necessário a utilização de instrumentos denominados osciloscópios.

Entre as opções de osciloscópios disponíveis para este projeto estão o modelo MO-1221 fabricado pela Minipa e o Soundcard Scope desenvolvido por Christian Zeitnitz. O osciloscópio MO-1221 é um equipamento comercial com taxa de amostragem de 20 MHz, 2 canais analógicos de leitura para pontas de prova, por ser um modelo antigo seus dados são única e exclusivamente exibidos em sua tela de tubo de raios catódicos, este modelo conta ainda com uma saída de sinal de onda quadrada de 1 KHz utilizado para aferições.

O Soundcard Scope é um *software* para Windows que utiliza como entrada de sinal a entrada de som do computador como entradas de pontas de prova, sua vantagem é de seus dados serem exibidos na tela do computador, o que proporciona melhor análise dos sinais capturados, além de poder exportar este sinal em formato compatível com o Excel do pacote Office do Windows,

em contrapartida, sua taxa de amostragem fica limitada à capacidade da placa de som utilizada no computador, que por padrão é de 22 KHz podendo ser alterada até 192 KHz e conversor ADC de 24 bits quando configurada manualmente.

A fim de validar os equipamentos foi realizado um teste de leitura utilizando a saída de sinal de 1KHz presente no osciloscópio MO-1221. O teste baseia-se em fazer a leitura deste sinal com os equipamentos disponíveis e comparar os dados adquiridos, a imagem 43 mostra o comparativo dos equipamentos durante o teste.

Imagem 43-Medições de um mesmo sinal.

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Com o ajuste do "botão" *timer* em ambos os osciloscópios utilizados foi definido que cada divisão da tela, blocos quadrados distribuídos na tela do osciloscópio, corresponda a 500 ms, desta forma, a análise visual é possível.

Com base no teste realizado é possível perceber a precisão do *software* Soundcard Scope, apesar de sua taxa de varredura ser baixa. A ferramenta mostra-se eficiente para baixas frequências. Vale ressaltar também que no modelo MO-1221 não é possível fixar a imagem exibida em sua tela, só é possível visualizar o sinal enquanto o mesmo está sendo amostrado, o que limita sua utilização, pois os dados transmitidos na porta serial serão feitos em rajadas, o que dificultaria a análise do sinal.

Neste caso então, foi optado por utilizar o *software* Soundcard Scope para fazer medições até o limite de varredura que o teorema de Nyquist recomenda, Nyquist diz que a taxa de amostragem deve ser de pelo menos o dobro da máxima frequência que o sinal amostrado pode alcançar.

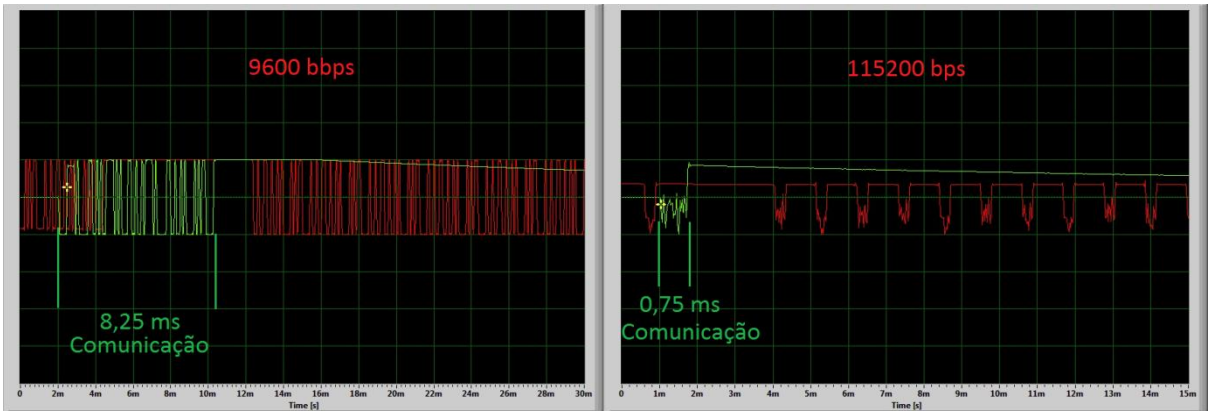
6.1.2 Atrasos de comunicação

A maneira com que o robô se comunica com o meio externo também é um limitante do desempenho da mesma, considerando a estrutura de funcionamento em que cada instrução para o robô é transmitida individualmente, o tempo levado para cada transmissão é acumulado ao longo do trabalho do robô. Para tanto, conhecer a capacidade real de comunicação e seu comportamento foi possível nesta etapa de coleta de informações.

Com o auxílio do osciloscópio foram realizadas medições de tempo gasto nas comunicações, vale ressaltar que as taxas de comunicação para o circuito projetado podem suportar até 115200 bps, o que representa algo na casa de 115 KHz na frequência a ser amostrada para visualizar a comunicação, contudo o osciloscópio utilizado suporta no máximo 192 KHz e como Nyquist afirma, a taxa de amostragem deve ser o dobro do sinal amostrado, contudo, deve-se considerar que o que está sendo analisado é o tempo da comunicação e não a comunicação propriamente dita. Não é a ideia deste teste coletar dados sobre os dados transmitidos e sim o tempo empregado na rajada de bits enviadas por ambos dispositivos, deste modo, visualizar o início e o final da transmissão de dados satisfazem o caso de uso.

A medição foi realizada repetitivamente em cada configuração de capacidade de comunicação, e com a variação do *clock* utilizado no microcontrolador. Os dados foram coletados pela imagem da rajada de dados no barramento de comunicação gerada pelo osciloscópio. A imagem 44 ilustra a diferença no comportamento do sinal amostrado quando configurado para a mínima taxa de dados e máxima taxa de dados.

Imagem 44-Tempo gasto na comunicação.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Como mencionado, a limitação da taxa de amostragem do osciloscópio utilizado limita a visualização da rajada de dados quando operando em 115200 bps, porém é perfeitamente nítido o início e final da transmissão, que de fato é aspecto relevante desta coleta.

A tabela 8 apresenta os valores de tempo gasto para transmissão do comando de 1 passo de motor no eixo Y, que corresponde a "x0y1z0\n\r", em suas respectivas taxas de dados.

Tabela 8-Tempos de comunicação observados.

Taxa de dados (bps)	**Tempo gasto em 20 MHz**	**Tempo gasto em 4 MHz**
9600	8.25 ms	8,25
19200	4,25 ms	4,25
57600	1,6 ms	Taxa não suportada
115200	0,75 ms	Taxa não suportada

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Observa-se que a variação de *clock* não interfere no tempo gasto na comunicação, porém é um limitante para taxas de transmissão mais altas, as taxas de 57600 e 115200 só podem ser configuradas para uso quando o microcontrolador opera em 20 MHz. Este limite é informado pelo compilador no.

6.1.3 Atrasos de processamento

Sempre que uma nova ação é recebida pela UC uma sequência de etapas de processamento são realizadas para que os atuadores possam ser acionados e os movimentos do robô realizados, estas etapas de processamento requerem certo tempo para serem executadas e é acumulativo ao longo de todo o processo. Conhecer este tempo pode ajudar no dimensionamento correto de partes do sistema como o *clock* de trabalho, algumas estruturas de código de processamento entre outros, para visualizar estes efeitos sobre o robô foi criada uma metodologia de testes baseada em diferenças de *clock* para o microcontrolador e o uso de instruções específicas que propõem melhorar o acesso às portas IO do microcontrolador.

O teste é baseado na comparação entre o tempo de processamento de duas configurações de *hardware*, variação de *clock*, e duas configurações de *software*, instruções para acesso rápido à IOs. Neste teste também foi utilizado o osciloscópio para a captura de dados, o osciloscópio foi ligado de modo a visualizar os dados sendo recebidos pela UC, conectando uma das pontas

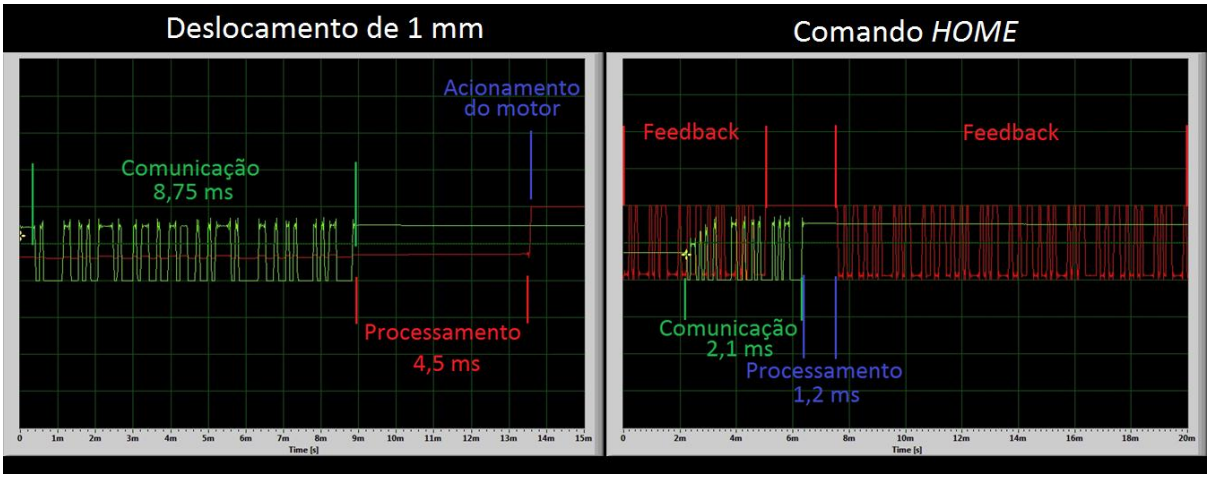
de prova diretamente ao terminal RX do microcontrolador, e a outra ponta de prova diretamente ao terminal da UC responsável pela bobina do motor que será acionada logo após o final do processamento, desta forma, é possível visualizar e medir o tempo gasto pela UC entre o final do recebimento de dados e o início do acionamento dos motores.

A variação na configuração de *hardware* definida para este teste teve como base a possibilidade do microcontrolador escolhido PIC16F628A ter a capacidade de operar com base em seu oscilador interno ou com um oscilador externo. Quando configurado para operar com seu oscilador interno o seu *clock* de trabalho máximo é de 4 MHz, e quando configurado para operar com oscilador externo é necessário então a adição de um cristal ao circuito para gerar o *clock*, neste modo de operação o cristal de *clock* máximo suportado pelo microcontrolador segundo o fabricante é de 20MHz. Com isso a coleta de dados foi realizada utilizando o mesmo comando para ambas configurações, desta maneira, evitando alterações por informações diferentes, os comandos foram enviados para a UC utilizando o *software* de comunicação desenvolvido para o robô.

Os comandos utilizados neste teste são: deslocamento manual 1 passo no motor do eixo Y em diversas velocidades, chamada da função *HOME*, acionamento do efetuador e o comando *set* para passagem de parâmetros para o robô.

A imagem 45 exibe a forma com que os dados foram analisados na tela do osciloscópio, os valores adquiridos neste teste estão disponíveis na tabela 9.

Imagem 45-Dados contidos nas análises.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A tabela 9 apresenta os resultados de tempos de atraso em função de cada critério citado.

Tabela 9-Tempo de processamentos encontrados.

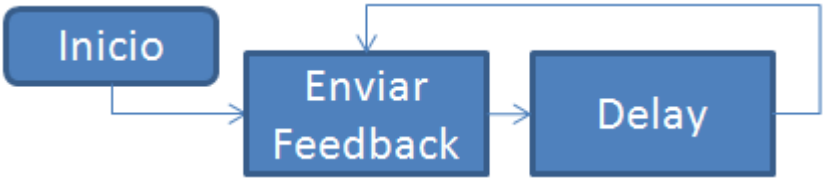
Rate 20 MHz	**Proc. 1 passo**	**Proc. home**	**Proc. Z**	**Proc. Set**
9600	1 ms	0,5 ms	0,65 ms	0,1 ms
19200	1 ms	0,75 ms	0,65 ms	1 ms
57600	1 ms	0,6 ms	0,7 ms	0,75 ms
115200	1 ms	0,2 ms	0,6 ms	0,5 ms
Rate 4 MHz	-	-	-	-
9600	3,75 ms	1,2	2 ms	0,4 ms
19200	3,75 ms	0,9 ms	2 ms	2,2 ms

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Um comportamento observado nesta etapa é a variação no tempo de *set* de informações, e os tempos encontrados variam entre 1 ms e 0,1 ms. O processamento do comando "*set*" consiste no processamento das informações recebidas pelo microcontrolador a atribuição do valor a uma variável, esta inconsistência de tempo entre os testes não é decorrida do processo e sim do sistema de *feedback*.

O fluxograma da imagem 46 exibe a estrutura básica do sistema de *feedback* utilizado na UC.

Imagem 46-Fluxograma de *feedback*.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O comportamento observado pode ser explicado ao analisar a etapa *delay* do fluxograma, uma vez que, só uma interrupção gerada pela comunicação pode interromper o *looping* apresentado, tal interrupção pode ocorrer durante o *delay*, sendo assim, a interrupção altera uma *flag* que faz o sistema sair do *looping*, porém a *flag* somente será lida após o término do *delay*, mesmo que a interrupção tenha ocorrido durante o mesmo. Portanto, se uma transmissão ocorrer durante o *delay* de 1ms, considerando um caso hipotético, aos 0,2ms, a *flag* que força a saída do *looping* é configurada de forma a permitir a saída, porém, após o tratamento

de interrupção alterar a variável de *flag* ainda restam 0,8 ms de tempo a ser esperado para a variável seja lida e o *looping* interrompido.

Uma solução encontrada foi a divisão do *delay* estático em um sistema de *delay* interrompível, basicamente isto é feito alterando o tempo do *delay* na programação de "*delay_ms(1)*" para "*delay*_us(1)" e colocando dentro de um laço de repetição "*for(flag=-2000;flag<0;flag++)*", desta forma, o *delay* inferido pela repetição de *delays* menores pode ser interrompido variável *flag* for ajustada para 0. Desta forma, não ocorre mais o tempo de término do *delay* para que a repetição de *feedback* seja interrompida e desta maneira corrigindo o comportamento observado.

O resultado da alteração refletiu em tempo de 0,1 ms para o comando *SET* de forma consistentes, se mantendo o mesmo em todas as condições, o que se esperava quando observado os demais resultados, com a técnica utilizada os tempos sofrem menos alterações entre as medições, sendo assim, podendo chegar mais próximo do valor real de tempo sem interferências significativas do *delay*. Outro ganho importante de tal alteração é poder utilizar um tempo maior entre cara reenvio do *feedback*, pois o tempo de espera entre os mesmos não é mais um problema, pois, utilizar um tempo maior diminui o número de dados transmitidos nesta etapa.

Outro ponto avaliado foi a utilização do comando "*#use fast_io(port)*", segundo Pereira (2011) este comando serve para informar o compilador que as portas do microcontrolador devem seguir a configuração dada pelo programa para entrada e saída, assim não utilizando mais o sistema de detecção automática de uso das portas. Quando utilizado este comando é necessário informar o comportamento de cada porta do microcontrolador, quais devem ser saída e quais devem ser entradas, para isso o comando "*set_tris_a(0b00011000)*" é utilizado, ele informa que cada porta, bit a bit, da esquerda para a direita, deve ser "0" para saída de dados e "1" para entrada de dados.

A diferença percebida ao analisar o microcontrolador programado com o comando "*#use fast_io(port*)" e sem o comando foi apenas redução de tamanho de código, número de instruções a serem gravadas no microcontrolador, não foi observado impacto nos tempos analisados. A Diferença de número de instruções, segundo mensagem apresentada após a compilação foi de 2%, o que apesar de não implicar em alteração de desempenho do microcontrolador, é muito importante considerando a baixa quantia de memória para programa disponível no microcontrolador PIC16f628A.

Com estes dados foi possível definir a frequência de trabalho do microcontrolador em 20MHz como frequência de operação padrão para o robô, quanto à taxa de dados, será mantido

o valor de 9600 bps como configuração inicial da UC, essa taxa pode ser alterada para as demais taxas suportadas em um campo na interface do *software* de comunicação.

6.2 Velocidade de deslocamento

O sistema de controle desenvolvido para o robô apresentado permite que a velocidade de rotação dos motores seja ajustada configurando o tempo entre cada passo de motor em grandezas de milissegundos, contudo, a alteração deste valor implica em maior ou menor rotação dos motores e consequentemente maior ou menor velocidade de deslocamento das juntas do robô. Alguns fatores influenciam nesta velocidade não ficando restrito somente ao intervalo entre os passos dos motores, mas também ao grau de deslocamento por passo de motor e ao passo de deslocamento da barra roscada utilizada, ou outra forma de transmissão de deslocamento.

Como o projeto se propõe a poder ser utilizado como ferramenta didática para aulas em laboratório, foi desenvolvido um método simples de medir essa velocidade de deslocamento em função da configuração do tempo entre passos de motor. A definição de velocidade diz que velocidade é o tempo decorrido para se deslocar um determinado espaço, desta forma, o método consiste de medir o tempo de deslocamento das juntas do motor em um determinado espaço, a imagem 47 exibe o método utilizado.

Imagem 47-Espaço delimitado para mensuração de velocidade.

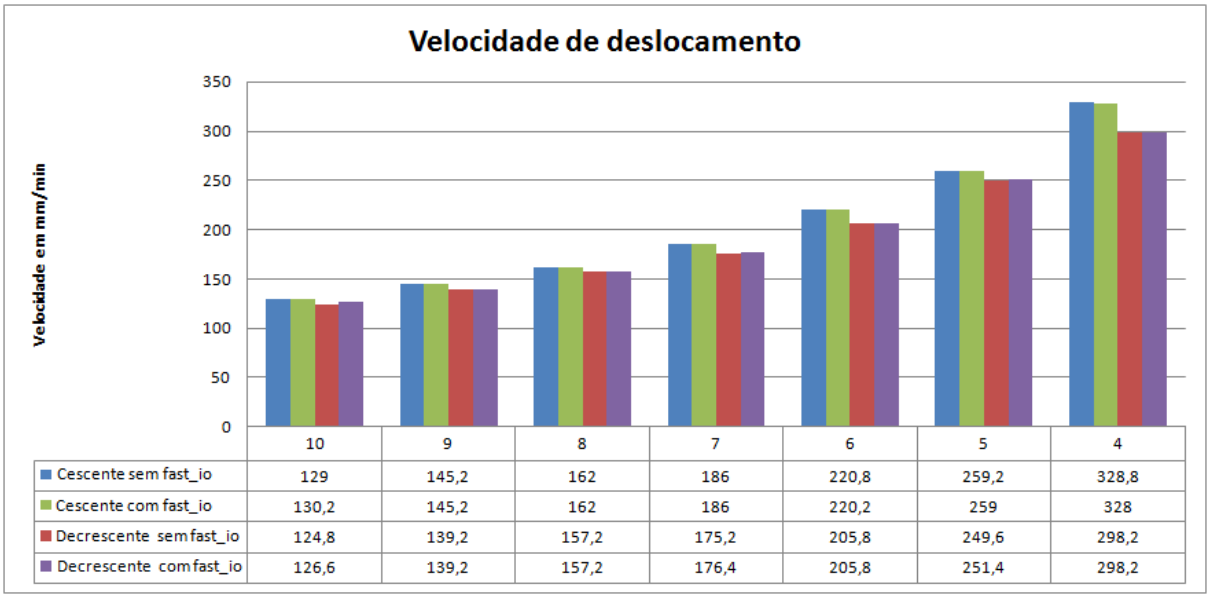
Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Para a mensuração do tempo decorrido para a junta do robô percorrer o espaço delimitado foi utilizado um celular com capacidade de registrar a imagem em formato de vídeo, assim, por meio de qualquer editor de vídeo, até mesmo do próprio celular, é possível identificar o tempo de início do movimento e o tempo final, momentos em que a junta alcança os espaços delimitados. Para este teste os espaços delimitados correspondem a 100 mm de deslocamento

crescente, sentido que o efetuador se afasta do *home*, e novamente 100 mm de deslocamento no sentido decrescente, retornando ao *home*. Como o algoritmo que gerencia o comando *home* se baseia em leitura dos interruptores para manter o movimento dos motores, o mesmo pode obter velocidades menores de deslocamento em função de diferença em números de instruções a serem processadas, desta forma, com os valores de velocidades mensurados é possível expressar esta diferença em razão de velocidade.

Fazer a medição desta forma torna possível ser executada em qualquer laboratório ou sala de aula sem a necessidade de instrumentos específicos para tal. O gráfico 1 apresenta os valores de velocidade de deslocamento encontrados para cada configuração de tempo de passos de motor.

Gráfico 1-Velocidades de deslocamento encontradas.



	10	9	8	7	6	5	4
Cescente sem fast_io	129	145,2	162	186	220,8	259,2	328,8
Cescente com fast_io	130,2	145,2	162	186	220,2	259	328
Decrescente sem fast_io	124,8	139,2	157,2	175,2	205,8	249,6	298,2
Decrescente com fast_io	126,6	139,2	157,2	176,4	205,8	251,4	298,2

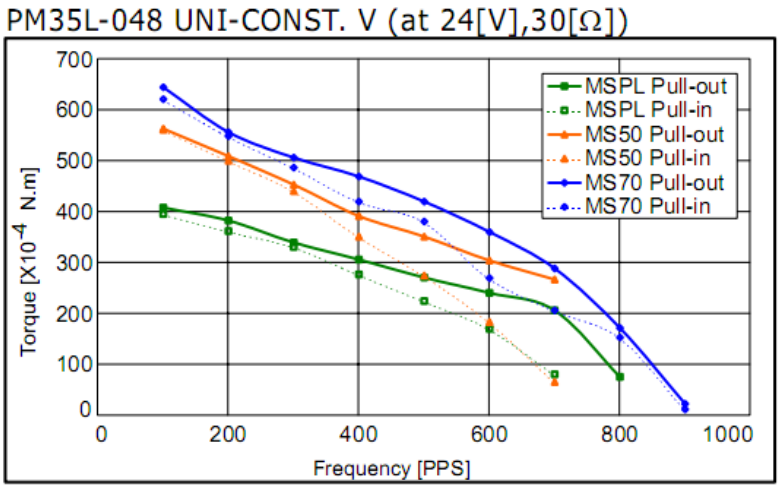
Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Ainda que já havia sido constatado que o comando "*#use fast_io(port)*"não implicou em alterações de tempo de processamento, este teste foi realizado duas vezes, uma com o comando e outra sem, e nota-se pelo gráfico que não houve diferenças nas velocidades encontradas, portanto, foi possível concluir que se trata de otimização para configuração das portas e não para operação das mesmas.

Ao mensurar a velocidades de deslocamento do robô em razão de tempos de passos de motores foi observado um limite para o tempo de passo, foi observado que quando configurado para intervalo de passo de motor em 3 ms os motores não são capazes de mover as juntas do motor em todo o espaço delimitado, seguindo a curva de torque dos motores, fornecido pelo

fabricante, é possível entender que a perda de torque em função da velocidade faz com que o motor não consiga vencer as resistências mecânicas do robô e promover o movimento. Então isso caracteriza uma limitação para a velocidade máxima configurável no robô em 4 ms ou 328 mm/s.

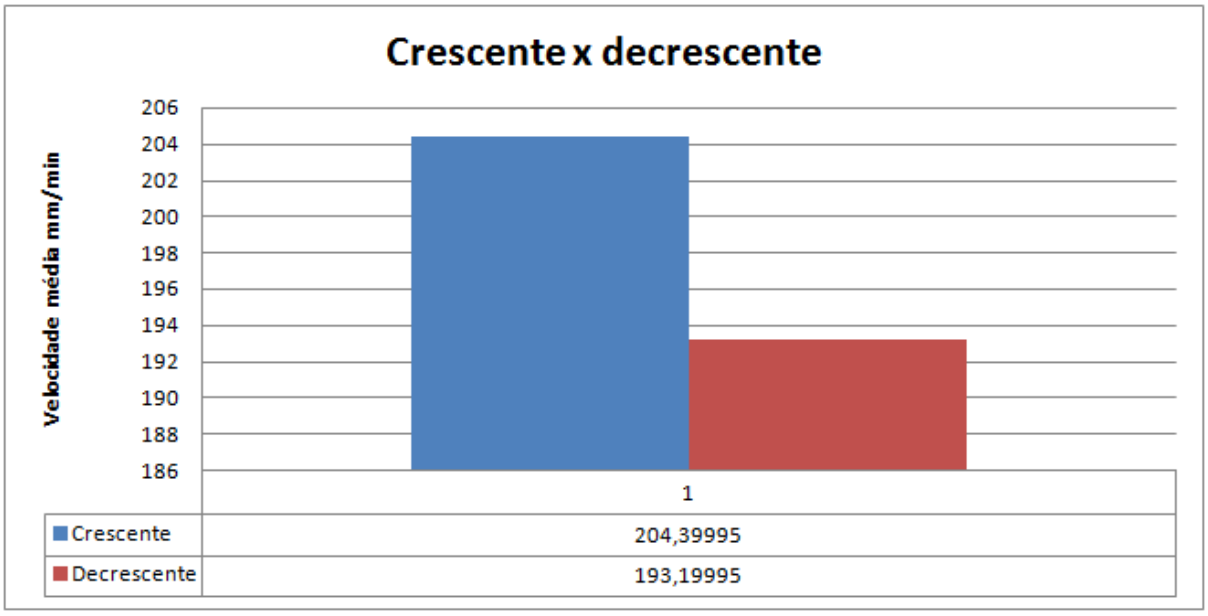
Gráfico 2-Torque do motor em função de tempo de passos.



Fonte: Adaptado do *datasheet* PM35l-48.

O Gráfico 2 apresenta os níveis de torque do motor em função da velocidade de aplicada, o motor utilizado é do tipo MSPL, característica do núcleo magnético do motor, e o sistema de movimentação utilizado pode ser considerado do tipo *pull-in*, considerando um tempo de passos de motor de 4 ms, temos: 1/0,004 = 250 Hz ou no caso do motor, PPS (*Passes Per Second*). Comparando com o gráfico 2, é possível observar que o torque de aproximadamente 320x10^-4 N.m. Podemos atribuir então que o robô desenvolvido necessita de no mínimo 320x10^-4 N.m de torque nos motores para vencer as cargas geradas por atrito e outras limitações mecânicas e trabalhar de forma correta.

Gráfico 3-Diferença entre velocidade crescente e decrescente.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O gráfico 3 expressa a diferença entre as médias de velocidades encontradas para o deslocamento do efetuador no sentido crescente e no sentido decrescente, foi observado uma diferença de 11,2 mm/min de velocidade. Isso se dá, como já descrito anteriormente, pela maneira como o comando *home* é implementado, os *switchs* que informam o *home* para a UC são lidos a cada passo de motor e o estado deles indica se o motor deve dar um novo passo ou não, desta forma, a cada passo dado pelos motores toda a lógica de deslocamento deve ser processada e adicionalmente a leitura dos *switchs* é processada também, esse acréscimo de instruções a serem processadas gera essa diferença de velocidades.

6.3 Operação do robô

O trabalho realizado pelo robô é o acúmulo de diversos comandos enviados sequencialmente, como cada etapa para a execução de um comando pelo robô consome uma porção de tempo já estudado anteriormente, é importante conhecer o impacto disto no tempo total de trabalho do robô. Para isto foi definido um teste padrão que foi realizado em diversos cenários de velocidades de motor e taxas de comunicação de dados.

Outro ponto importante a ser considerado é sobre falhas, como o sistema de atuadores se baseia somente em passos do motor para conhecer a posição do efetuador, a possível perda de passos que pode ocorrer nos motores é um problema a ser considerado. A comunicação também é um ponto de possível falha, informações perdidas podem provocar distorções no trabalho final do robô. Para isto, foi montado um sistema capaz de contar os passos perdidos

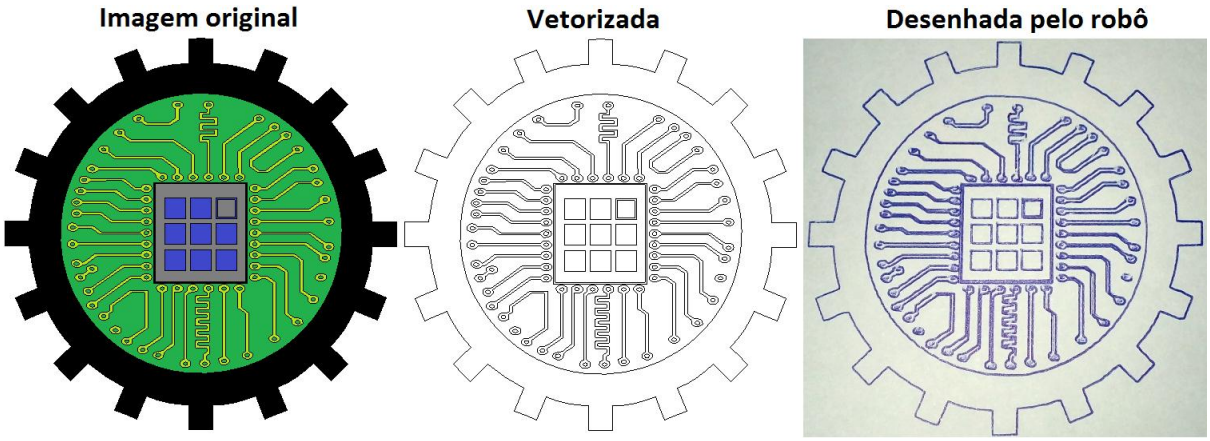
pelos motores, já a comunicação quando a falha ocorrer com comandos que realizam maior trabalho, será possível visualizar em deformações no trabalho final, mas de qualquer forma, irá implicar em perda de passos por falta de alguma instrução.

6.3.1 Teste padrão

Para que os testes sejam válidos é necessário que o robô execute sempre a mesma rotina de código G para que apenas as variações alteradas em cada teste afetem o resultado, por este motivo, foi definido um teste padrão, um código G gerado a partir de uma imagem em formato .BMP.

A imagem 48 exibe a imagem original utilizada para gerar o código G de testes, para gerar o arquivo .nc contendo a programação do robô utilizando o *software* ArtCAM JewelSmith, *software* este capaz de editar e desenvolver trabalhos em 2D e 3D com sistemas vetoriais, desta forma, a imagem foi importada para o mesmo e uma imagem vetorial foi traçada sobre as margens da imagem e dimensionado para o tamanho de 60 mm de altura e 60 mm de largura, a imagem vetorial resultante pode ser vista na imagem 48, após vetorizada, o *software* foi utilizado para gerar um percurso de perfilagem ao longo dos vetores e então salvar este percurso no formato .nc contendo a programação em código G, este arquivo então foi importado no *software* de comunicação do robô e executado, o resultado também pode ser visto na imagem 49.

Imagem 48-Teste padrão.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Neste cenário, todos os testes dinâmicos que envolvem comparação de tempo ou precisão de execução de trabalho pelo robô, serão realizados utilizando este mesmo arquivo .nc com a programação em código G gerada pelo *software* ArtCam e que contém 1930 linhas de instruções de translação para o robô.

6.3.2 Acúmulo de erros

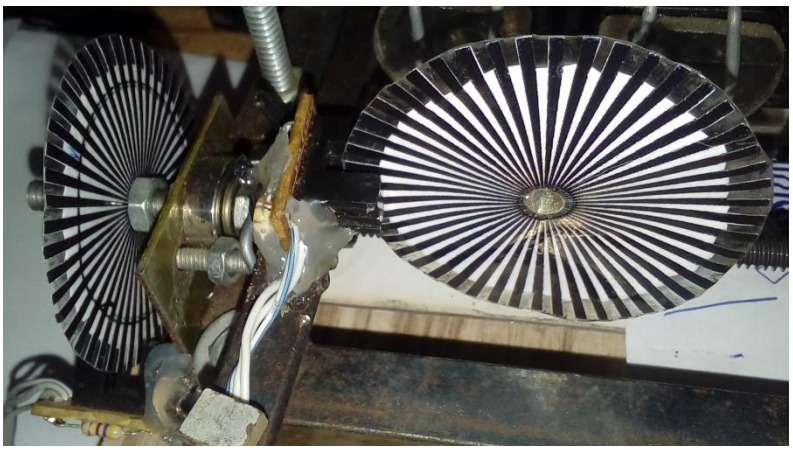
Uma das características do robô construído é não possuir *encoders*, assim há uma maior probabilidade de falhas de precisão provocadas por perda de passos nos motores, com o uso de *encoders*, sempre que um motor efetua um passo de deslocamento o *encoder* "informa" a UC que o passo ocorreu, caso um passo não seja detectado a UC é responsável por repetir este passo perdido, então sem o *encode*r esses possíveis passos perdidos se acumulam durante todo o trabalho realizado pelo robô até que a função *home* seja chamada.

O acúmulo de passos perdidos ao longo do trabalho do robô provoca deformações no traçado à medida que esse número aumenta. Segundo Athani (2005) o aumento de velocidade com que cada motor de passo gira, entende-se menor tempo entre passos, provoca diminuição do torque gerado pelo mesmo. Com isso e associando com as restrições mecânicas que podem gerar forças opostas ao deslocamento, o que demanda mais torque do motor para vencê-las, o resultado pode ser maior número de passos perdidos.

Para mensurar o número de passos perdidos ao realizar o teste padrão definido anteriormente foi construído um sistema de *encoders* gerenciado por um Arduino com a finalidade de contar os passos do motor e exibir este valor no computador. Rosário (2005) esclarece que *encoders* são sensores digitais utilizados normalmente para fornecer realimentação de posição de atuadores, são normalmente compostos por um disco de plástico ou vidro que se move entre uma fonte de luz (LED) e um foto receptor, o disco por sua vez é codificado com setores de transparência e opacidade, desta forma gerando pulsos de luz e escuridão.

Um disco *encoder* com 48 pontos que equivalem ou número de passos necessários para completar uma volta, 360° com os motores de 7,5° por passo, foi confeccionado, impressão por uma impressora laser em uma lâmina plástica e fixado na extremidade de cada barra roscada e um conjunto óptico foi acoplado ao *encoder* para gerar interrupções no Arduino que, por sua vez, irá contar estas interrupções.

Imagem 49-Encoders fixados nos eixos X e Y.



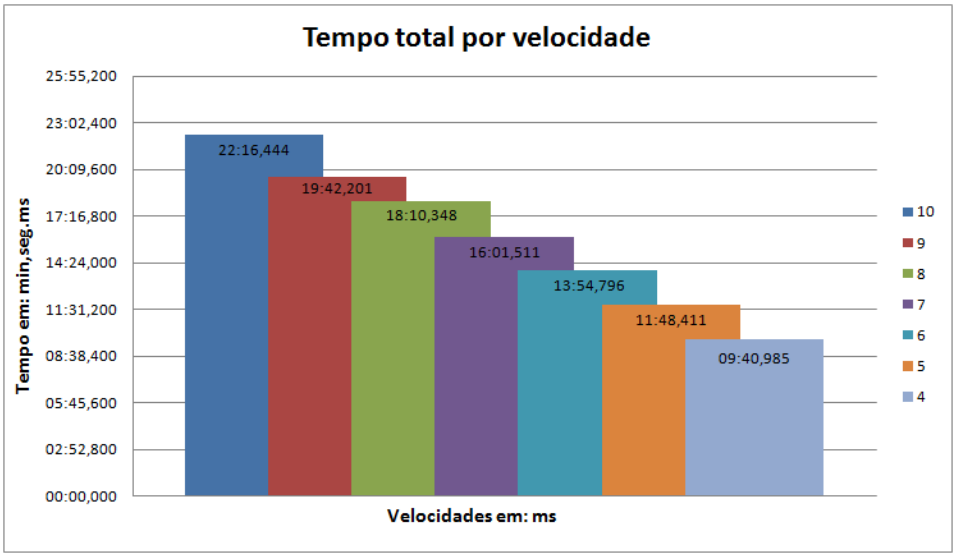
Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O teste consiste em deslocar o efetuador 46 passos, equivalente a 1mm, em cada eixo e zerar a posição do efetuador, criar um falso *home*, após, realizar o teste padrão e ao final executar o comando *home,* o deslocamento que ocorre entre o fim do trabalho e o *home* deve corresponder aos 46 passos, 1 mm deslocado inicialmente, no entanto os possíveis erros podem alterar este valor, então o número de passos perdidos é contabilizado com a subtração do número de passos contados dos 46 passos iniciais, o resultado em módulo decorrente desta subtração corresponde ao acúmulo de passos perdidos. Vale ressaltar que não somente passos perdidos podem gerar este deslocamento, mas também possíveis folgas mecânicas e principalmente arredondamento de valores na etapa de transformações geométricas e de rasterização, porém esse acúmulo de erros independente da natureza de sua origem será expresso em deslocamento, e sendo assim pode ser mensurado em forma de passos de motor.

Foram realizados 3 testes (triplicata) em cada configuração, estas utilizadas correspondem a velocidade de passo de motor entre 10ms e 4ms e a variação da taxa de dados 9600, 19200, 57600 e 115200 bps em cada configuração de velocidade, de maneira geral, foram realizados 84 testes. Os valores de tempo de execução considerados para cada configuração é a média aritmética da triplicata correspondente. Dados de acúmulo de erros são tratados estatisticamente com o todo de testes, não se baseando em média aritmética, dessa forma, são considerados 84 amostras de erros.

O gráfico 4 exibe a variação de tempo total de trabalho em função de cada velocidade considerando a mesma taxa de dados, neste caso, 9600 bps, que é a configuração padrão do robô.

Gráfico 4- Tempo total de velocidade.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

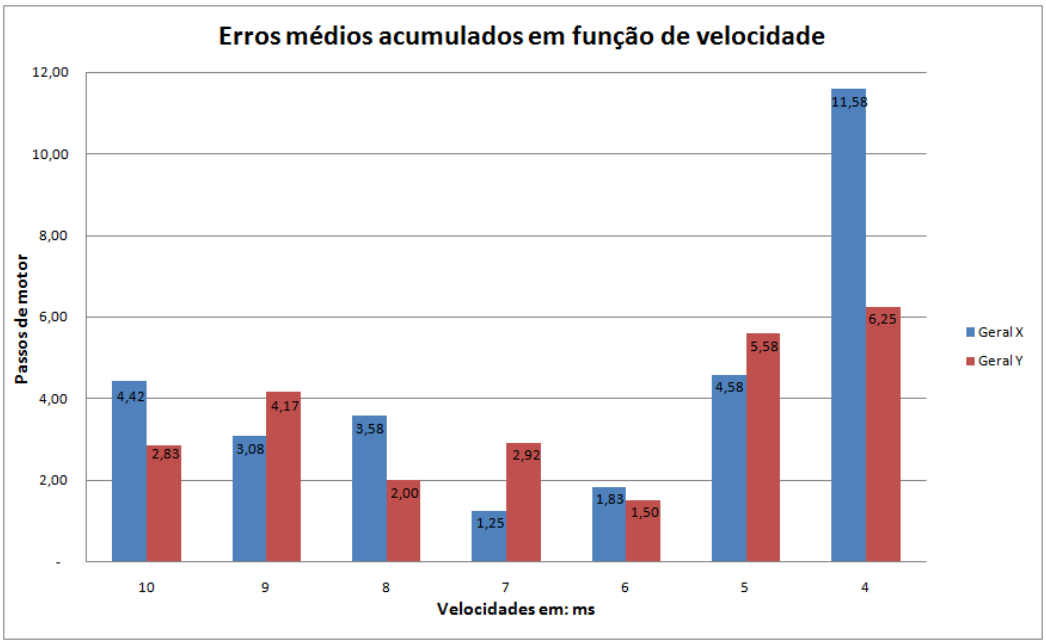
Observando o comportamento ilustrado no gráfico 4 é possível perceber uma variação "constante" na forma com que o tempo total diminui em função da velocidade dos motores, essa variação fica compreendida entre os tempos de 2:07,427 e 2:00,534 para cada taxa de dados testada.

Analisando o número de passos perdidos, considerando as 85 amostras, os valores de mediana (maior ocorrência dentre as amostras) encontrados para X foram de 8 passo e para Y 17 passos. Como esses valores são muito recorrentes dentre as amostras e as demais amostras não se desviam muito deste valor, é possível atribuir esse número de passos de erros à arredondamentos matemáticos uma vez que são recorrentes e presentes em todas as amostras. Arredondamentos de valores ocorrem em dois momentos do processamento do Robô, um no programa de comunicação quando faz os cálculos de translação e converter para número de passos de motores, estes em inteiros, e outro na fase de rasterização na UC do robô, segundo Lopes (2004) o algoritmo incremental básico utilizado para rasterização apresenta um problema de acúmulo de erros em virtude ao limite de algarismos significativos que um valor real é representado na memória, e assim a realização de sucessivas adições faz o aumentar o erro durante o cálculo.

Considerando então como erro de arredondamentos matemáticos os valores de 8 passos para X e 17 passos para Y, todos os valores que se desviaram disto foram tratados como perda de passos de motor por outras razões como atrito, por exemplo. O gráfico 5 apresenta um comparativo quanto a número de passos perdidos considerando a média aritmética das 3

amostras de cada configuração de velocidade, números estes já subtraídos do número de passos perdidos por arredondamentos matemáticos.

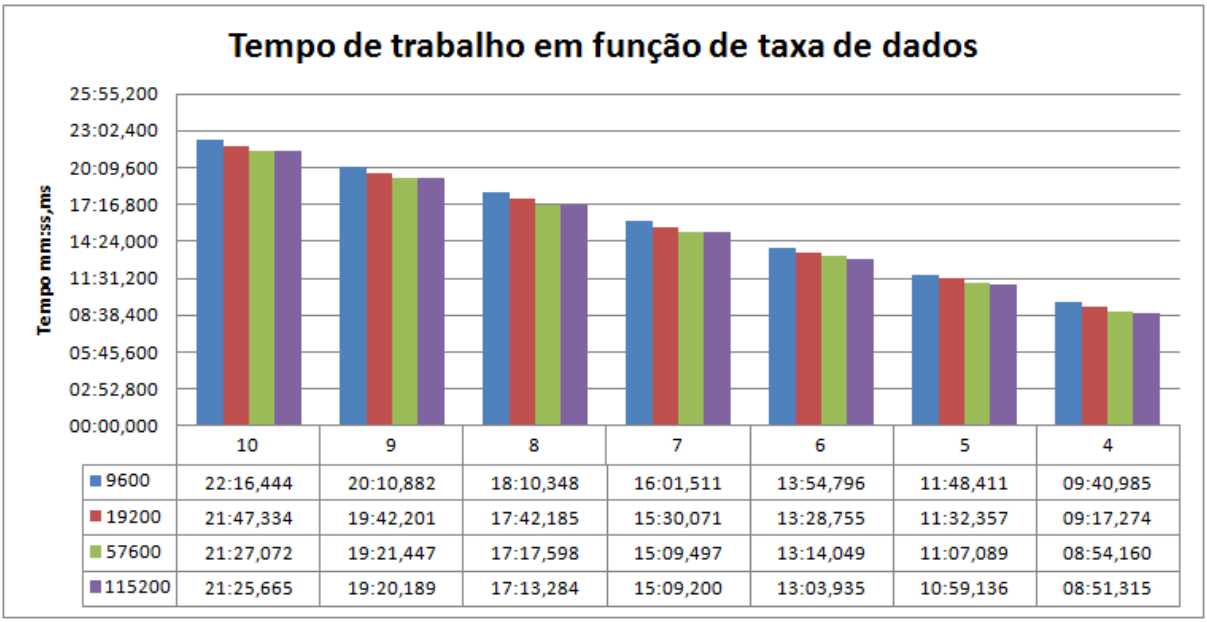
Gráfico 5-Erros em função de velocidade.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O tempo acumulado ao longo de cada transmissão de dados para o robô também contribui para o tempo geral de execução de um trabalho pelo mesmo, o gráfico 6 apresenta o tempo médio observados no robô em função da taxa de dados em diversas configurações de velocidade. É possível observar que não somente a velocidade do robô impacta no tempo total de trabalho, mas a taxa de dados também, ainda que em proporção menor. Observa-se também uma leve alteração de tempo entre as taxas de 115200 e 57600 *baud rate*, alteração está menor que nas demais taxas. Este comportamento é esperado pois não somente a transmissão é contabilizada no tempo geral, mas a capacidade de processamento do computador que está a enviar os dados, desta forma o tempo de processamento do computador não é alterado juntamente com a taxa de dados.

Gráfico 6-Tempo em função da taxa de dados.

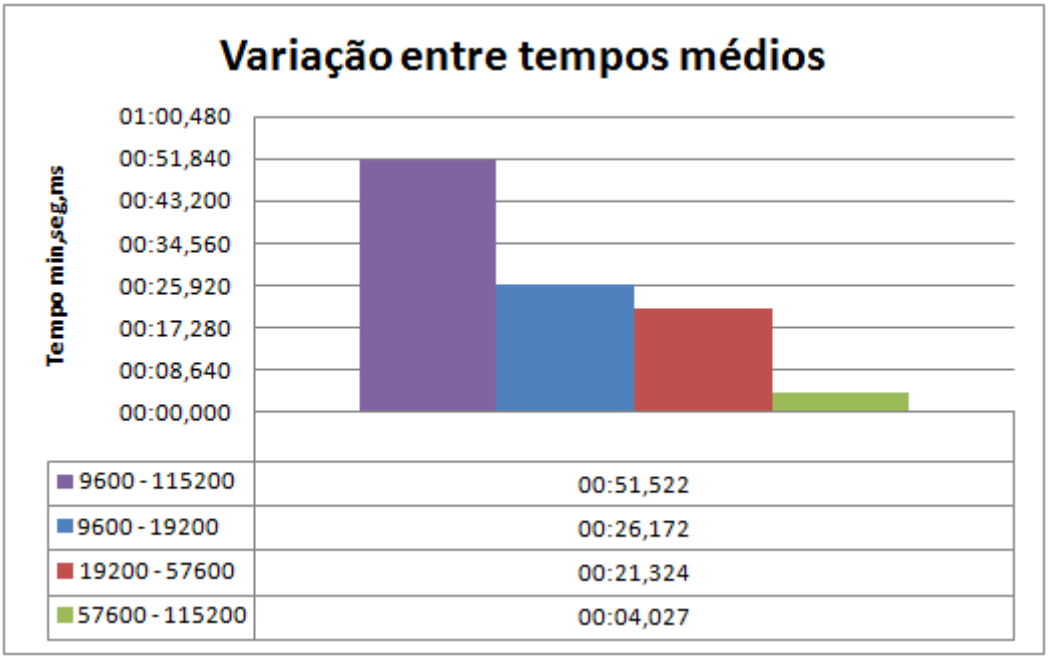


	10	9	8	7	6	5	4
9600	22:16,444	20:10,882	18:10,348	16:01,511	13:54,796	11:48,411	09:40,985
19200	21:47,334	19:42,201	17:42,185	15:30,071	13:28,755	11:32,357	09:17,274
57600	21:27,072	19:21,447	17:17,598	15:09,497	13:14,049	11:07,089	08:54,160
115200	21:25,665	19:20,189	17:13,284	15:09,200	13:03,935	10:59,136	08:51,315

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

A diferença mencionada é visível melhor no gráfico 7 que apresenta a diferença entre as médias de diferença tempo entre as taxas de dados utilizadas. Nota-se uma diferença de apenas 4 segundos entre a taxa de dados de 57600 e 115200 de *baud rate*.

Gráfico 7-Variações de tempo.



9600 - 115200	00:51,522
9600 - 19200	00:26,172
19200 - 57600	00:21,324
57600 - 115200	00:04,027

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

6.4 Repetibilidade e precisão

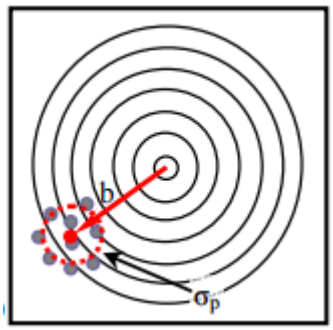
Alguns conceitos importantes a serem considerados no estudo de robôs industriais são suas propriedades de exatidão, resolução e repetibilidade. Estas características analisadas em robôs não se restringem somente a um tipo de tipo de robô, mas sim a praticamente todos os robôs manipuladores.

A resolução do robô está associada a medida de alguma grandeza, neste trabalho, a grandeza trabalhada se caracteriza por deslocamento linear, neste caso, a resolução é o menor valor encontrado medido que o robô pode executar, se deslocar. Na aplicação desenvolvida, cara motor de passo tem ângulo de deslocamento de 7,5°, e este valor corresponde ao incremento mínimo que o robô pode fazer, então para um deslocamento de 1mm o robô precisa realizar 45,8 passos, valor este definido na seção de calibração, isso corresponde então a uma resolução de 1/45,8 ou 0,021834 mm.

A exatidão se refere à distância mínima que é possível posicionar o efetuador do robô em uma determinada posição no espaço, como detalham Monico. et al. (2009). A exatidão está diretamente relacionada ao erro de posicionamento absoluto do robô.

A repetibilidade é a medida encontrada ao analisar o resultado de repetitivas ações do robô para se posicionar em um ponto específico e previamente armazenado. Este valor é relacionado ao erro de posicionamento relativo, podendo ser calculado com a média aritmética de todas as tentativas de posicionamento em comparação com o ponto que realmente se deseja posicionar o robô. A Imagem 50 demonstra o valor calculado para um cenário de cálculo de repetibilidade, a distância da reta b na imagem representa o valor encontrado para a repetibilidade no teste.

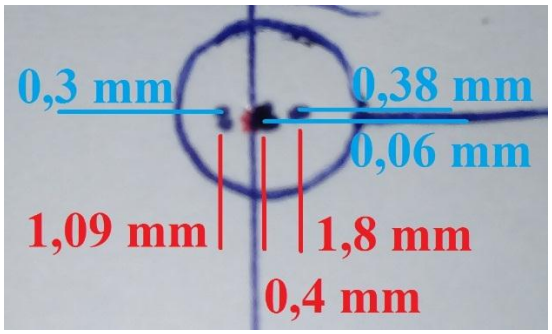
Imagem 50 – Cálculo de repetibilidade.



Fonte: MONICO, João Francisco Galera. et al. 2009.

A imagem 51 apresenta o resultado do teste de repetibilidade realizado pelo robô desenvolvido, através da imagem é possível calcular o ponto médio de posicionamento e também a distância encontrada entre o ponto médio e o ponto desejado.

Imagem 51- Teste de repetibilidade.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

O teste foi realizado com a marcação de 10 pontos pelo robô na posição de 200 mm em X e 100 mm em Y, dos 10 pontos marcados 4 pontos se mostram mais afastados da concentração dos demais, 6 pontos, considerando os valores mensurados na imagem 51 o valor encontrado mostra um erro médio de 0,99 mm em relação ao ponto desejado, Romano (2002) menciona que robôs industriais se caracterizam basicamente por possuírem uma alta repetibilidade e baixa exatidão, um número muito popular para repetibilidade em robôs industriais é 0,1 mm, existem ainda robôs mais precisos, como robôs do tipo SCARA que podem alcançar valores de repetibilidade na faixa de 0,01mm.

6.5 Falhas gerais

Com base nos dados apresentados é possível identificar os pontos do sistema que estão gerando erros, principalmente erros matemáticos nas etapas de cálculo de translação e rasterização. Como o projeto se propõe a ser uma ferramenta funcional para possível uso didático, o sistema deve ser o mais simples possível, e com isso, algoritmos mais simples foram utilizados, consequentemente estes algoritmos podem não ser a melhor opção para o que se propõem, como exemplo pode-se usar o algoritmo de Bresenham para substituir o algoritmo básico incremental na rasterização, como trabalha só com inteiros o erro acumulado é muito menor, mas em contrapartida, é um algoritmo mais complexo e extenso.

As falhas encontradas no robô são caracterizadas basicamente por erros de arredondamentos de valores e falhas mecânicas, esta última implicando em um número bem menor de falhas, para tanto, as falhas não comprometem o trabalho final ao que o robô se propõe.

7 AVALIAÇÕES

Este tópico trata de análises sobre algumas avaliações do robô que não suas capacidades e sim sua compatibilidade, custo estimado de construção, *softwares* CAM testados.

7.1 Custos

Para a construção física do robô foi optado por utilizar o máximo possível de materiais reciclados como: motores, guias lineares e rolamentos que são facilmente encontrados em sucata eletrônica, porém para um projeto deste tipo o conhecimento do custo total é muito importante, então na tabela 10 são apresentados os componentes utilizados no robô e informações como valor em caso de compra do componente novo e a informação se o que foi utilizado é de origem de sucata ou foi adquirido novo.

Tabela 10-Tabela de custos do projeto

Item	Quantidade	Origem	Valor
Motor de passo	3	Sucata	120,00
Solenoide	1	Sucata	10,00
Barras lineares	4 (50 cm)	Sucata	60,00
Barras roscadas 5 mm	2 (metro)	Novo	15,00
Rolamentos	3	Sucata	15,00
Cantoneira 1/2 “	4 (metro)	Novo	13,00
Castanhas/buchas	6	Sucata	60,00
Acoplamento	3	Sucata	45,00
Parafusos	32	Novo	6,00
Porcas	40	Novo	4,00
Ventoinhas 40x40	3	Sucata	21,00
Microcontrolador	1	Sucata	10,00
CI ULN2003	2	Sucata	4,00
CI MAX232	1	Novo	5,00
Demais componentes	X	Novo	25,00
Base de MDF	1 (55X45) cm	Novo	40,00
Fonte 12V	2	Sucata	20,00
Total para componentes novos:			473,00

Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Os custos de montagem, soldagem, corte e afins não foram contabilizados, pois, não foram utilizados de terceiros. A construção se deu com o uso de ferramentas comuns sem a necessidade de trabalho especializado, o que não gerou custo adicional.

7.2 Compatibilidade

A capacidade do robô em ser acionado por código G visa garantir seu funcionamento com diversos *softwares* CAM disponíveis, porém existem algumas limitações que são particulares de cada robô deste tipo, para isso alguns *softwares* possibilitam ajustes em seu sistema capaz de garantir o funcionamento destes robôs mesmo com suas particularidades, outros podem não permitirem tantos ajustes, o que limita o número de robôs compatíveis, contudo, alguns dos *softwares* mais populares entre os hobistas e comerciais foram testados. Todos esses *softwares* testados possuem a capacidade de exportar um arquivo no formato .nc com o código G gerado, dessa forma, podem ser testados no robô desenvolvido.

Os *softwares* CAM testados foram: PCItoGcode que foi utilizado como referência para geração de código G na etapa de desenvolvimento e o ArtCam que em buscas principalmente pela internet foi um *software* muito comentado em fóruns e discussões entre profissionais que trabalham com sistemas CNC além de programações em código G escritos manualmente em arquivos texto.

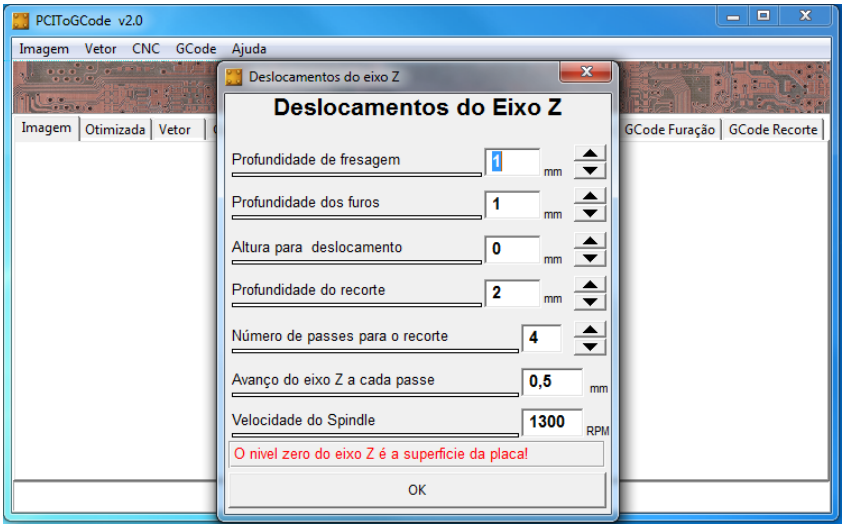
7.2.1 PCItoGcode

Este *software* permite a configuração de valores para altura para deslocamento no eixo Y, profundidade de corte, profundidade de furação e profundidade de recorte além de avanço para cada passo no eixo z e velocidade do *Spindle* “ferramenta de corte”.

Considerando as características do robô desenvolvido, é possível então definir a profundidade de fresagem e de furação para 1mm, como qualquer valor diferente de 0 implica em acionamento do efetuador, ao informar um valor diferente de 0 o efetuador será acionado, para a altura de deslocamento deve-se informar o valor 0, para que o efetuador seja desligado, profundidade de corte também segue o mesmo padrão, porém o *software* exige um valor mais que o valor de fresagem, então foi escolhido arbitrariamente o valor 4, em avanço do eixo Z também precisa de um valor mínimo, ainda que seja diferente de 0 não interfere no funcionamento do robô, então foi configurado em 0,5mm e por fim a velocidade do *Spindle*, que, neste caso, pode ser qualquer valor, pois, o comando de velocidade de *spindle* será ignorado pelo robô pois este comando não é implementado neste projeto.

A imagem 52 exibe a tela de configurações do *software* já com os valores citados configurados.

Imagem 52- Configuração do eixo Z no PCIToGCode.



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Seguindo estas configurações o robô funcionou perfeitamente com o *software* em questão, nenhum problema foi detectado durante seu funcionamento.

7.2.2 ArtCam

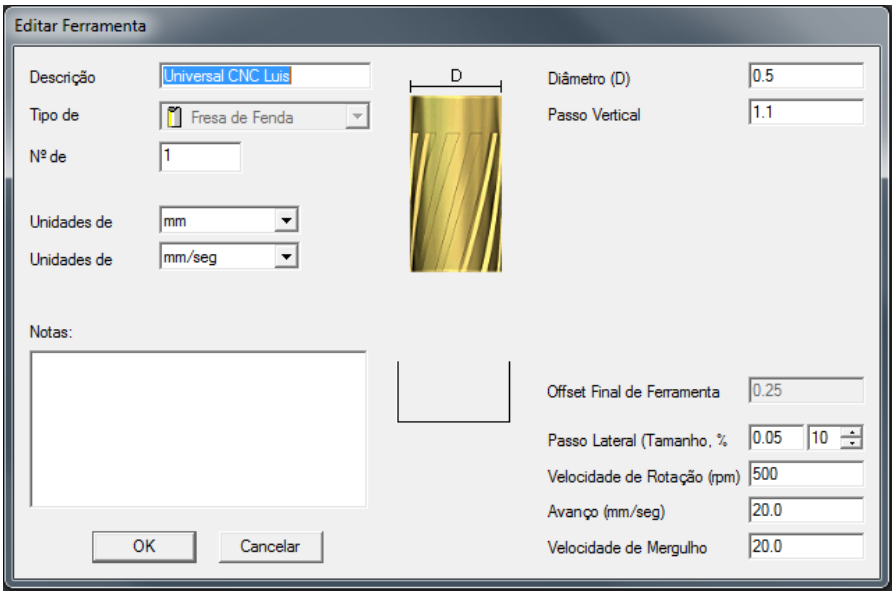
O *software* ArtCam é desenvolvido pela Autodesk voltado para usinagem de modelos 2D e 3D em madeira e em outros materiais. O ArtCam permite que possa ser projetado diretamente em sua interface modelos 3D e 2D ou importado de alguma outra ferramenta de modelagem 3D, CAD.

Uma vez que o modelo a ser usinado esteja pronto, o ArtCam conta com uma ferramenta CAM incorporada, capaz de gerar a programação em código G para diversos modelos de robôs industriais compatíveis com CNC. Em virtude de sua compatibilidade e versatilidade é largamente utilizado por hobistas e entusiastas além dos profissionais da área.

Este *software* foi utilizado ao longo do desenvolvimento do trabalho para gerar alguns dos testes aplicados no robô desenvolvido, para isto, as configurações do *software* foram ajustadas para que o robô fosse capaz de executar o código gerado. A configuração mais relevante para garantir o funcionamento do robô desenvolvido é a adição de uma ferramenta de trabalho no ArtCam, a ferramenta adicionada tem as seguintes características: Tipo Fresa de

Fenda, unidades em mm e mm/seg, diâmetro 0,5 mm, passo vertical 1,1 mm, passo lateral 0.05%, as demais características não são implementadas no robô portanto não farão diferença.

Imagem 53-Configurações da ferramenta



Fonte: OLIVEIRA, Luis Felipe dos Santos. 2016.

Com a ferramenta já presente no software, bas então selecionar o vetor que deseja trabalhar e traçar um perfil em 2D que pode ser traçado ao longo do vetor, na face interna ou na face interna, e configurar a profundidade inicial da ferramenta em 0 mm e a profundidade final em -1 mm, desta forma o efetuador será acionado sempre que a ferramenta incidir na posição -1, por fim basta salvar o percurso da ferramenta no formato “Axyz (*.nc)” presente no campo saída de formato de máquina.

O arquivo gerado pelo ArtCam possui o formato .nc com a programação em código G do vetor escolhido no programa, este por sua vez deve ser importado no *software* de comunicação desenvolvido neste trabalho para que o robô possa executá-lo. Como mencionado, este *software* foi utilizado ao longo do trabalho utilizando o detalhamento descrito, desta forma, a compatibilidade com este *software* foi testada.

8 CONSIDERAÇÕES FINAIS

A realização deste trabalho envolveu pesquisas em diversas áreas, o envolvimento de mecânica, eletrônica, computação, programação e pesquisa o torna um trabalho muito abrangente. As dimensões tomadas pelo desenvolvimento geral do trabalho limitaram o aprofundamento em alguns aspectos, porém de forma geral os pontos abordados possuem detalhamento suficiente para o tornar compreensível, o que é um dos objetivos alcançados pelo trabalho, além de fornecer material de referência para outros trabalhos ou projetos de hobistas e ainda práticas de sala de aula.

Ainda que o trabalho não envolva programações complexas, os limites principalmente de *hardware* envolvendo o microcontrolador tornou-se um desafio, desta forma, o conhecimento e práticas de programação em linguagens Java e C foram imprescindíveis para garantir o sucesso do trabalho. Com isso, a gama de recursos que podem ser explorados é muito grande, o que possibilita que outros trabalhos explorem técnicas em áreas como processamento, comunicação, controle de motores e atuadores, interpretação de programação entre outros.

Os dados gerados e analisados neste trabalho visam servir de referência e material para comparações em trabalhos futuros de mesmo segmento, confrontar algoritmos e técnicas de soluções de problemas, bem como outras formas de, por exemplo, controlar os motores ou o uso de outros motores, uso de encoders, outra interface para promover a comunicação podem gerar resultados diferentes, então estes dados apresentados aqui podem servir como ponto de partida para melhorias e analises de atuação de robôs cartesianos. O cruzamento de dados utilizado também pode ser comparado com outros resultados, como a influência do hardware do PC utilizado e o impacto dele no trabalho final é outro ponto a ser explorado, além dos dados cruzados com os dados já obtidos.

Os materiais utilizados principalmente na composição da UC foram escolhidos considerando também a facilidade de localização no comércio e o baixo custo, e é claro, sendo capaz de atender as necessidades do projeto, desta forma, a replicação do projeto envolve um custo relativamente baixo, sem a necessidade de componentes e ferramentas complexas, o que por sua vez favorece a replicação do projeto e seu uso como ferramenta para auxílio do professor em práticas de laboratório. O método de gravação do microcontrolador exposto no trabalho e a técnica de coleta de dados também foram utilizados e desenvolvidos para que possa ser facilmente replicado em o uso de ferramentas especificas.

Sendo assim, é possível afirmar que o objetivo geral do trabalho de desenvolver um robô cartesiano foi alcançado de forma satisfatória, seu funcionamento pôde ser demonstrado e

avaliado, algumas características do robô foram mensuradas, analisadas e apresentadas, além de sua compatibilidade com programação em código G testada e utilizada durante seu desenvolvimento e fase de testes, e por fim, o objetivo de ser referência para outros trabalhos e poder servir como uma ferramenta para professores também foi alcançado com uso de técnicas simples, porém eficientes para criar e avaliar o projeto desenvolvido.

REFERÊNCIAS

ATHANI, V. V. Stepper Motors: Fundamentals, Aplications and Design. New Delhi: New Age International (P) Ltd. 2005.

BARDELLI, Rubens. Bard HP. 2012. Disponível em: <http://www.vabsco.com/bardhp/proj/cnc/main.html>. Acesso em: 08 maio 2016.

BLIKSTEIN, Paulo. Digital Fabrication and 'Making' in Education: The Democratization of Invention. 2013. Disponível em: <https://tltl.stanford.edu/sites/default/files/files/documents/publications/2013.Book-B.Digital.pdf >. Acesso em: 20 maio, 2016.

BRITES, F. G.; SANTOS, V. P. de A. Motor de Passo. Rio de Janeiro: PETele. 2008. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/tutoriais/stepmotor/stepmotor2k81119.pdf>. Acesso em: 21 maio, 2016.

CASSANIGA, Fernando Aparecido. Fácil Programação do Comando Numérico FANUC. São Paulo: CNC Tecnologia Editora LTDA, 2005.

CRAIG, Jhon J.. *Introduction of robotics*: Mechanics and control. 2 ed. California: Addidon-Wesley Publishing Company, 1989.

DATASHEET do CI max232x. Texas Istruments, 2014. Disponível em: <http://www.ti.com/lit/ds/symlink/max232.pdf>. Acesso em: 28 maio, 2016.

DATASHEET do microcontrolador PIC16F628A. Texas Microchip inc., 2007. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/40044F.pdf>. Acesso em: 28 maio, 2016.

DATASHEET do microcontrolador AT90S2313. Atmel Corp., 2002. Disponível em: <http://www.atmel.com/images/doc0839.pdf>. Acesso em: 28 maio, 2016.

DATASHEET do CI ULN2003. Texas Istruments, 2016. Disponível em: <http://www.ti.com/lit/ds/symlink/uln2003a.pdf>. Acesso em: 28 maio, 2016.

DATASHEET do CI LM7805. Fairchild, 2014. Disponível em: <https://www.fairchildsemi.com/datasheets/LM/LM7805.pdf>. Acesso em: 28 maio, 2016.

DATASHEET do motor PM35L-48. Minebea, 2002. Disponível em: <https://www.futurlec.com/Datasheet/Stepper/PM35L_Stepper_Motor.pdf>. Acesso em: 30 maio, 2016.

FREIRE, J. M. Fresadora. Rio de Janeiro: Editora S. A., 1983.

GIL, Antônio Carlos. Métodos e técnicas de pesquisa social. São Paulo: Atlas, 2008.

GONÇALVES, E.L.Z. Inovação no processo produtivo no segmento metal-mecânico com uso de tecnologia a CNC (pesquisa do perfil profissional). In: Anais do XXXIV COBENGE, 2006, Passo Fundo. Anais. Passo Fundo: Universidade de Passo Fundo, 2006, p. 13.6-13.21.

KRAR, Stephen F.; GILL, Arthur; SMID, Peter. *Computer Numeric Controls Simplified. New York*: Industrial Press Inc., 2001.

LAURETO, Jhon J.; DESSIATOUN, Serguei V.; OHADI, Michael M.; PEARCE, Joshua M.. *Open Source Laser Polymer Welding System*: Design and Characterization of Linear Low-Density Polyethylene Multilayer Welds. Disponível em: <http://www.mdpi.com/2075-1702/4/3/14>. Acesso em: 22 maio, 2016.

LAZZARIN, J. C., construção de um manipulador robótico de baixo custo para ensino. Cascavel. 2012. Disponível em: <http://www.inf.unioeste.br/~tcc/2012/TCC_Julio.pdf>. Acesso em: 20 de maio, 2016.

LEÃO, Lucas. *CAD, CAE e CAM*: Qual a diferença. 2015. Disponível em: <http://www.cim-team.com.br/blog-engenharia-eletrica-moderna/cad-cae-e-cam-qual-a-diferenca>. Acesso em: 15 de maio 2016.

LOPES, João Menual Brisson. Rasterização. 2013. Disponível em: <http://disciplinas.ist.utl.pt/leic-cg/textos/livro/Rasterizacao.pdf >. Acesso em: 11 out. 2016.

LYNCH, Mike. Moderne Machine Shop. 1998. Disponível em: <http://www.mmsonline.com/articles/the-key-concepts-of-computer-numerical-control>. Acesso em: 10 de maio 2016

MACHADO, Aryoldo. *Comando Numérico*: Aplicado às máquinas industriais. São Paulo: Ícone Editora, 1994.

MARTIN, Lee. The Promise of the Maker Movement for Education. Journal of Pre-Chollege Engineering Education Research. California, n. 4, p. 30-39, 2015.

MENDES, D. R. *Programação Java*: Com ênfase em orientação a objetos. São Paulo: Novatec, 2009.

MONICO, João Francisco Galera. et al. *Acurácia e precisão*: Revendo os conceitos de forma acurada. Curitiba, 2009. Disponível em: <http://revistas.ufpr.br/bcg/article/viewFile/15513/10363>. Acesso em: 11 out. 2016.

MUSSOI, Fernando. Luiz. R. Fundamentos de Electromagnetismo. Disponível em: <http://intranet.ctism.ufsm.br/gsec/Apostilas/Eletromagnetismo.pdf>. Acesso em: 24 de maio 2016.

ORLANDINI, Caio Bussadori. Adaptador isolado de comunicação serial. 2011. Disponível em: <http://www.tcc.sc.usp.br/tce/disponiveis/18/180450/tce-19102011-100526/publico/Orlandini_Caio_Bussadori.pdf >. Acesso em: 25 de maio 2016.

PAIOTTI, Renato. Comunicação Serial Usando o Protocolo RS232. 2003. Disponível em: <http://www.newtoncbraga.com.br/index.php/eletronica/52-artigos-diversos/12095-comunicacao-serial-usando-o-protocolo-rs232-tel213>. Acesso em: 11 maio, 2016.

PCITOGCODE: conversão de imagem em código G, voltado para placas de circuito impresso. São Paulo: Rubens Bernardi, 2013. Disponível em: <https://sourceforge.net/projects/pcitogcode/>. Acesso em: 12 dez. 2016.

PERCHÉ, Carlos Felipe de Paiva. Desenvolvimento de um sistema de prototipagem para placas de circuitos impressos, Itabira, 2013. Disponível em: <http://pt.slideshare.net/CarlosFelipe4/tcc-carlos-felipe-de-paiva-perch>. Acesso em 22 maio, 2016.

PEREIRA, Fábio. *Microcontroladores PIC*: programação em C. São Paulo: Editora Ética Ltda, 2011.

PINTO. Luiz A. V. Técnicas com Sistemas Digitais. [2008?]. Disponível em: <http://www.vargasp.com/download/livros/Tecnicas_digitais.pdf>. Acesso em: 04 maio, 2016.

ROMANO, Vitor Ferreira (Org.). *Robótica Industrial*: Aplicação na indústria de manufatura e de processos. São Paulo: Editora Edgard Blücher LTDA, 2002.

ROSÁRIO, João Maurício. Princípios de Mecatrônica. São Paulo: Editora Pearson Prentice Hall, 2005.

SOUZA, A. F.; ULBRICH, C.B.L. *Engenharia integrada por computador e sistemas CAD/CAM/CNC*: Princípios e aplicações. São Paulo: Artliber Editora, 2009.

SUEIRO, Diego. Estado do Movimento Maker no Brasil, 2015. Disponível em: <http://www.embarcados.com.br/estado-do-movimento-maker-no-brasil/>. Acesso em: 10 maio, 2016.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. Fresadora PCI João de Barro. Porto Alegre, 2014. Disponível em: <http://cta.if.ufrgs.br/projects/fresadora-pci-joao-de-barro/wiki/Wiki>. Acesso em: 21 de maio, 2016.

VALE, Valentina C. Controle de Posição de um Robô Cartesiano por meio de Técnicas Adaptativas. 2011. Disponível em: <http://bdtd.biblioteca.ufpb.br/bitstream/tede/5311/1/arquivototal.pdf>. Acesso em: 22 maio, 2016.

WALKOWIAK, Tomasz. Applications of Virtual Laboratories in Teachin at Techinical Universities. Journal of Digital Information Management. Wroclaw, n. 4, p. 202-204, September, 2005.

WANG, Yiqiang; JIA, Yazhou; YU, Junyi; YI, Shangfeng. Field failure database of CNC lathes. International Journal of Quality & Reliability Management, Changchum, n. 4, p. 330-343, 1999.