CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

***Campus* DIVINÓPOLIS**

GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

DESENVOLVIMENTO DE UM MANIPULADOR DIDÁTICO COM CONTROLADOR EMBARCADO OPERADO NO ESPAÇO ORTONORMAL POR JOYSTICK

Vinícius Luís dos Reis

Divinópolis, 2014.

Vinícius Luís dos Reis

DESENVOLVIMENTO DE UM MANIPULADOR DIDÁTICO COM CONTROLADOR EMBARCADO OPERADO NO ESPAÇO ORTONORMAL POR JOYSTICKS

Monografia de Trabalho de Conclusão de Curso apresentada ao Colegiado de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para obtenção do título de Engenheiro Mecatrônico.

Eixos de Formação: Elétrica, Mecânica, Programação

Orientador: Renato de Sousa Dâmaso

Divinópolis, 2014.

Centro Federal de Educação Tecnológica de Minas Gerais
CEFET-MG / Campus Divinópolis
Curso de Engenharia Mecatrônica

Monografia intitulada "*Desenvolvimento de um manipulador didático com controlador embarcado operado no espaço ortonormal por joysticks* ", de autoria do graduando Vinícius Luís dos Reis, aprovada pela banca examinadora constituída pelos seguintes professores:

_____

Prof. Dr. Renato de Sousa Dâmaso - CEFET-MG / Campus Divinópolis - Orientador

_____

Prof. M.Sc. Alan Mendes Marotta - CEFET-MG / Campus Divinópolis

_____

Prof. M.Sc. Josias Gomes Ribeiro Filho - CEFET-MG / Campus Divinópolis

_____

Prof. Dr. Valter Júnior de Souza Leite
Coordenador do Curso de Engenharia Mecatrônica
CEFET-MG / Campus Divinópolis

Divinópolis - Abril de 2014

RESUMO

Este trabalho tem como objetivo o desenvolvimento de um manipulador robótico articulado de pequeno porte com seis graus de liberdade e controlador embarcado. A estratégia principal de construção se baseia na utilização de itens de baixo custo e fácil acesso, tais como servo motores utilizados em modelismo e estrutura em madeira MDF, sugerindo-se a utilização deste modelo de manipulador em kits educacionais e aulas experimentais.

O controle do robô é feito de forma que o operador manipule variáveis referentes à posição da garra do robô em relação à base. Esta manipulação é feita a partir de um *joypad* comercial, utilizado em *videogames* e empregado neste projeto como terminal de programação *(teach-in pendant),* em modo de aprendizagem. Para que o manipulador possa realizar a tarefa ensinada, é utilizado o modelo cinemático do sistema definido na etapa de projeto.

A eletrônica do robô é baseada no microcontrolador PIC18F46K22 de forma a proporcionar que esse sistema seja embarcado. Toda a programação é feita em linguagem C e desenvolvida de forma que os valores recebidos do *teach-in pendant* por comunicação serial SPI fossem utilizados nas equações de cinemática inversa, cujos resultados são traduzidos em pulsos para o controle de cada motor individual.

O controlador também utiliza a memória EEPROM do microcontrolador para disponibilizar a opção de treinamento para o manipulador. Nesta opção de treinamento (*teaching*), é possível gerar trajetórias para o robô utilizando polinômios cúbicos, se determinado os pontos de via e o intervalo de tempo em que o movimento deve ser feito. Para cumprir estas trajetórias, em modo automático, o robô utiliza os polinômios cúbicos gerados, de forma a conferir suavidade ao movimento.

Ao final, é conduzido um conjunto de testes de funcionamento e desempenho do projeto, determinando parâmetros como repetitividade e precisão. Concluindo que o manipulador possui funcionamento satisfatório.

Palavras chaves: manipulador, robótica, embarcado, didático

ABSTRACT

This work has as objective the development of a robotic articulated manipulator, small size, six degrees of freedom and embedded controller. The main strategy in the construction is based on the utilization of low cost items and easily accessed, as servo motors used in modelism and structure in MFD wood, suggesting the utilization of this manipulator model on educational kits e experimental classes.

The robot control is made in a way that the operator is able to manipulate variables referred to the position of the robot grab in relation to the base. That manipulation is make from a commercial joypad, common in videogames and used in this work as Teach-in Pendant, in the teaching mode. To the manipulator be able to perform a teached task, it is used a kinematic model which is defined in the project stage.

The robot electronic is based on the microcontroller PIC18F46K22 in a way to provide that this system become embedded. All the programming are based in C language and was developed in a way that the values received from the teach-in pendant through SPI wore utilized in the inverse kinematics equations, which results are translated in pulses for the control of each individual motor.

The controller also utilizes the EEPROM memory from the microcontroller to provide the training option for the manipulator. In that teaching option, it is possible to generate trajectories for the robot utilizing cubic polynomials, if determined the via points and the time interval in which the movement must be made. To fulfill those trajectories, in automatic mode, the robot uses the cubic polynomials generated, in a way to grant suavity to the movement.

In the end, it will be conducted the tests of the functioning and the performance of the project, determining parameters like repeatability and accuracy. Concluding that the manipulator has a satisfactory functioning.

Keywords: manipulator, robotics, embedded, didactic

AGRADECIMENTOS

Ao professor Renato Dâmaso, pelo suporte, confiança, incentivo e orientação durante a realização deste trabalho.

Ao professor Sandro Mordente, pela orientação durante a primeira etapa deste trabalho e pelas indicações que foram necessárias.

Ao professor Alan Marotta, pelas diversas vezes que precisei de direções em relação à parte eletrônica do meu projeto.

Ao professor Josias Ribeiro, pela ajuda em análises estruturais do projeto.

Ao meu colega Leonardo Elias Antunes, pela ajuda diversas vezes, tanto na parte de projeto mecânico quanto na montagem.

À minha família, pelo incentivo diversas vezes, não só durante a etapa de realização deste trabalho, mas também durante todas as etapas de meu curso.

À minha namorada Mariane, por ter me acompanhado desde o início deste trabalho, sendo suporte e incentivo.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMO

GDL: Grau de liberdade.

TP: *Teach in Pendant.*

n.d.: Não datado.

DH: Denavit-Hartenberg.

CAD: Computer Aided Design

$s\theta_i$: $sen\theta_i$

$c\theta_i$: $\cos\theta_i$

1. INTRODUÇÃO

A robótica industrial surgiu da necessidade de máquinas que pudessem substituir o homem em operações repetitivas e muitas vezes em ambientes de difícil acesso. Muito antes disso, a humanidade já possuía a fantasia de desenvolver máquinas que substituíssem seu esforço. Provas arqueológicas mostram que desde os tempos helenísticos – por volta de dois mil anos atrás – a humanidade se ocupava em criar aparatos autônomos com o objetivo de servir outros seres humanos, como, por exemplo, instrumentos de guerra e de trabalho (D´Ignazio, 1982). O próprio termo robô tem origem na palavra tcheca *robotnik,* que quer dizer 'servo', o que demonstra a ideia de desenvolver uma máquina que fizesse esforço no lugar do homem.

Charles Carter menciona em (Nof, 1985, p. 9) que, historicamente, o trabalho automatizado foi criado por três razões básicas:

1. A energia necessária para conduzir uma tarefa, ou o ambiente onde é conduzida a tarefa, muitas vezes vai além da resistência humana.
2. A habilidade necessária para produzir vai além da capacidade humana.
3. A demanda pelo produto é tão grande que motiva a procurar métodos melhores.

É possível observar que a robótica industrial é capaz de suprir as necessidades citadas analisando um exemplo básico de aplicação, a soldagem. A robótica aplicada à soldagem, aplicação mais comum entre os robôs industriais comercializados pela empresa sueca ABB, com um percentual de 33% (ABB, 2000), foi uma ferramenta de grande avanço para a indústria, incluindo a indústria automotiva onde o índice de automatização em âmbito mundial é de atualmente 90% (Rosário, 2005).

Além de possuir capacidade de trabalhar sem intervalos, devendo parar somente se for necessária alguma manutenção demonstrando sua superioridade sobre um operário em energia para trabalhar, os manipuladores industriais podem também ser programados. Utilizando linguagens próprias de programação é possível

treinar um manipulador para trabalhar com capacidade de precisão e repetitividade muito altos, o que é inatingível a habilidade humana.

A história demostra que a robótica industrial tem evoluído exponencialmente ao redor do mundo, resultando tanto na diminuição de preços dos equipamentos quanto em avanços em eficiência (Nof, 1985). É possível citar, como exemplo de fator responsável pela evolução na robótica industrial, o desenvolvimento da microeletrônica (Rosário, 2005, p. 145). A possibilidade de concentrar diversos componentes em um espaço mínimo teve impacto direto na capacidade de processamento de informações.

A partir da evolução da robótica foi possível observar uma grande diminuição nos custos, tornando os produtos muito mais acessíveis. Com o avanço em atuadores, cada vez mais simples e menores, é possível projetar e desenvolver um manipulador robótico com características complexas, como um alto número de graus de liberdade, mesmo com um baixo orçamento.

Algumas características, como alcance e força, são diretamente relacionadas aos atuadores utilizados nas juntas. No desenvolvimento de um manipulador de baixo custo é possível que estas especificações sejam bastante limitadas, o que restringe a utilidade do robô.

Diversas aplicações optam por um manipulador robótico de baixo custo e que não requerem grandes especificações de alcance ou força. Como exemplo, tem-se o ambiente de sala de aula. A robótica nas escolas é um importante aliado em desenvolver conhecimentos e motivar o interesse de alunos pela tecnologia. Diversos artigos como (Veja, 2012), (G1, 2011) e (UOL, 2011) discorrem sobre a importância e a aplicabilidade da robótica para alunos de ensino fundamental e médio. Destes artigos é possível destacar a reação dos estudantes ao encontrar na robótica aplicabilidade direta a conceitos matemáticos vistos em salas de aula.

Em universidades e instituições de ensino superior também é possível observar aplicabilidade para um manipulador com estas características, visto que a aquisição de um manipulador industrial nem sempre é possível, dependendo de diversos fatores como custos e dificuldades burocráticas.

A utilização do manipulador desenvolvido no presente trabalho em uma sala de aula de ensino superior de engenharia pode ser muito útil na apresentação de conceitos básicos de robótica industrial, tais como posicionamento e orientação, e também conceitos aplicados como modelagem cinemática e dinâmica, além de geração de trajetória, programação e controle.

1.1. Objetivo

O objetivo deste trabalho é apresentar o desenvolvimento de um manipulador robótico articulado, de pequeno porte, com seis graus de liberdade e controlador embarcado. A estratégia principal de construção se baseia na utilização de itens de baixo custo e fácil acesso, tais como estrutura em madeira MDF e servo motores comuns em modelismo. Tal estratégia sugere a obtenção de uma alternativa para kits educacionais e aulas experimentais em decorrência deste trabalho.

O controle do manipulador é feito de forma que o operador insira dados de posicionamento e orientação da ferramenta terminal do manipulador através de um joystick, utilizado neste trabalho como terminal de programação. O manipulador realiza os cálculos referentes à variação em ângulo para cada junta com o objetivo de atingir o ponto esperado com precisão.

A montagem mecânica do manipulador é desenvolvida de forma a conferir à estrutura resistência, suavidade nos movimentos e precisão. Desta forma, todos os materiais utilizados na montagem foram escolhidos e dimensionados para atender a tais exigências.

1.2. Metodologia

A robótica é uma das áreas mais multidisciplinares da engenharia, ela busca integrar técnicas e algoritmos na criação de dispositivos mecatrônicos (robôs). Entre as áreas mais estudadas estão a mecânica, eletrônica, programação e controle, com aplicação de diversos conceitos da física e da matemática. Como metodologia adotada para o projeto, foi desenvolvido um estudo relacionando cada área da robótica separadamente e integralizando todos os conteúdos com a montagem do manipulador.

Como primeiro passo, foi realizado um estudo referente aos aspectos estruturais do manipulador articulado desenvolvido. Ainda nesta etapa, é apresentada a modelagem cinemática para o robô, desenvolvida através de métodos como *Denavit-Hartenberg* e cinemática inversa. Através do uso de equações de cinemática inversa, são determinados os valores articulares utilizando coordenadas cartesianas referentes à posição final da ferramenta para a tarefa. O pacote *Robotics Toolbox* do software *Matlab,* desenvolvido por Corke (2002), foi utilizado para esse desenvolvimento*,* permitindo comparar os valores parametrizados com a especificação do manipulador.

Para o projeto mecânico, foi feito o desenho técnico utilizando software de CAD e modelagem 3D. O projeto é feito cotando materiais para a montagem, considerando fatores como suavidade e precisão. Logo em seguida, é desenvolvida a estrutura a partir de ferramentas computacionais de modelagem e avaliadas as resistências de partes dessa estrutura a partir de equações de torque.

O controle do robô foi feito de forma que o operador manipule variáveis referentes à posição da garra do robô em relação à base. Esta manipulação é realizada a partir de um *joypad* comercial, utilizado em *videogames* e empregado neste projeto como terminal de programação (*teach-in pendant*), em modo de aprendizagem. Para que o manipulador possa realizar a tarefa ensinada, é utilizado o modelo cinemático do sistema definido na etapa de projeto.

O controlador do robô foi desenvolvido utilizando o microcontrolador PIC18F46K22, de forma a proporcionar que esse sistema fosse embarcado. Sistemas embarcados são, por definição, sistemas microprocessados desenvolvidos para empenhar uma função específica e totalmente dedicado ao dispositivo que controla (Architects, 2010), (Health, 2003). Toda a programação é feita em linguagem C. Ela foi desenvolvida de forma que os valores recebidos do *teach-in pendant* por comunicação serial SPI fossem utilizados nas equações de cinemática inversa. Os resultados obtidos são traduzidos em pulsos para o controle individual de cada motor.

O controlador também utiliza a memória RAM do microcontrolador para disponibilizar a opção de treinamento para o manipulador. Nesta opção de treinamento (*teaching*) é possível gerar trajetórias para o robô, se determinado os

pontos de via e o intervalo de tempo em que o movimento deve ser feito, utilizando polinômios cúbicos. Para cumprir estas trajetórias, em modo automático, o robô utiliza os polinômios cúbicos gerados, de forma a conferir suavidade ao movimento.

Para o projeto mecânico, foi feita a escolha de materiais baseando em quatro aspectos: preço, resistência, suavidade nos movimentos e precisão. Toda a modelagem mecânica foi desenvolvida utilizando ferramentas em CAD 3D, por oferecer, dentre outras vantagens, a praticidade em gerar planilhas de corte em madeira.

Ao final do trabalho é apresentado um conjunto de testes de funcionamento e desempenho do projeto, determinando parâmetros como repetitividade e precisão.

2. MODELAGEM CINEMÁTICA

Grande parte dos manipuladores robóticos são constituídos por uma cadeia cinemática aberta, o que significa que são composto por corpos rígidos conectados em série por juntas (Spong, et al., 1989). Tais juntas podem ser, por exemplo, rotacionais ou prismáticas, o que diz respeito ao aspecto construtivo e à classificação do robô.

Os principais objetivos de um manipulador robótico são orientar e posicionar um objeto, mais comumente uma ferramenta, no espaço, com a intenção de executar uma tarefa determinada pelo operador. O ato de posicionar e orientar a ferramenta significa definir parâmetros de posicionamento e orientação, denominados graus de liberdade (GDL), em relação a uma referência, normalmente a base do manipulador.

Para que o robô possa definir o movimento angular de cada junta (variáveis articulares) em relação aos valores de posicionamento e orientação no espaço ortonormal, é necessário que se conheça a modelagem cinemática de determinado manipulador (Rosário, 2005). A modelagem cinemática é feita através da parametrização do manipulador utilizando a notação de *Denavit-Hartenberg*, obtenção de matrizes de transformação que definem posicionamento e orientação entre links e, a partir destas matrizes, a obtenção das equações de cinemática inversa que definem as variáveis articulares para cada junta em relação às variáveis relativas aos graus de liberdade.

A parametrização do manipulador somente pode ser feita se forem conhecidas as especificações básicas do projeto, tais como configuração cinemática, comprimento de juntas e distribuição de graus de liberdade.

2.1. Especificações básicas do manipulador

O projeto do manipulador desenvolvido neste trabalho possui como configuração cinemática o tipo articulado, chamado também de robô antropomórfico. Basicamente, manipuladores articulados são aqueles que possuem três juntas de revolução para posicionamento, codificado como RRR, apresentado pela Figura 2.1.

Manipuladores articulados tem como característica, minimizar a interferência da estrutura na área de trabalho, tornando possível atingir espaços mais confinados, como é mencionado por Craig (1986, p. 235).

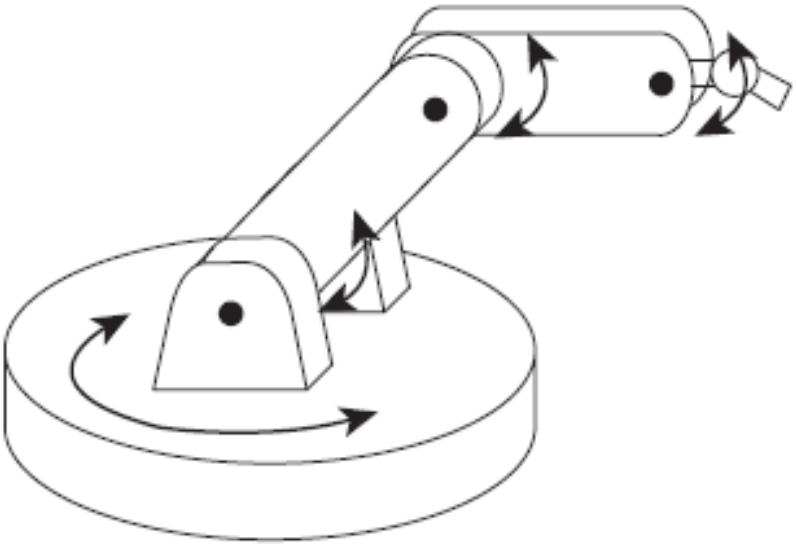

Figura 2.1: Manipulador articulado (Onwubolu, 2005).

O número de graus de liberdade escolhido para o projeto foi seis (três graus de posicionamento e três de orientação), dimensionado de forma a conferir à ferramenta terminal a possibilidade de se posicionar e orientar de qualquer forma no espaço, dentro dos limites de movimentação para cada junta de somente 180°.

O comprimento de cada junta foi escolhido como: 180mm de antebraço, 150mm de braço, 53mm de comprimento da primeira junta do pulso e 47mm de comprimento da segunda junta do pulso. A disposição dos graus de liberdade e a especificação de cada junta pode ser observada na Figura 2.2.

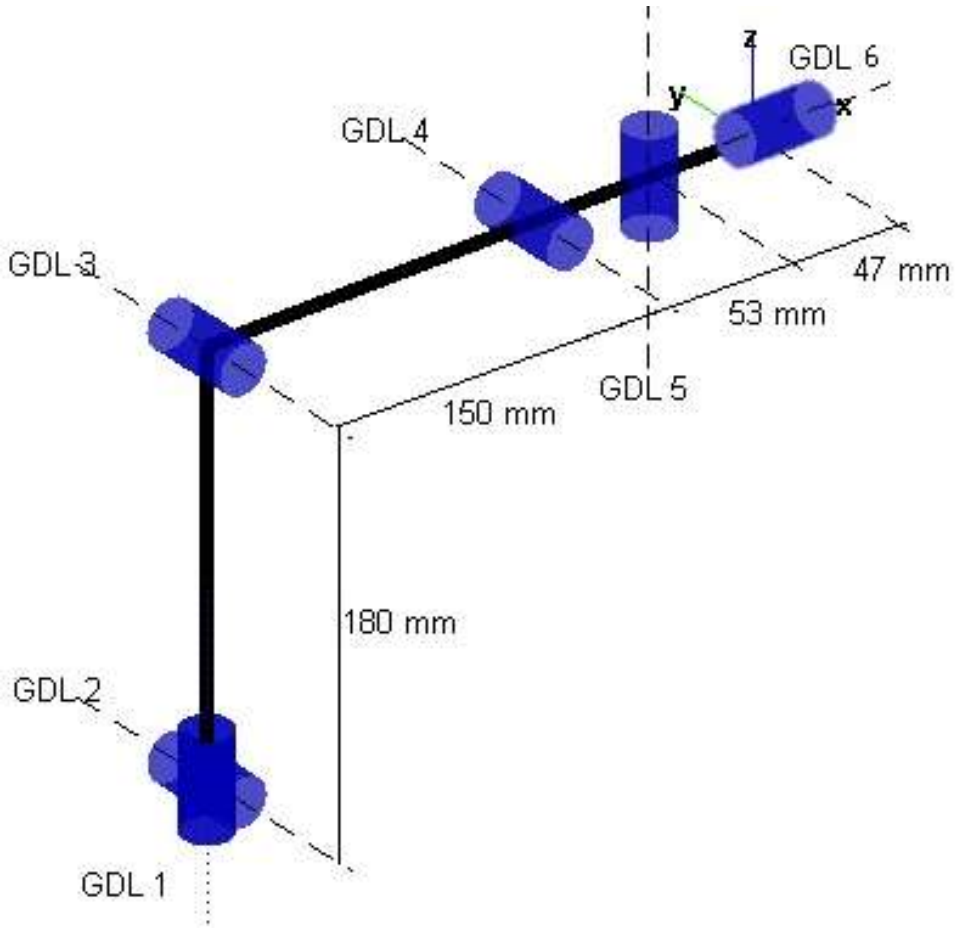

Figura 2.2: Especificação do manipulador.

Os dados apresentados, até então, são referentes somente às especificações básicas do manipulador que são nescessárias para a modelagem cinemática. De posse destes dados é possível desenvolver a parametrização do manipulador em notação específica, como a apresentada a seguir.

2.2. Notação de Denavit-Hartenberg

A notação de *Denavit-Hartenberg* foi desenvolvida para ser uma ferramenta de descrição cinemática de sistemas mecânicos com *n* graus de liberdade (Denavit, 1955). Com a notação de *Denavit-Hartenberg* (DH) é possível definir qualquer eixo de um manipulador em relação aos quatro parâmetros a_i, α_i, d_i e θ_i. Neste trabalho será usada a notação modificada apresentada por Craig (1986, p. 117). Os parâmetros são:

a_i = Distância entre os eixos $\hat{Z}_i$ e $\hat{Z}_{i+1}$ medido através do eixo $\hat{X}_i$;

α_i = Ângulo entre os eixos $\hat{Z}_i$ e $\hat{Z}_{i+1}$ medido através do eixo $\hat{X}_i$;

d_i = Distância entre os eixos $\hat{X}_{i-1}$ e $\hat{X}_i$ medido através do eixo $\hat{Z}_i$;

θ_i = Ângulo entre os eixos $\hat{X}_{i-1}$ e $\hat{X}_i$ medido através do eixo $\hat{Z}_i$.

A Figura 2.3 representa dois eixos de uma cadeia cinemática com os valores dos parâmetros a_i, α_i, d_i e θ_i representados.

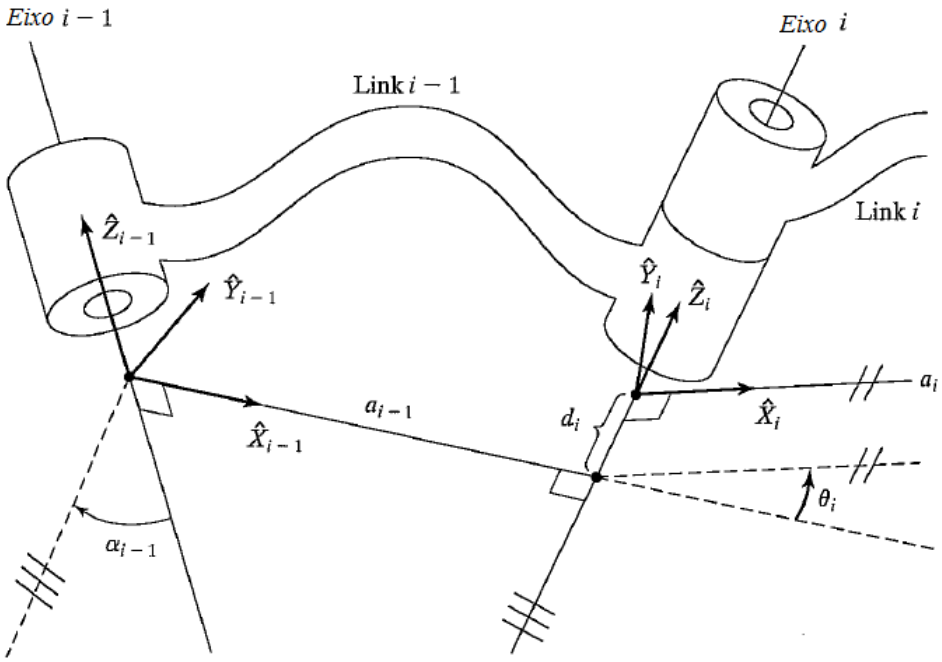

Figura 2.3: Representação dos parâmetros de DH (Craig, 1986)

A Tabela 1 apresenta os valores para os parâmetros de DH definidos para o manipulador projetado:

Tabela 1: Parâmetros de *Denavit-Hartenberg*.

Eixo:	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	$\frac{\pi}{2}$	0	$\frac{\pi}{2}+\theta_2$
3	a	0	0	$-\frac{\pi}{2}+\theta_3$
4	b	0	0	θ_4
5	c	$-\frac{\pi}{2}$	0	θ_5
6	d	0	0	0

Os valores de a, b, c e d são 180mm, 150mm, 53mm e 47mm, respectivamente.

O último grau de liberdade θ_6 (*Roll*) não está sendo representado dentro da notação de DH com o objetivo de simplificação de cálculos, visto que seu valor não influencia nas outras variáveis rotacionais.

Utilizando o pacote *Robotic Toolbox* (Corke, 2002) para *MATLAB* é possível simular a disposição dos eixos do manipulador a partir dos parâmetros de DH. O objetivo desta simulação é observar se os parâmetros realmente caracterizam a estrutura do manipulador, como é possível observar na Figura 2.4.

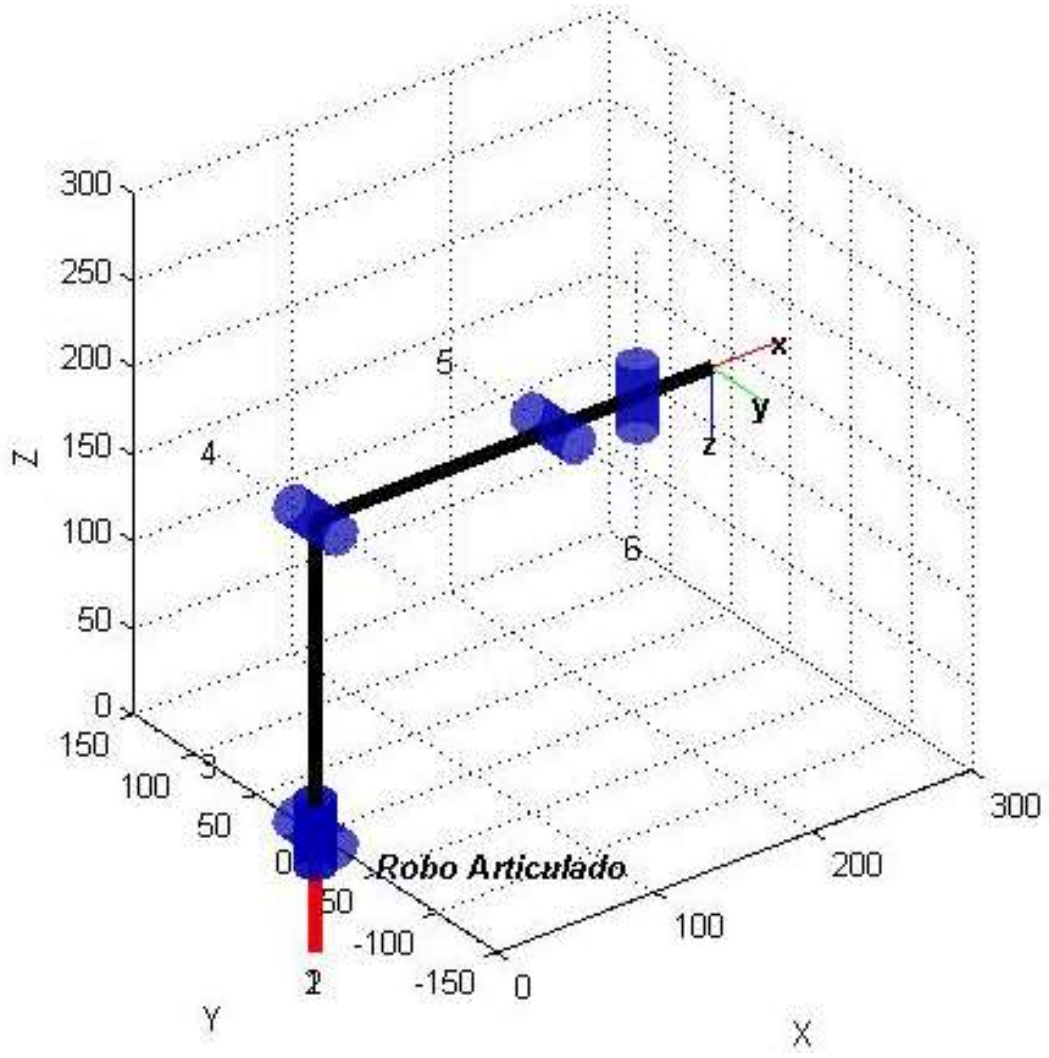

Figura 2.4: Plotagem obtida utilizando *Robotic Toolbox* aplicando os valores de DH (Corke, 2002)

De posse dos valores dos parâmetros de DH, é possível desenvolver as matrizes de transformação homogêneas que definem as variações de um sistema de coordenadas (*frame*) em relação a outro (Craig, 1986). Para isto utiliza-se da Equação (2.1).

$$ {}^{i-1}_{i}T = Screw_X(a_{i-1}, \alpha_{i-1}) Screw_Z(d_i, \theta_i) \qquad (2.1) $$

Onde ${}^{i-1}_{i}T$ representa a matriz de transformação homogênea do sistema de coordenadas i em relação ao sistema $i-1$. Os valores obtidos para cada transformada são apresentados na Equação (2.2). As notações $c\theta_i$ e $s\theta_i$ representam $\cos\theta_i$ e $sen\theta_i$ respectivamente.

$$
{}^{0}_{1}T=\begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}^{1}_{2}T=\begin{bmatrix} -s\theta_2 & -c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{2}_{3}T=\begin{bmatrix} -s\theta_3 & c\theta_3 & 0 & a \\ -c\theta_3 & s\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}^{2}_{3}T=\begin{bmatrix} -s\theta_3 & c\theta_3 & 0 & a \\ -c\theta_3 & s\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}^{3}_{4}T=\begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & b \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.2)
$$

$$
{}^{4}_{5}T=\begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & c \\ 0 & 0 & 1 & 0 \\ -s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}^{5}_{6}T=\begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

A partir das equações de transformação é possível descrever as coordenadas cartesianas em relação às coordenadas angulares (cinemática direta). Em seguida, será apresentado o conceito de como obter valores de coordenadas angulares em relação às coordenadas cartesianas de posicionamento.

2.3. Cinemática inversa

A partir de uma equação de cinemática inversa é possível obter os valores que cada junta de um manipulador precisa atingir para alcançar dada posição cartesiana pretendida pelo operador. Tais equações são de vital importância para o projeto, visto que, estes valores serão convertidos em pulsos para cada atuador do manipulador.

As equações podem ser obtidas a partir de soluções algébricas desenvolvidas por meio das matrizes de transformação obtidas anteriormente, utilizando o método apresentado por Paul, *et al*. (1981).

Temos que:

$$
{}^{0}_{6}T={}^{0}_{1}T\cdot{}^{1}_{2}T\cdot{}^{2}_{3}T\cdot{}^{3}_{4}T\cdot{}^{4}_{5}T\cdot{}^{5}_{6}T \quad (2.3)
$$

Se desenvolvermos a Equação (2.3) colocando a dependência em θ_1:

$$[{}^{0}_{1}T]^{-1} \cdot {}^{0}_{6}T = {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T \quad ((2.4)$$

Equacionando a segunda linha e quarta coluna de ambos os lados da Equação ((2.4), é possível obter que:

$$-s\theta_1 \cdot p_x + c\theta_1 \cdot p_y = d \cdot s\theta_5 \quad ((2.5)$$

Onde as variáveis p_x, p_y e p_z são referentes ao posicionamento ortonormal da ferramenta medidos em relação ao frame da base. Tais valores serão inseridos pelo operador utilizando o terminal de programação para determinar o posicionamento da ferramenta terminal do manipulador.

Para resolver a Equação ((2.5) serão utilizadas as substituições trigonométricas:

$$\begin{aligned} p_X &= \rho \cdot c\phi, \\ p_Y &= \rho \cdot s\phi, \end{aligned} \quad (2.6)$$

Onde:

$$\begin{aligned} \rho &= \sqrt{{p_X}^2 + {p_Y}^2}, \\ \phi &= \tan^{-1}\frac{p_Y}{p_X}. \end{aligned} \quad ((2.7)$$

Substituindo as Equações ((2.7) e ((2.5), é possível obter:

$$c\theta_1 \cdot s\phi + s\theta_1 \cdot c\phi = \frac{d \cdot s\theta_5}{\rho} \quad (2.8)$$

Utilizando a fórmula de diferenças de ângulos é possível obter:

$$s(\phi - \theta_1) = \frac{d \cdot s\theta_5}{\rho} \quad (2.9)$$

A patir disto:

$$c(\phi-\theta_1)=\pm\sqrt{1-\frac{d^2\cdot s^2\theta_5}{\rho^2}} \tag{2.10}$$

Desenvolvendo (2.9) com (2.10) é possível obter:

$$\phi-\theta_1=\tan^{-1}\left(\frac{d\cdot s\theta_5}{\sqrt{{p_x}^2+{p_y}^2-d^2\cdot s^2\theta_5}}\right) \tag{2.10}$$

E, a partir de então, obtem-se:

$$\theta_1=\tan^{-1}\left(\frac{p_y}{p_x}\right)+\tan^{-1}\left(\frac{d\cdot s\theta_5}{\sqrt{{p_x}^2+{p_y}^2-d^2\cdot s^2\theta_5}}\right) \tag{2.11}$$

Para equacionar os próximos ângulos de posicionamento a partir dos valores ortonormais são utilizados os elementos da matriz ((2.4), obtidos pela linha 1, coluna 4, linha 2, coluna 4 e linha 3, coluna 4. Elevando todos estes elementos ao quadrado e somando é possível obter que:

$$-A\cdot s\theta_3+B\cdot c\theta_3=K\,, \tag{2.12}$$

Sendo:

$$\begin{aligned}&A=-b-c\cdot c\theta_4-d\cdot c\theta_4\cdot c\theta_5,\\&B=c\cdot s\theta_4+d\cdot s\theta_4\cdot c\theta_5,\\&K=\frac{a^2+b^2+c^2-p_x-p_y-p_z+2\cdot b\cdot c\cdot c\theta_4+2\cdot c\cdot d\cdot c\theta_5+2\cdot b\cdot d\cdot c\theta_4\cdot c\theta_5}{2\cdot a}\end{aligned} \tag{2.13}$$

A partir da equação (2.12) e utilizando os mesmos métodos de substituições trigonométricas apresentados em (2.6) é possível obter:

$$\theta_3=-\tan^{-1}\left(\frac{c\cdot s\theta_4+d\cdot s\theta_4\cdot c\theta_5}{b+c\cdot c\theta_4+d\cdot c\theta_4\cdot c\theta_5}\right)-\tan^{-1}\left(\frac{K}{\sqrt{\rho^2-K^2}}\right), \tag{2.14}$$

sendo:

$$\rho^2 = d^2 \cdot c^2\theta_5 + b^2 + c^2 + 2 \cdot b \cdot c \cdot c\theta_4 + 2 \cdot c \cdot d \cdot c\theta_5 + 2 \cdot b \cdot d \cdot c\theta_4 \cdot c\theta_5 . \quad (2.15)$$

A partir da equação (2.3) é possível obter:

$$[{}^0_3T]^{-1} \cdot {}^0_6T = {}^3_4T \cdot {}^4_5T \cdot {}^5_6T \quad (2.16)$$

Elevando todos os elementos obtidos pela linha 1, coluna 4, linha 2, coluna 4 e linha 3, coluna 4 da matriz (2.16) ao quadrado e somando é possível obter que:

$$s(\theta_2 + \theta_3) = \frac{p_z \cdot (a \cdot s\theta_3 + b + c \cdot c\theta_4 + d \cdot c\theta_4 \cdot c\theta_5) - (a \cdot c\theta_3 + s\theta_4 \cdot (c + d \cdot c\theta_5)) \cdot (p_x \cdot c\theta_1 + p_y \cdot s\theta_1)}{(p_x \cdot c\theta_1 + p_y \cdot s\theta_1)^2 + p_z{}^2}, \quad (2.17)$$

$$c(\theta_2 + \theta_3) = \frac{p_z \cdot (a \cdot c\theta_3 + s\theta_4 \cdot (c + d \cdot c\theta_5)) + (a \cdot s\theta_3 + b + c \cdot c\theta_4 + d \cdot c\theta_4 \cdot c\theta_5) \cdot (p_x \cdot c\theta_1 + p_y \cdot s\theta_1)}{(p_x \cdot c\theta_1 + p_y \cdot s\theta_1)^2 + p_z{}^2}$$

Desenvolvendo as equações (2.17), é possível obter:

$$\theta_2 = \tan^{-1}\left(\frac{p_z \cdot (a \cdot s\theta_3 + b + c \cdot c\theta_4 + d \cdot c\theta_4 \cdot c\theta_5) - (a \cdot c\theta_3 + s\theta_4 \cdot (c + d \cdot c\theta_5)) \cdot (p_x \cdot c\theta_1 + p_y \cdot s\theta_1)}{p_z \cdot (a \cdot c\theta_3 + s\theta_4 \cdot (c + d \cdot c\theta_5)) + (a \cdot s\theta_3 + b + c \cdot c\theta_4 + d \cdot c\theta_4 \cdot c\theta_5) \cdot (p_x \cdot c\theta_1 + p_y \cdot s\theta_1)} \right) - \theta_3 \quad (2.18)$$

As equações desenvolvidas nesta etapa são referentes somente ao posicionamento do manipulador, dispensando assim o cálculo das equações de cinemática inversa das variáveis angulares de orientação θ_4, θ_5 e θ_6. Os valores de orientação são inseridos manualmente pelo operador utilizando o terminal de programação.

As equações de cinemática inversa obtidas serão utilizadas em um sistema microcontrolado para realizar o treinamento do manipulador. Outro conceito importante, que precisou ser abordado, é o de geração de trajetória que, utilizando das posições gravadas na etapa de treinamento, permite que o manipulador realize movimentos suaves e coordenados.

3. GERAÇÃO DE TRAJETÓRIAS

Uma importante característica de um manipulador industrial é a capacidade de realizar movimentos previamente programados automaticamente. Os métodos matemáticos utilizados para tornar estes movimentos possíveis são chamados de métodos de geração de trajetórias.

A geração de trajetória precisa ser feita de forma que, além de levar o manipulador a sair de uma posição inicial para atingir uma posição final, obedeça a importantes especificações. A primeira especificação vem do fato de que é muito importante, para a estrutura do manipulador, que tais movimentos sejam feitos de forma suave. Isto significa em termos matemáticos que a velocidade inicial ($t = 0$) e final ($t = t_F$) de cada eixo sejam nulas, durante uma trajetória.

Outra importante especificação nesta etapa é a de que o tempo necessário para que cada eixo atinja sua posição final (t_F) seja sempre igual para todos os eixos. Santos *et al*. (2004) propõe uma otimização desta variável de tempo final, de forma a atender as restrições de velocidade e aceleração, determinando assim um tempo mínimo. A técnica escolhida para definição desta variável de tempo neste projeto é a de disponibilizar ao operador que escolha qual deva ser este intervalo de tempo, sem exceder o tempo mínimo. Desta forma, o operador pode definir diretamente a velocidade que a trajetória será efetuada.

O método de geração de trajetória adotado neste trabalho é o de polinômios cúbicos apresentado por Craig (1986, p. 204). Neste método, é desenvolvida uma função para cada variável de posicionamento angular em relação ao tempo. Esta função é desenvolvida de forma a atender as especificações mencionadas anteriormente. A equação cúbica é apresentada pela Equação (3.1).

$$\theta(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3 \tag{3.1}$$

Onde os valores de a_i são apresentados nas Equações (3.2) (Craig, 1986).

$$
\begin{aligned}
a_0 &= \theta_0, \\
a_1 &= 0, \\
a_2 &= \frac{3}{t^2{}_f}(\theta_f - \theta_0), \\
a_3 &= -\frac{2}{t^3{}_f}(\theta_f - \theta_0).
\end{aligned}
\tag{3.2}
$$

De posse de todas as ferramentas matemáticas necessárias, é possível desenvolver um sistema eletrônico que realize o treinamento e a operação em automático do manipulador. O próximo capítulo irá tratar deste desenvolvimento.

4. O CONTROLADOR EMBARCADO

Todo manipulador industrial precisa de um sistema de supervisão e controle. Este sistema tem como função coordenar o ângulo de cada junta fazendo com que o manipulador possa atender aos comandos do operador, sejam estes comandos escritos ou simplesmente guiados por *Joysticks*.

Este projeto propõe como maneira de controlar o manipulador, um sistema embarcado baseado no microcontrolador PIC18f46k22 e que possa ser comandado simplesmente por um controle de vídeo game, adotado neste trabalho como *teach-in pendant*. A ideia de utilizar este controle se justifica pela praticidade de substituição em casos de defeitos, além do fato de ser facilmente encontrado no comércio.

Os principais objetivos do controlador embarcado serão: coletar dados, referentes à posição cartesiana desejada para a junta terminal do manipulador, através do *teach-in pendant*; transformar os valores de posição cartesiana para variáveis angulares e transformar as variáveis angulares em pulsos de acionamento dos servo motores aplicados à cada junta. Além disso, o controlador também ficará encarregado de guardar as posições do manipulador no modo treinamento, para poder acessar estas posições no modo automático.

Este capítulo é destinado a explicar importantes partes para o funcionamento do controlador embarcado. Ao final, é demonstrado o desenvolvimento eletrônico e o esquema básico da programação.

4.1. O microcontrolador PIC18f46k22

Durante a escolha do manipulador para o projeto somente um fator foi importante, velocidade de processamento. Visto que o controlador precisa fazer diversas operações matemáticas em tempo real, somente um processador de alta performance supriria esta necessidade.

Inicialmente, foi utilizado o microcontrolador PIC18F4550 no projeto. Porém, ele não se adequava às necessidades por estar, ao mesmo tempo, subdimensionado e superdimensionado. O PIC18F4550 utiliza de comunicação USB que não está

sendo empregada. Ao mesmo tempo, este microcontrolador não possui velocidade de processamento suficiente.

O microcontrolador PIC18F46k22 da *Microchip*, apresentado na Figura 4.1, se destaca de outros microcontroladores da família 18F pela alta taxa de milhões de instruções por segundo (MIPS), sendo 16 MIPS, apresentado no seu *datasheet* (Microship, 2012). Desta forma, utilizando de paralelismo, o microcontrolador consegue enviar uma alta taxa de informações por segundo, não importando quanto tempo cada instrução realmente leva para ser concluída (Tanenbaum, 1992). Este fator é essencial para o projeto, visto que o microcontrolador irá precisar fazer diversas contas matemáticas em tempo real.

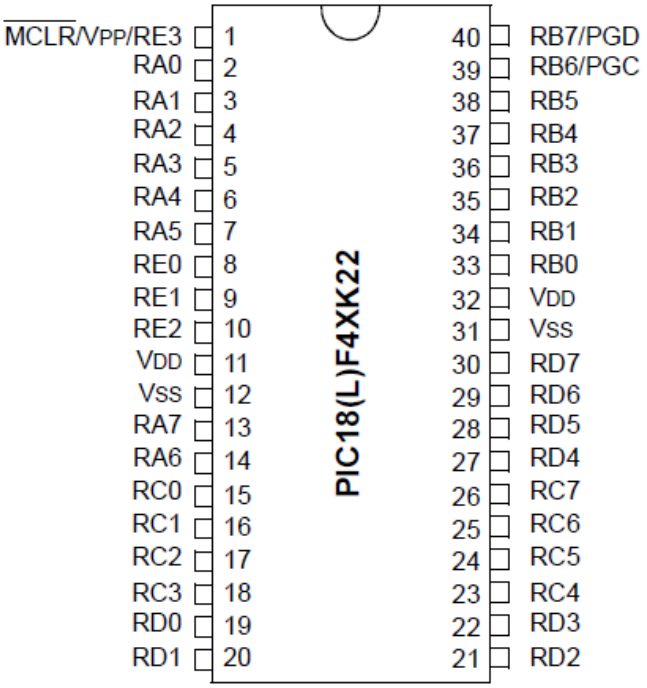

Figura 4.1: Microcontrolador PIC18F46K22 (Microship, 2012)

Outras características também contribuíram para a escolha do microcontrolador, como o alto número de portas de entrada/saída (totalizando 36), timers de 16-bits e memória EEPROM de 1024 Bytes.

No projeto, o microcontrolador fará uso de um timer de 16-bits para o acionamento dos servo motores. Em seguida, será introduzido este funcionamento.

4.2. Acionamento dos servo motores

Os atuadores utilizados neste projeto são todos servo motores de posição, dispositivos de malha fechada controlados por pulso de largura modulada (PWM), como apresentado na Figura 4.2.

O microcontrolador é responsável por gerar pulsos de largura variada. Esta largura influencia diretamente na posição do motor. Os pulsos emitidos são convertidos para tensão pelo próprio circuito do servo motor. Esta tensão alimenta um capacitor a uma taxa constante, se a largura do pulso for modificada, o erro entre o valor novo do pulso e o valor inserido é amplificado e enviado ao motor para produzir movimento (Technologies, 2011).

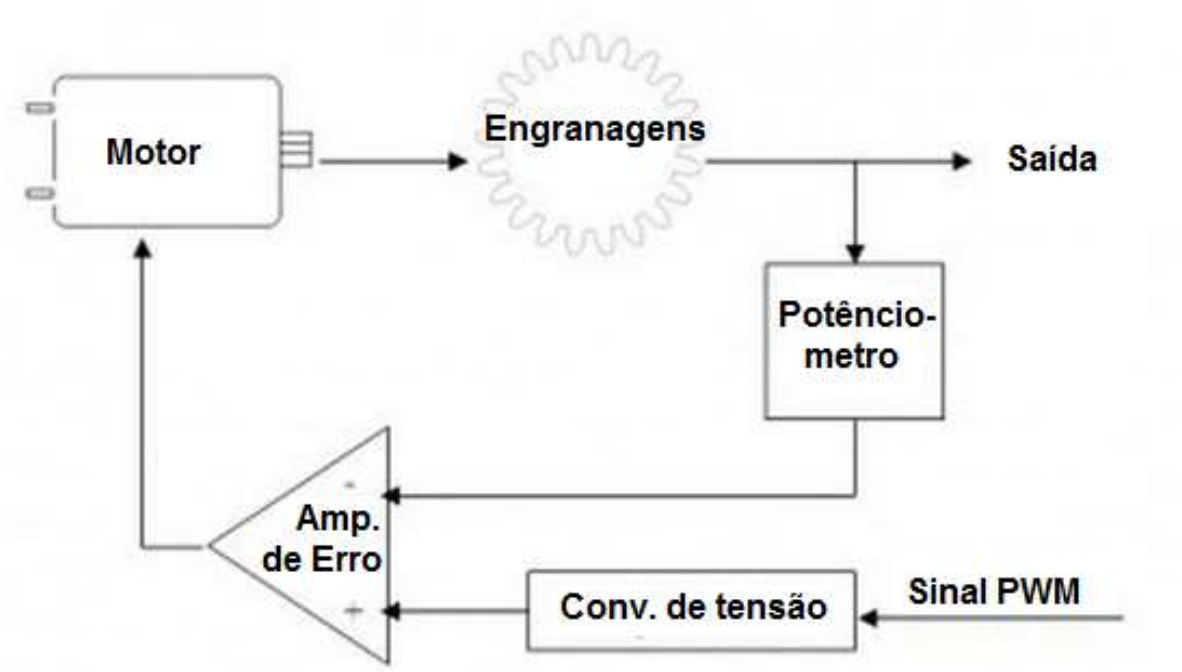

Figura 4.2: Diagrama de controle do servo motor (Technologies, 2011)

O período utilizado para o acionamento dos servo motores é de 20ms, com *duty-cicle* variando dependendo da posição desejada para o servo como apresentado na Figura 4.3. Experimentalmente, foi constatado que os manipuladores de diferentes modelos possuem posicionamentos diferentes para o mesmo *duty-cycle,* porém aproximados.

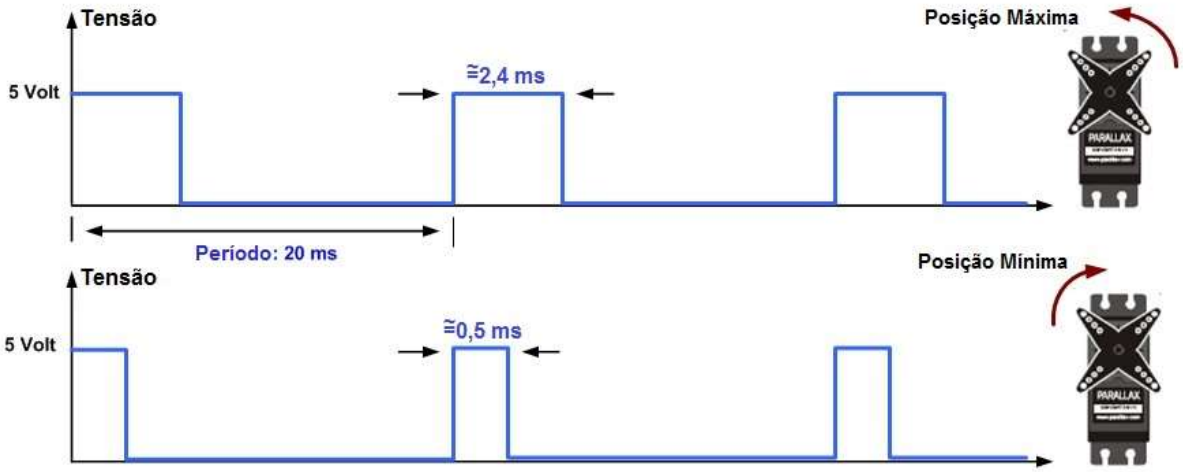

Figura 4.3: Modulação da largura do pulso para posicionamento de servo motor (ErmicroBlog, n.d.)

Utilizando uma equação de linearização, foi possível ajustar a angulação para cada modelo de servo motor, de forma que ele tivesse uma posição mínima e máxima, totalizando uma angulação de 180°. Os servo motores poderiam alcançar faixas maiores de angulação que a de 180°. No entanto, por haver trepidação, foi escolhido limitar esta angulação no programa. A largura do pulso na posição mínima dos servo motores (–90°) é de aproximadamente 500ns, enquanto que na posição máxima (90°) é de 2500ns.

Para que fosse possível regularizar os pulsos com uma frequência de 50Hz foram utilizadas as interrupções por temporização internas dos microcontroladores. É importante que os pulsos obedeçam à especificação de frequência para definir o torque do servo motor, visto que, no caso de estar sendo controlado em uma frequência menor, o motor não “entende” em que posição deve ficar nos intervalos entre pulsos e por este motivo não é ativado.

Esta característica de estar sendo ativado com torque reduzido foi utilizado como estratégia para erguer o manipulador quando ligado. Desta forma, o manipulador é erguido mais lentamente para não comprometer a estrutura com arranques.

4.3. Operação com o *Teach-In Pendant*

A maneira pela qual o operador introduzirá instruções no sistema, escolhida para este projeto, foi adotar o controle, comercialmente vendido para jogos eletrônicos no console *play-station*, como *Teach-in Pendant* (TP). O controle conta com uma variedade muito grande de botões e a disposição de dois joysticks, cada um com uma faixa de variação de 2 Bytes, um para a vertical e outro para a horizontal.

Visto que, o objetivo em se utilizar o TP é poder manipular os graus de liberdade de posicionamento e orientação, seriam necessários doze botões para operar o avanço e recuo de cada variável atribuída. Utilizando os *joysticks*, é possível controlar os três graus de liberdade destinados ao posicionamento utilizando os bytes horizontal e vertical do primeiro *joystick*, junto como o vertical do segundo. Os outros três graus de liberdade, destinados à orientação da ferramenta terminal, são operados utilizando o byte de comando horizontal do segundo *joystick*, junto com os botões direcionais verticais e horizontais.

Outros botões do TP são utilizados para funções específicas do controlador como modificar o formato treinamento/automático e reiniciar o sistema. A Figura 4.4 apresenta o controle com os botões e funcionalidades associadas.

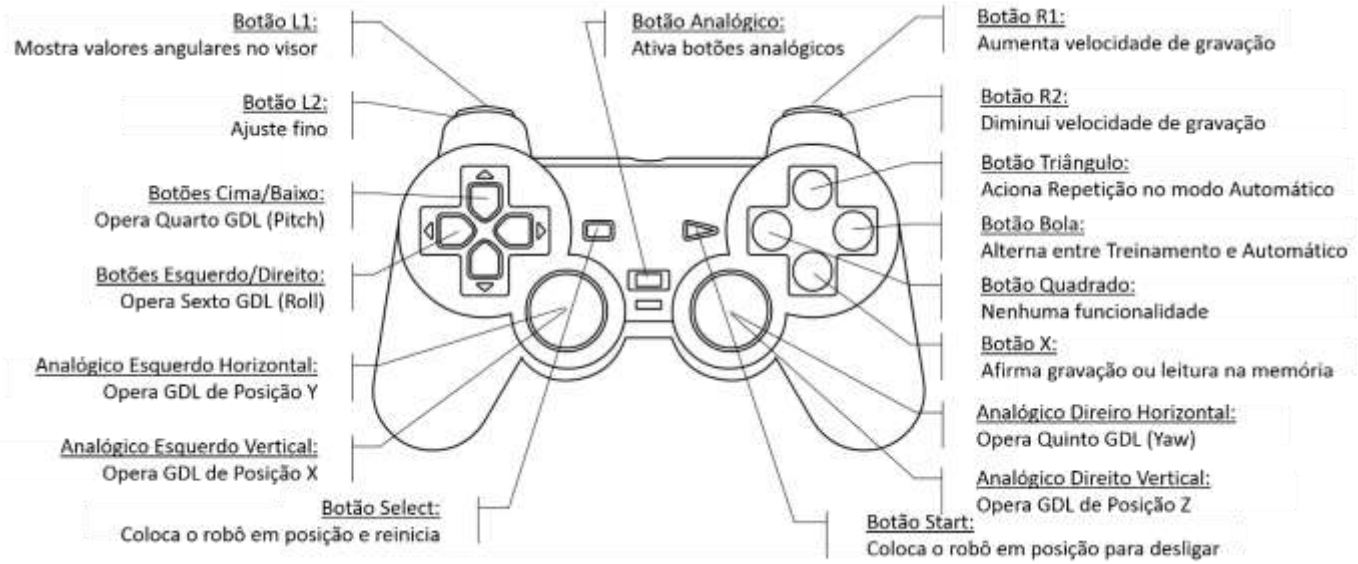

Figura 4.4: Botões e funcionalidades do controle

O controle comunica com o microcontrolador por interface periférica serial (SPI), utilizando um protocolo próprio de aquisição de dados.

4.4. Desenvolvimento eletrônico

O microcontrolador é o componente central do controlador embarcado desenvolvido para o projeto, dentro dele são feitas todas as operações matemáticas e armazenamento de dados. Além do microcontrolador, outros componentes foram escolhidos para funções específicas do sistema, é o caso dos transistores NPN, do visor de LCD e do cristal de oscilação.

Os transistores NPN, BC 548B, foram aplicados ao sistema eletrônico para oferecer uma amplificação de corrente na saída do microcontrolador para os servo motores durante uma determinada etapa do trabalho. Porém, a partir de testes feitos utilizando o osciloscópio pôde ser constatado que a presença dos transistores modificava o sinal, como pode ser observado na Figura 4.5, e que esta modificação tinha influência na estabilidade dos motores. Por este motivo foi feita a escolha de retirar os transistores do sistema.

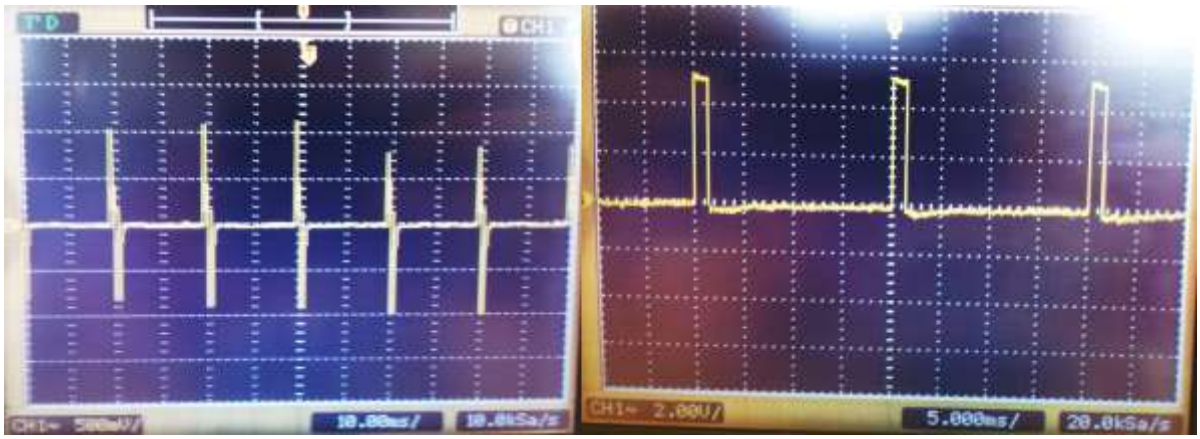

Figura 4.5: PWM com e sem a utilização dos transistores NPN

Outro importante componente no sistema é o visor de LCD, por oferecer interface que informa ao operador a posição cartesiana da ferramenta terminal em relação à base em milímetros, como mostrado na Figura 4.6, além de mostrar os ângulos de orientação.

Figura 4.6: Visor de LCD

Um cristal de oscilação externo de 20MHz foi escolhido para maximizar a frequência interna do sistema.

A fonte de alimentação necessária para o sistema precisou fornecer uma tensão de 5V e uma corrente de pelo menos 2A, visto que os atuadores utilizados precisam de uma corrente alta para operarem. Sendo assim, foi adotada uma fonte de alimentação utilizada para computadores que são capazes de fornecer 5V e até 30A.

Foram feitas, inicialmente, simulações computadorizadas do sistema eletrônico apresentado. Em seguida, foi montado e testado o sistema em *protoboard* e confeccionada placa de circuito impresso. As faces do circuito impresso confeccionado pode ser observado na Figura 4.7.

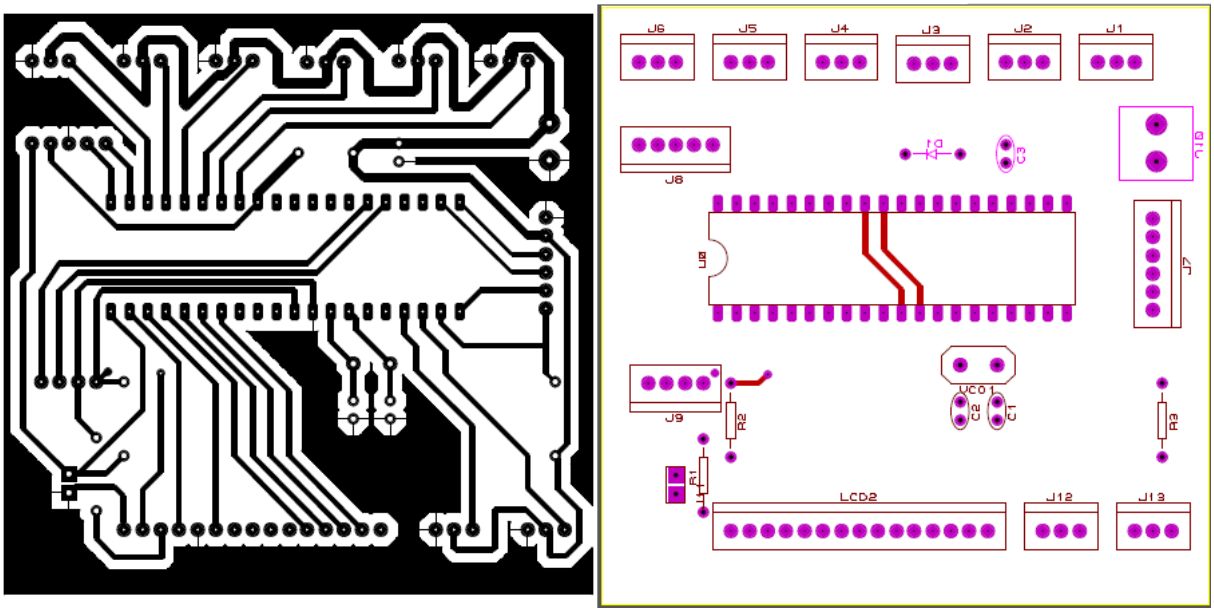

Figura 4.7: Circuito impresso confeccionado

Após completada a descrição sobre o desenvolvimento eletrônico do controlador embarcado será abordada a parte da programação do sistema.

4.5. Programação

Todas as operações a serem realizadas pelo microcontrolador precisam, inicialmente, serem codificadas e embutidas no sistema. Para este projeto, toda a programação foi feita em linguagem C, por ser uma linguagem de alto nível e com funções de fácil utilização.

As principais funções do programa do microcontrolador neste projeto são: adquirir os dados do TP; converter estes dados de coordenadas cartesianas para coordenadas articulares; converter os valores de coordenadas articulares para valores referentes à largura do pulso; atualizar os valores de largura de pulso que irão para a saída destinada ao servo motor, em intervalos regulares; realizar operações referentes à geração de trajetória, quando em modo automático, e atualizar os valores de LCD.

Ao adquirir os dados pelo TP, o microcontrolador necessita enviar comandos e receber dados de acordo com um padrão pré-definido pelo fabricante do modelo do controle (protocolo) apresentado por (Inventor, n.d.). Os dados recebidos pelos *joysticks* destinados aos graus de liberdade de posicionamento são convertidos para variáveis referentes ao posicionamento carteziano, em milimetros, utilizando uma equação de linearização respondendo ao posicionamento do *joystick*, como é apresentado na Figura 4.8.

```
185             /*============ Joypad Esquerdo X = Y ==================== */
186
187 if(ps_anLx>=0  && ps_anLx<=255)
188 {
189    y_passo = (2*passo/256)*ps_anLx - passo; // 10 em 256; 0 em 128; -10 em 0
190    y = y + y_passo;
191    if(y>=quad_max_y)
192    {
193       y=quad_max_y;
194    }
195    if(y<=quad_min_y)
196    {
197       y=quad_min_y;
198    }
199 }
```

Figura 4.8: Fragmento do programa, captura de valores do eixo Y pelo joypad esquerdo

Além disto, o programa ainda recolhe os dados referentes à outros botões do TP e trata estes valores, como por exemplo, os GDL referentes à orientação e outras funções como reinicialização e gravação.

Os dados obtidos pelos GDL de posicionamento e orientação são utilizados nas equações matemáticas de cinemática inversa como pode ser visto na Figura 4.9. Para poder realizar estas equações, é utilizada uma biblioteca própria que possui funções matemáticas necessárias como tangente, seno e cosseno.

```
582 c4 = cos(angulo[3]);
583 s4 = sin(angulo[3]);
584 c5 = cos(angulo[4]);
585 s5 = sin(angulo[4]);
586 K0 = sqrt(abs(x*x+y*y-d*d*s5*s5));
587 angulo[0] = atan2(y,x)+atan2(d*s5,K0);
588 K1 = (a*a+b*b+c*c+d*d-x*x-y*y-z*z+2*b*c*c4+2*c*d*c5+2*b*d*c4*c5)/(2*a);
589 K2 = d*d*c5*c5+b*b+c*c+2*b*c*c4+2*c*d*c5+2*b*d*c4*c5;
590 angulo[2] = -atan2(c*s4+d*s4*c5,b+c*c4+d*c4*c5) + atan2(-K1,sqrt(K2-K1*K1));
591 c1 = cos(angulo[0]);
592 s1 = sin(angulo[0]);
593 c3 = cos(angulo[2]);
594 s3 = sin(angulo[2]);
595 K3 = z*(a*s3+b+c*c4+d*c4*c5)-(a*c3+s4*(c+d*c5))*(x*c1+y*s1);
596 K4 = z*(a*c3+s4*(c+d*c5))+(a*s3+b+c*c4+d*c5*c4)*(x*c1+y*s1);
597 angulo[1] = atan2(K3,K4)-angulo[2];
```

Figura 4.9: Fragmento do programa, equações de cinemática inversa

Os valores recebidos pelas equações de cinemática inversa são utilizados em equações de linearização para se obter os valores de largura de pulso para cada servo

motor, como pode ser observado na Figura 4.10. Cada modelo de servo motor possui uma equação de linearização diferente.

```
634       /*============ Atualiza valores para motores  =================== */
635
636       ton[0] =   8.744*ang[0]+1420; //MG995
637       ton[1] =   8.055*ang[1]+1375; //MG996R
638       ton[2] =   8.055*ang[2]+1500; //MG996R
639       ton[3] =  -8.744*ang[3]+1444; //MG995
640       ton[4] =  10.522*ang[4]+1447; //SG90
641       ton[5] =  10.522*ang[5]+1447; //SG90
642       ton[6] =  10.522*ang[6]+1447; //SG90
643       ton[7] =  -8.055*ang[1]+1455; //MG995 Inverso
```

Figura 4.10: Fragmento do programa com equações de linearização com os pulsos

Os botões R1 e R2 do TP são responsavéis por alterar os valores referentes à velocidade em que será programada a trajetória. Os valores variam de um à cinco, sendo um o mais lento e cinco o mais rápido, e estão ligados aos valores de tempo nescessários para a realização da trajetória e, indiretamente, ligados à velocidade do movimento.

Se setado o valor do bit para ‘escolher opção’, obtido pelo TP, no modo de ‘treinamento’, o programa entra em uma rotina de gravação dos valores para cada uma das seis variáveis de 1 byte referentes aos graus de liberdade, junto com uma variável de 1 byte referente à velocidade desejada para a trajetória e uma variável de 1 byte referente à posição da ferramenta, tudo na memória EEPROM do microcontrolador de uma forma fácil de ser acessada no modo automático, como apresentado no fragmento da Figura 4.11.

```
438  /*==============  Botão X - Grava posição ==================== */
439  if(!bit_test(ps_dig2,6)&& n_mgrav==1){flagXg = 1;} // Botão X
440  if(bit_test(ps_dig2,6) && n_mgrav==1 && flagXg == 1)// Grava posição no modo leitura
441  {
442     flagXg = 0;
443     for(i=0;i<7;i++)
444     {
445        ang_m[i]= ang[i]+90;
446     }
447     for(i=0;i<7;i++)
448     {
449        write_eeprom((n_mem-1)*8+i+1,ang_m[i]);
450     }
451     write_eeprom(n_mem*8,vel);
452     n_mem++;
453     if(n_mem>128)n_mem=1;
454     n_mem_f = n_mem;
455  }
```

Figura 4.11: Fragmento do programa, gravação no modo treinamento

Se caso setado o valor do bit de ‘escolher opção’ no modo ‘automático’, o programa entra em uma rotina de leitura da memória EEPROM do microcontrolador. A rotina inicia carregando os valores gravados referentes aos 6 GDL da posição desejada, a posição da garra e a variável referente ao intervalo de tempo desejado. Tais valores são utilizados em uma equação de geração de trajetória para cada junta, resultando em um movimento suave e coordenado.

O modo como são acionados os atuadores é feito através de um temporizador que seta os bits de saída para os servo motores em intervalos regurares de 50Hz. O processo para gerar o pulso é feito de forma que, inicalmente, as saídas são setadas, em seguida é gerado um atraso, que depende da posição angular desejada para servo motor. E em seguida são reiniciados. Este procedimento é feito de forma a simular um sinal de PWM adequado para os motores.

Uma preocupação durante a etapa de programação foi que o programa nunca faça movimentos que possam prejudicar a estrutura do manipulador. Para isto, foi feita uma rotina em que, ao ligar o manipulador, ele seja setado com uma frequência de PWM reduzida, de forma que, não importando de qual posição ele foi deixado antes de ser desligado, o robô possa atingir a posição inicial de forma suave. Também com a preocupação de não prejudicar a estrutura, foi implementada uma rotina de movimentação suave utilizando equações de geração de trajetória nas ocasiões de reiniciar o manipulador, preparar o manipulador para ser desligado e quando é modificado a forma de operação entre automático e treinamento.

O visor LCD é atualizado a cada interação utilizando uma biblioteca própria para trabalhar com tal dispositivo. Os valores que aparecem são: o formato (Treinamento/Automático), a variável referente ao ajuste de velocidade escolhida para o movimento em automático, a variável referente à posição de memória em que está sendo gravado os valores de posicionamento do manipulador e, na segunda linha, os valores cartezianos de posicionamento do manipulador. Se pressionado o botão L1 no TP, é possível observar, pelo LCD, os valores angulares de cada um dos atuadores.

O programa desenvolvido tem funcionamento satisfatório. Os resultados obtidos serão abordados ao final deste trabalho. Um fluxograma foi preparado com a intenção de facilitar o compreendimento da lógica de programação como na Figura 4.12.

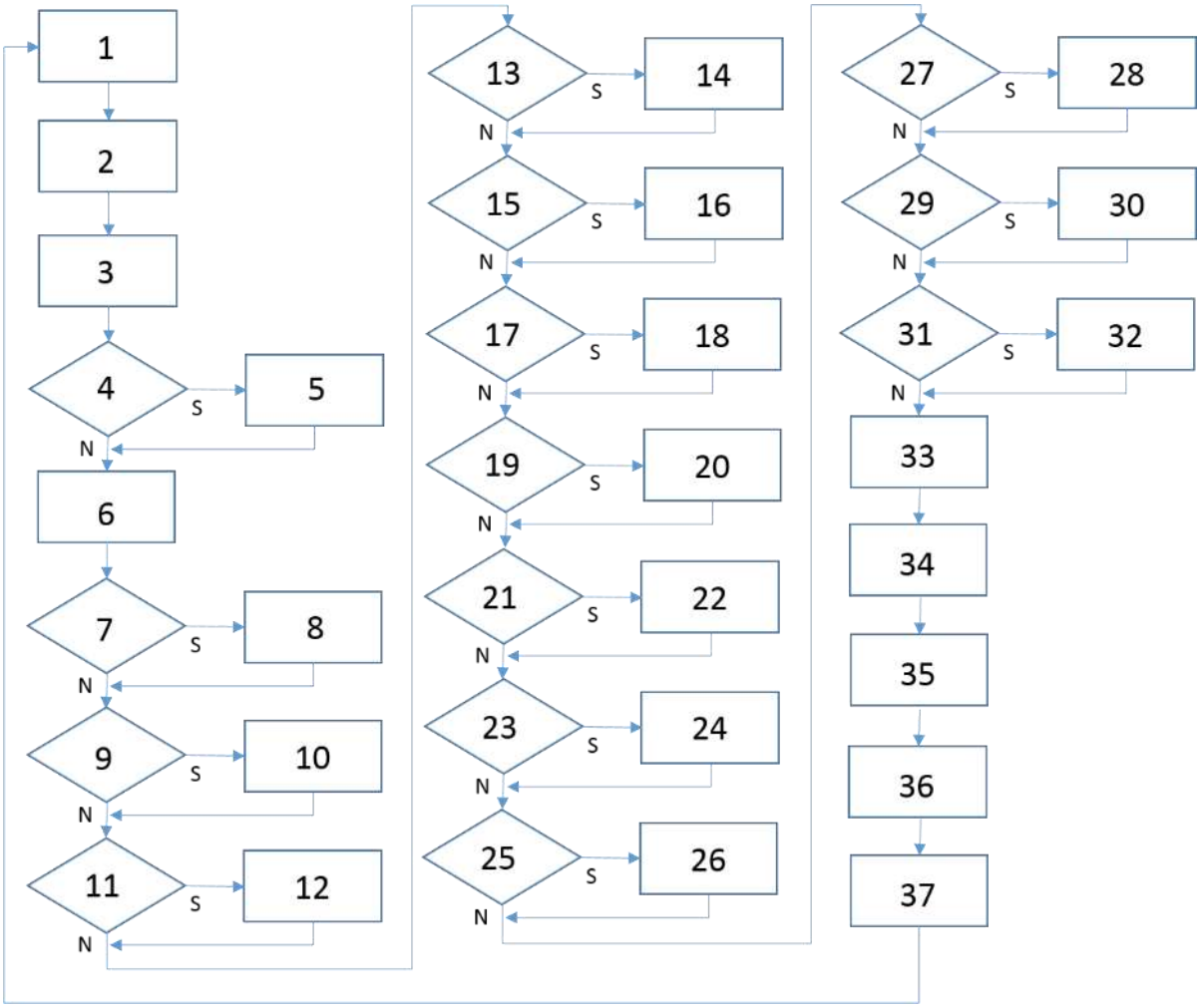

Figura 4.12: Fluxograma do programa

1. Declarações iniciais.
2. Iniciar robô alterando a frequência dos pulsos, inicialmente com nenhum pulso, depois com frequência baixa até estabilizar em 50 MHz.
3. Recebe valores novos do teach in pendant.
4. Verifica botão L2.
5. Passo pequeno para ajuste fino.
6. Passo normal.
7. Verifica analógico esquerdo horizontal.
8. Modifica valor de Y, tratando para não passar dos limites máximos e mínimos.
9. Verifica analógico esquerdo vertical.
10. Modifica valor de X, tratando para não passar dos limites máximos e mínimos.
11. Verifica analógico direito vertical.
21. Verifica botão triângulo.
22. Ativa modo de repetição.
23. Verifica botão R2.
24. Diminui variável referente à velocidade.
25. Verifica botão R1.
26. Aumenta variável referente à velocidade.
27. Verifica botão Bola.
28. Alterna entre modo treinamento e automático, mudando valores mostrados no display.
29. Verifica botão X no modo treinamento.
30. Grava valores de posicionamento e velocidade na memória.
31. Verifica botão X no modo leitura.
32. Lê valores da memória e inicia rotina de geração de trajetória para realizar todos os

12. Modifica valor de Z, tratando para não passar dos limites máximos e mínimos.
13. Verifica analógico direito horizontal.
14. Modifica valor do quinto GDL, tratando para não passar dos limites máximos e mínimos.
15. Verifica botões cima/baixo.
16. Modifica valor do quarto GDL, tratando para não passar dos limites máximos e mínimos.
17. Verifica botão Select.
18. Inicia rotina de levar o robô para uma posição propícia para ser desligado e reinicia.
19. Verifica botão Start.
20. Inicia rotina de levar o robô para uma posição propícia para ser desligado e retorna para modo treinamento.
33. Se no modo treinamento, voltando diretamente do modo automático, inicia rotina para retornar à posição inicial.
34. Utiliza equações de cinemática inversa para determinar valores angulares.
35. Trata valores para não passar dos valores máximos.
36. Atualiza variáveis referentes ao tempo em que o pulso passa ativo para o acionamento de cada motor dependendo do modelo do servo motor.
37. Atualiza display de LCD.

A próxima etapa de desenvolvimento deste projeto é a de desenvolvimento mecânico.

5. DESENVOLVIMENTO MECÂNICO

O desenvolvimento mecânico foi feito respeitando limitações dos atuadores e a estrutura do manipulador. Inicialmente, foram cotados materiais baseando-se em aspectos escolhidos como importantes para o projeto. Logo em seguida, foi desenvolvida a estrutura do manipulador, baseada em um modelo obtido em plataformas de projetos em CAD.

Os materiais e a estrutura escolhidos são avaliados a partir de cálculos de momento com a intensão de descobrir os valores máximos de força que o manipulador poderia suportar.

5.1. Materiais

Para a montagem física do manipulador, alguns aspectos foram destacados como mais importantes. São estes o preço, resistência, suavidade de movimentação e precisão de posicionamento. Para isto, foram eleitos alguns materiais como mais cotados para este projeto.

Painéis de fibra de madeira de média densidade (MDF) foram selecionados para a estrutura por possuir resistência à tensão e compressão suficiente para a aplicação, 18 MPa e 10 MPa respectivamente (MakeItFrom, 2009). Estes valores são mais que suficientes para suportar o peso da própria estrutura. Porém, se sobrecarregado a estrutura pode ceder. Outra vantagem é que estes painéis possuem um custo baixo e são muito práticos de serem cortados.

A razão pela qual foi escolhida a fibra de madeira de média densidade (MDF) ao invés de acrílico, material empregado em diversos outros projetos de desenvolvimento de manipulador semelhantes, como por exemplo o projeto desenvolvido em laboratório pelo grupo de estudo de robótica do CEFET-MG do campus de Divinópolis (GER), é de que o acrílico possui uma densidade (e consequentemente peso) consideravelmente superior à do MDF, 1,17 g/cm^3 comparado com 0,75 g/cm^3 do painel de fibra de madeira de média densidade (MDF) (MakeItFrom, 2009). É fato que o acrílico possui um valor de tensão de ruptura muito superior ao do MDF (70 Mpa comparado com 18 Mpa do MDF) (MakeItFrom, 2009).

Porém o robô projetado não está sendo dimensionado para suportar um peso muito grande, visto que sua capacidade está restrita à capacidade de seus atuadores, e por esta razão não é necessária a utilização de um material com uma resistência muito alta.

Os atuadores escolhidos são servo motores de posição. Destes, tem-se cinco servo motores de alto torque, que estão distribuídos em: um Tower Pro MG-996R na base, dois Tower Pro MG-996R no antebraço, um Tower Pro MG-996R no braço, um Tower Pro MG-995 na primeira junta do pulso, e dois micro servos Tower Pro SG90 nos últimos dois graus de liberdade. As especificações de torque e de peso podem ser observadas na Tabela 2. Os servos foram escolhidos levando em conta a massa, o torque e a precisão.

Tabela 2: Especificação dos atuadores

	Torque	Massa
Tower Pro MG-996R	1,1 N.m	55 g
Tower Pro MG-996R	1,1 N.m	55 g
Tower Pro SG90	0,18 N.m	9 g

Outra peça escalada para o projeto foi o uso de rolamentos radiais SKF 623 apresentado na Figura 5.1, por fornecer estabilidade e suavidade aos movimentos nos eixos. Os rolamentos axiais SKF 51116, mostrados na Figura 5.2, foram aplicados à base para suavizar a movimentação do primeiro grau de liberdade.

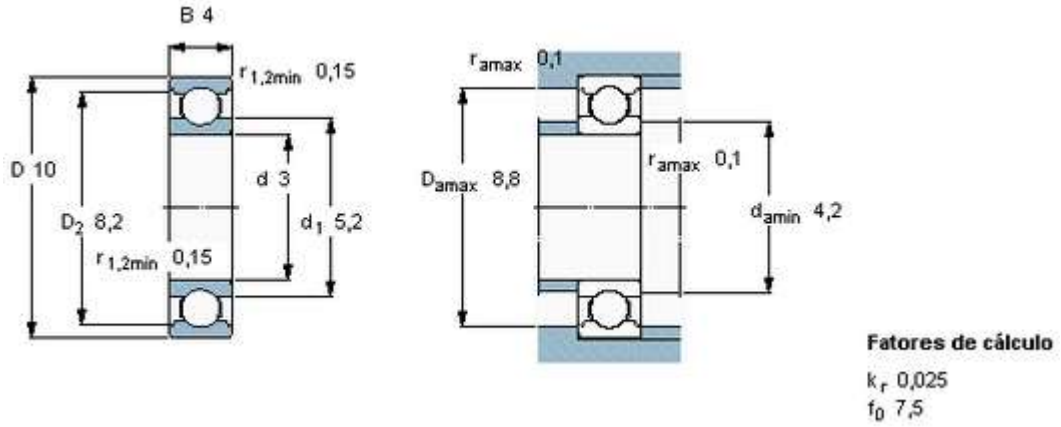

Figura 5.1: Rolamentos radiais SKF 623 (SKF, 2013)

novo modelo	SKF 51116 rolamentos
modelo antigo	8116 rolamentos
tipos de, Categorias	Rolamentos axiais de esferas
marca de	SKF rolamentos
diâmetro interno (d)	80 mm
diâmetro exterior (D)	105 mm
espessura (B)	19 mm

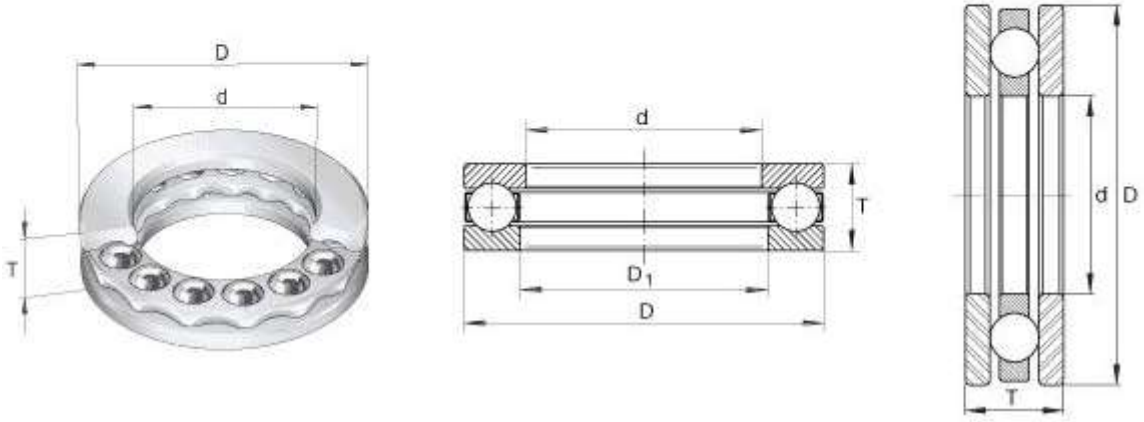

Figura 5.2: Rolamentos axiais SKF 51116 (SKF, 2013)

5.2. Projeto

O projeto mecânico foi inteiramente desenvolvido utilizando ferramentas de CAD em 3D. Esta ferramenta foi escolhida por oferecer recursos visuais que facilitam o desenvolvimento mecânico do projeto. Além disso, é possível gerar planilhas em desenho técnico com bastante praticidade. O modelo pronto feito em CAD 3D pode ser visto pela Figura 5.3.

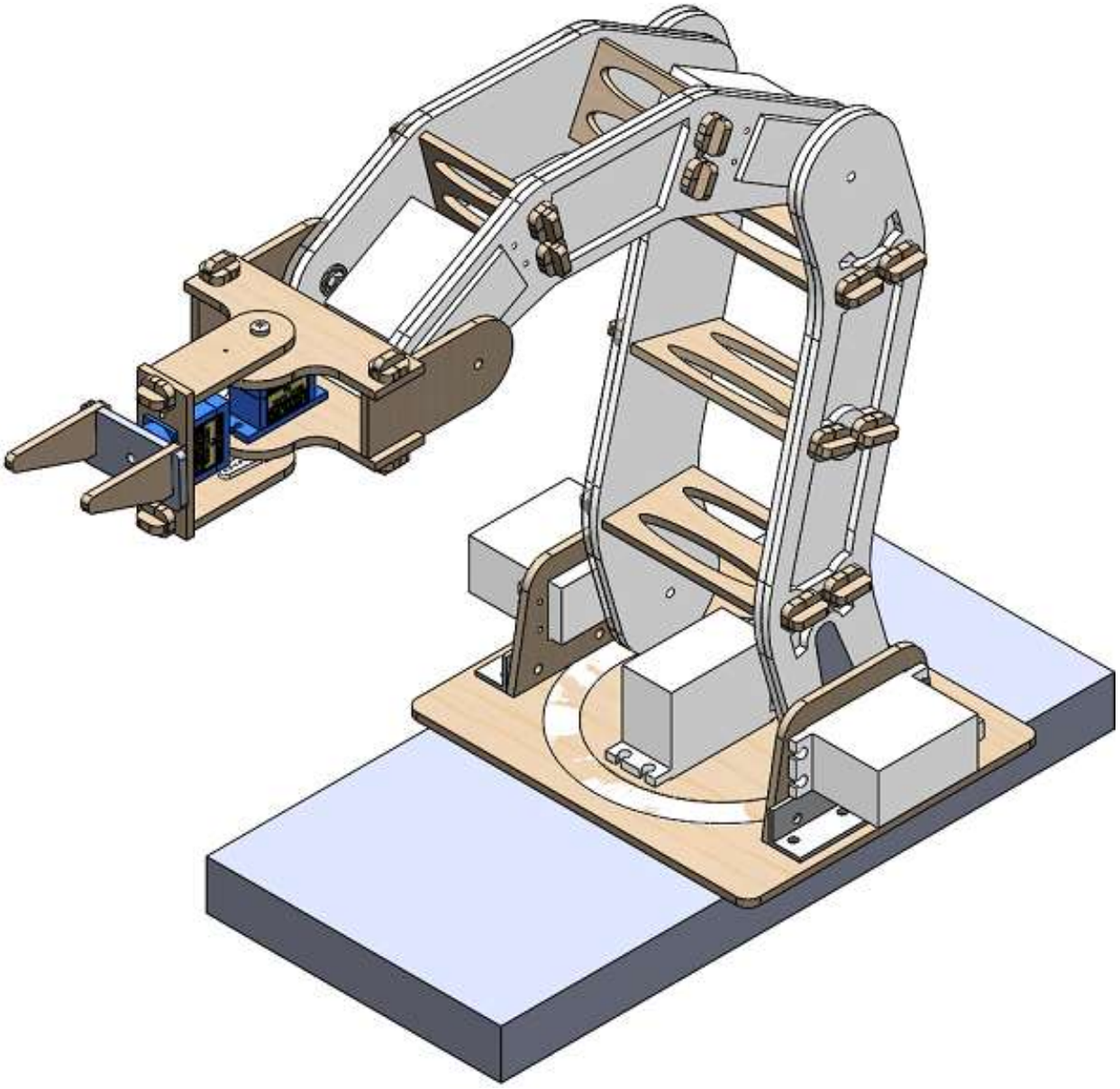

Figura 5.3: Modelo do manipulador desenvolvido em CAD 3D

Grande parte do projeto mecânico desenvolvido foi baseando em um modelo obtido em (GrabCAD, 2012). Diversos detalhes foram adicionados, como o uso de chapas perpendiculares às estruturas dos braços com a intensão de realizar o espaçamento entre as estruturas. Os lados da peça são presos às estruturas por presilhas, como apresentado na Figura 5.4.

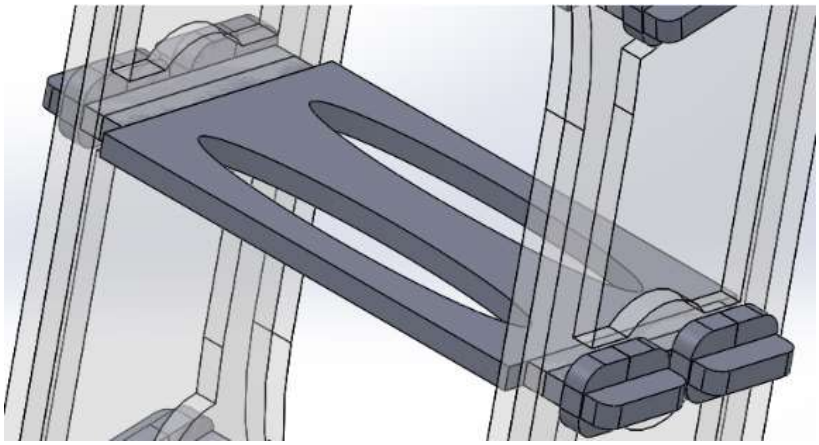

Figura 5.4: Detalhe, chapa espaçadora

Outra modificação feita no projeto foi a inclusão de uma base de nylon fresada com a utilização da fresadora CNC presente no laboratório de robótica do CEFET-MG, campus Divinópolis. A base é apresentada na Figura 5.5. A partir da base foram adicionados os rolamentos e os atuadores responsáveis pela movimentação do primeiro grau de liberdade.

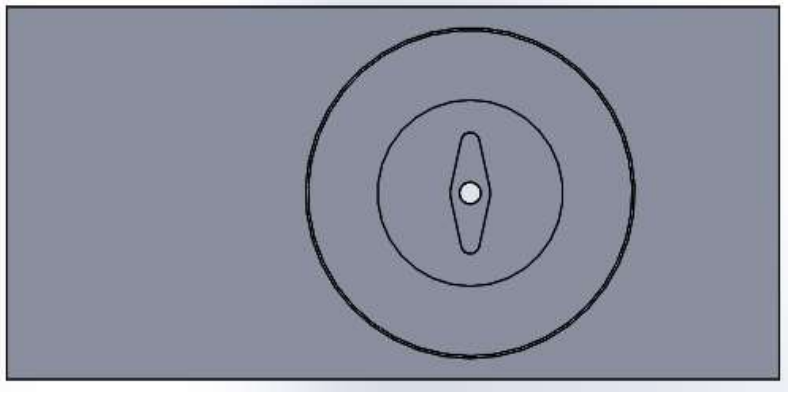

Figura 5.5: Detalhe, base de nylon

Os rolamentos SKF 623 adicionados às juntas foram acompanhados de parafusos de 3mm e porcas para oferecer fixação e estabilidade nos eixos, como pode ser observado pela Figura 5.6.

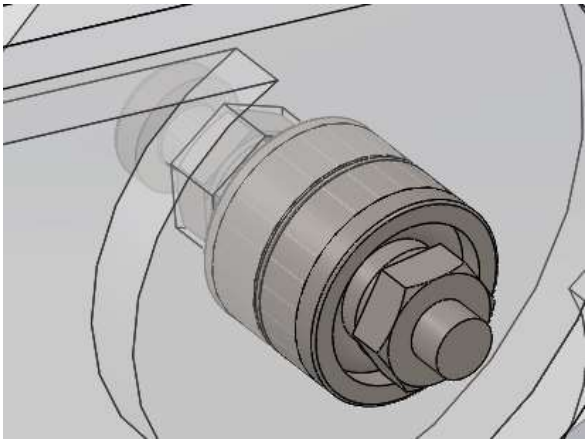

Figura 5.6: Detalhe, fixação das juntas com parafusos e porcas

Utilizando o modelo feito em CAD em 3D foi possível gerar planilhas que foram utilizadas para fazer os cortes nos painéis de madeira. A planilha pode ser observada resumidamente pela Figura 5.7.

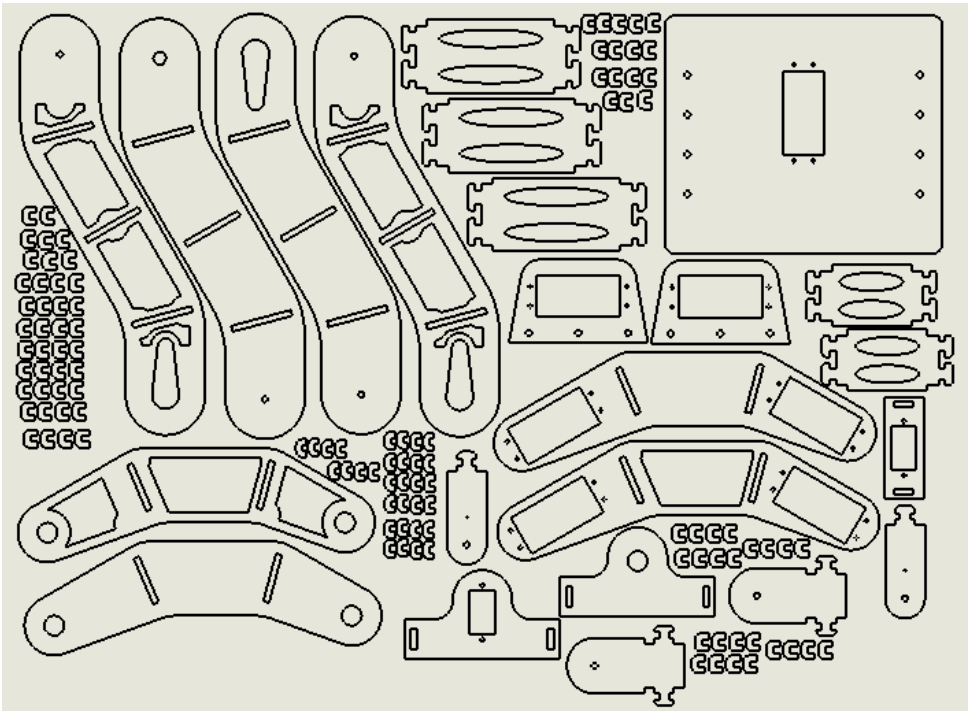

Figura 5.7: Planilha para corte à laser gerada

Tendo desenvolvido o projeto mecânico da estrutura do manipulador junto com a escalação dos materiais, foram são realizados, em seguida, cálculos para determinar se o torque dos motores que estão sendo utilizados é capaz de suportar a estrutura.

5.3. Cálculos de torque

Utilizando equações do somatório dos momentos e a partir das limitações dos atuadores em relação ao torque, pretende-se descobrir o valor máximo de força que o sistema é capaz de suportar em uma situação crítica como é o caso do robô completamente esticado.

Para isto foi realizado a simplificação de que o peso de cada parte do manipulador está sendo aplicado completamente sobre o meio.

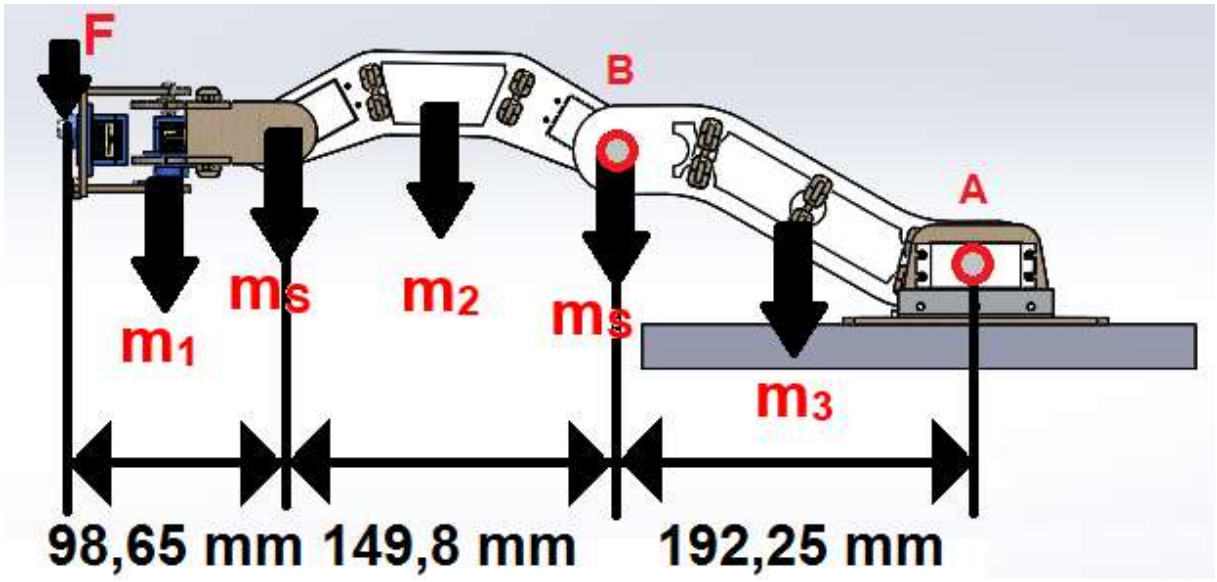

Figura 5.8: Representação da estrutura do manipulador

A partir do modelo em CAD, foi possível obter os valores de massa para cada parte do manipulador, considerando uma densidade para o MDF de 950Kg.m^{-3}. Os valores de massa representados na Figura 5.8 podem ser observados pela Tabela 3.

Tabela 3: Valores de massa para cada parte

m_1	43,77g
m_2	60,26g
m_3	102,46g
m_s	55g

Considerando que cada servo motor é capaz de suportar um torque máximo de 1,1N.m e que no eixo do antebraço estão sendo utilizados dois motores é possível aplicar a equação de somatório dos momentos no ponto A.

$$\sum M_A = 2{,}2N.m \tag{5.1}$$

Desenvolvendo esta equação é possível obter a equação (5.2), em função da força aplicada F.

$$F \cdot 440{,}7 \cdot 10^{-3} + (m_1 \cdot 391{,}375 + m_S \cdot 342{,}05 + m_2 \cdot 267{,}15 + m_S \cdot 192{,}25 + m_3 \cdot 96{,}125) \cdot 9{,}8 \cdot 10^{-6} = 2{,}2N.m \tag{5.2}$$

Aplicando os valores de massa e desenvolvendo em função da força F é possível observar que:

$$F = 3{,}381N \tag{5.3}$$

Através do desenvolvimento dos cálculos é possível observar que o valor máximo de força que é pode ser aplicado na extremidade do manipulador estando na posição de completamente esticado apresentado na Figura 5.8 é de 3,381N, o que é um valor satisfatório.

É possível também calcular o somatório do momento aplicado no ponto B para avaliar a resistência do atuador no eixo do terceiro grau de liberdade (braço). Considerando que o servo motor é capaz de realizar um torque de até 1,1N.m, desenvolve-se a equação (5.4).

$$\sum M_B = 1{,}1N.m \tag{5.4}$$

Desenvolvendo a equação em função da força aplicada:

$$F \cdot 248{,}45 \cdot 10^{-3} + (m_1 \cdot 199.125 + m_S \cdot 342{,}05 + m_2 \cdot 149.8) \cdot 9{,}8 \cdot 10^{-6} = 1{,}1N.m \tag{5.5}$$

A partir da equação (5.5) é possível observar que, a partir das especificações do atuador do braço do manipulador é possível aplicar uma força de F = 3,581N.

$$F = 3{,}581N \tag{5.6}$$

Após realizada toda a construção do manipulador, incluindo a parte mecânica e eletrônica, foram realizados testes para constatar o desempenho do projeto. Fatores como repetitividade e precisão serão analisados.

6. RESULTADOS E ANÁLISE

O resultado final da montagem mecânica pode ser visto através da Figura 6.1. A montagem se mostrou estável e com mínima folga para trepidação. O manipulador apresenta movimentos coordenados mesmo em trajetórias feitas em intervalos curtos de tempo. Foi possível observar que a utilização de rolamentos, tanto os axiais quanto radiais, contribuíram muito para oferecer suavidade aos movimentos.

Neste capítulo serão feitas considerações em relação ao resultado da montagem mecânica do manipulador, da confecção do circuito eletrônico do controlador e dos movimentos em modo automático. Em seguida serão apresentados resultados referentes à testes desenvolvidos com a intenção de avaliar fatores importantes na robótica industrial como precisão e repetitividade.

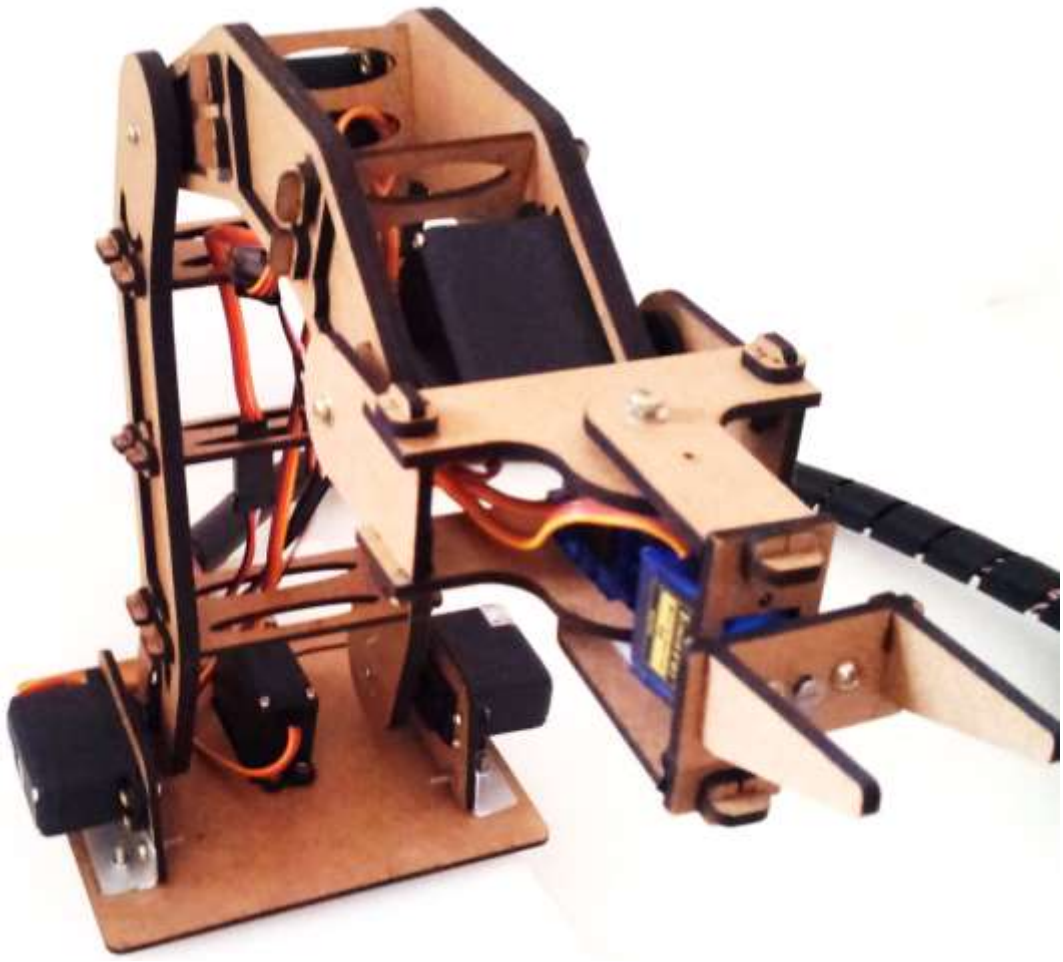

Figura 6.1: Resultado final da montagem mecânica

Alguns detalhes da estrutura foram destacados como mais importantes para análise. A Figura 6.2 mostra o terceiro eixo do manipulador, onde é possível observar na imagem o servo motor de posicionamento Tower Pro MG-995 e os dois rolamentos radiais SKF 623 utilizados.

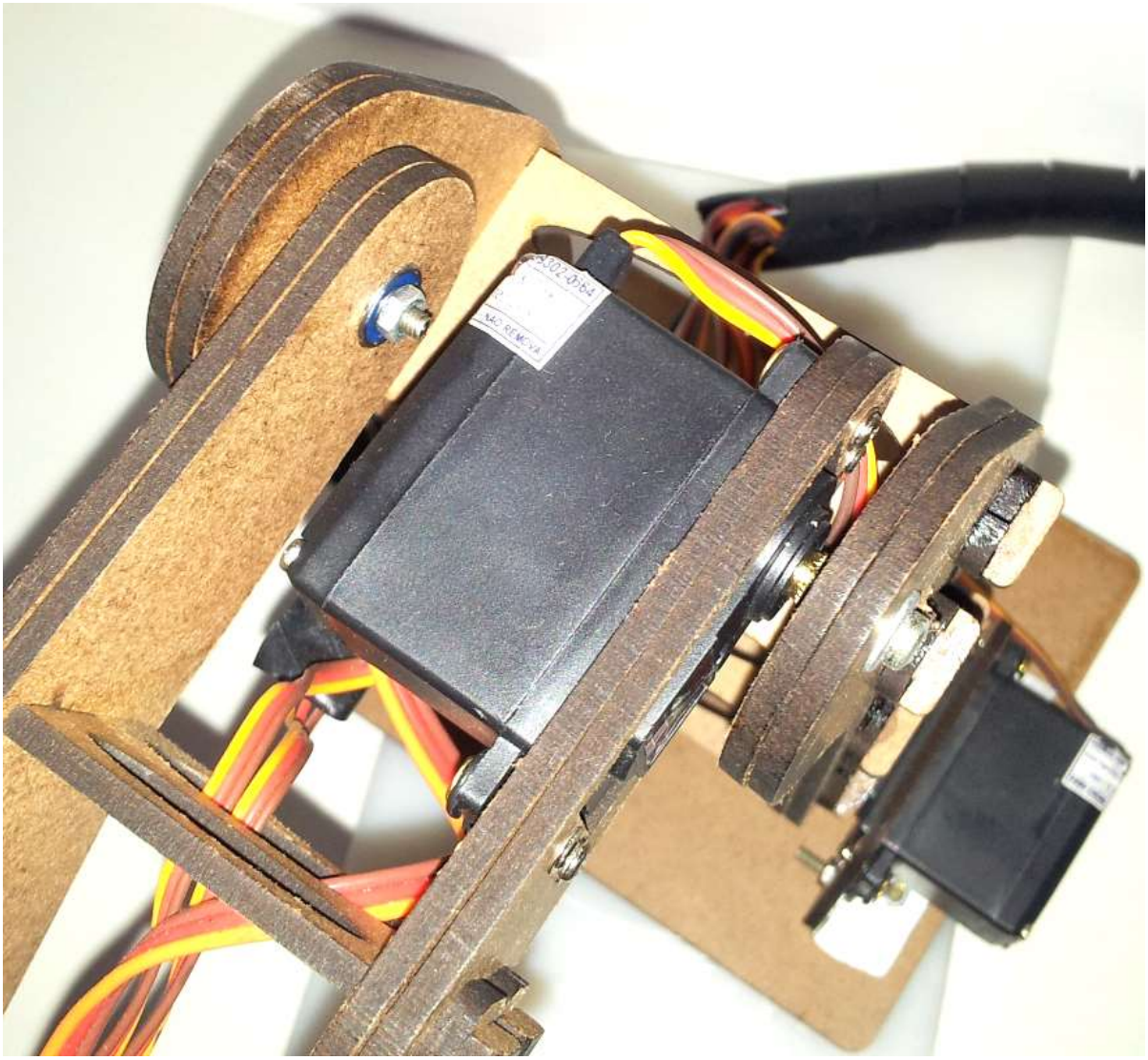

Figura 6.2: Detalhe, servo motor de posicionamento e rolamentos radiais

A Figura 6.3 mostra o detalhe da ferramenta adotada pelo sistema. O manipulador projetado e desenvolvido neste trabalho pode trabalhar com qualquer ferramenta terminal genérica se acoplada na posição final do manipulador. Esta ferramenta terminal pode ser uma garra com movimento de abertura, uma ponta de solda ou uma broca, por exemplo. Foi adotada uma garra simples sem movimento neste projeto.

Também pode ser observado pela Figura 6.3 os eixos destinados à movimentação dos graus de liberdade relacionados à orientação da ferramenta terminal. Os servo motores adotados para o quinto e sexto grau de liberdade, Tower Pro SG 90, foram escolhidos pelo seu pouco peso. Rolamentos radiais SKF 623 foram adotados, dois no quarto eixo e um quinto, para oferecer estabilidade e reduzir trepidação.

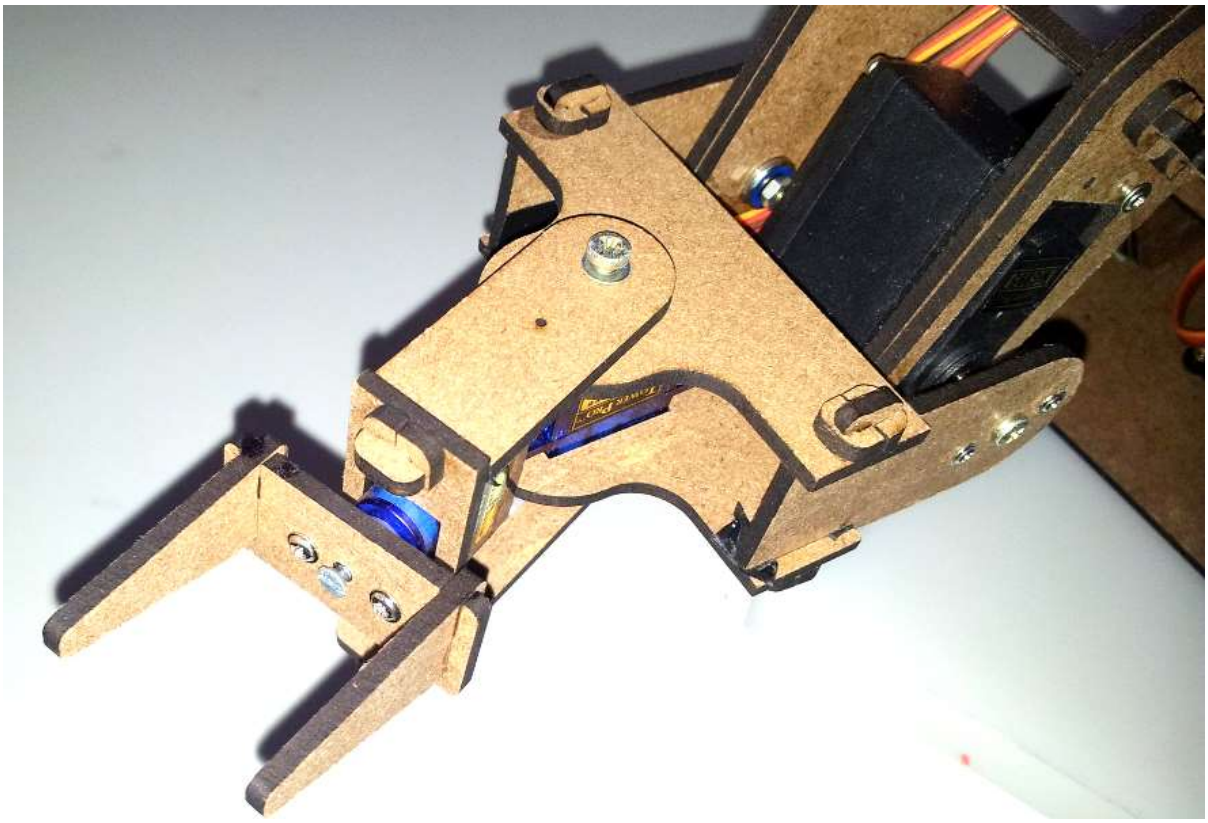

Figura 6.3: Detalhe, eixos de orientação e ferramenta terminal

A Figura 6.4 mostra em detalhe o primeiro e segundo eixos do manipulador. O primeiro eixo é afixado a uma base de acrílico usinada utilizando fresadora CNC e apoiado em um rolamento axial SKF 51116 para reduzir trepidação, ambos a base e o rolamento podem ser vistos pela Figura 6.5. O segundo eixo foi reforçado com o uso de dois servo motores MG-996R, como pode ser observado na Figura 6.4. O objetivo foi duplicar a capacidade de carga da estrutura visto que este é o eixo mais crítico. Calibragens foram necessárias para que um servo motor não represente carga para o segundo, visto que eles apresentam rotações opostas.

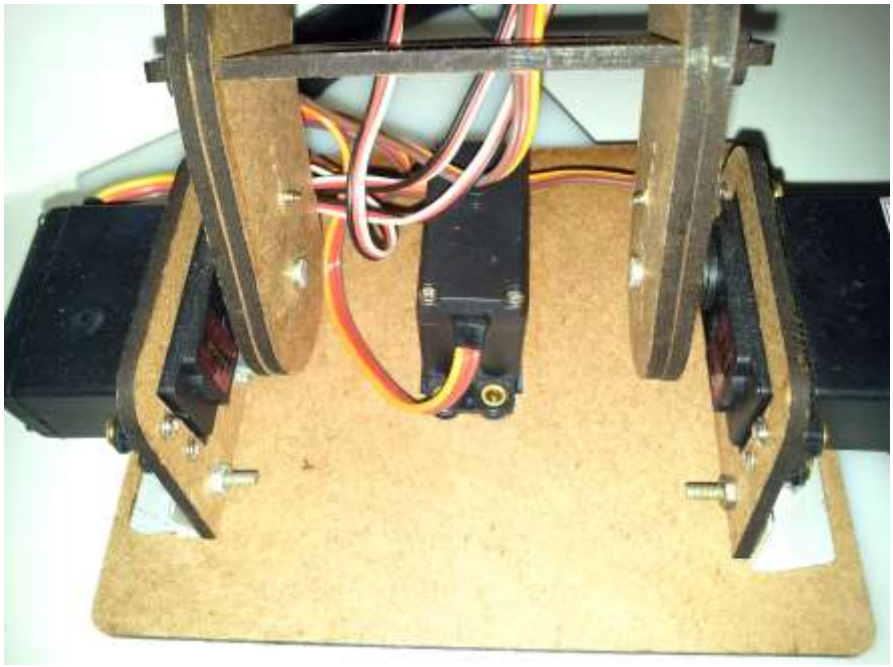

Figura 6.4: Detalhe, primeiro e segundo eixo do manipulador

Figura 6.5: Detalhe, base de nylon e rolamento axial

A Figura 6.6 apresenta a placa de circuito impresso confeccionada já com os componentes soldados. É possível observar a presença dos elementos principais do sistema controlador, como por exemplo o microcontrolador PIC18F46K22 (1), a entrada de dados pelo TP (2), a entrada de tensão pela fonte de alimentação (3), a saída de pulsos para os servo motores (4), a saída de dados para o LCD (5), o botão de *reset* manual (6), os componentes responsáveis pela oscilação do *clock*: o cristal piezoeletrônico e dois capacitores (7).

Através da Figura 6.6 é possível observar a presença de pinos de saída para um servo motor extra (8). A intenção de dispor uma saída para um sétimo atuador é tornar possível acrescentar um servo motor destinado ao movimento de uma ferramenta, caso seja a opção do operador. Também é possível observar a presença de pinos extras para entrada e saída de dados em comunicação serial (9) que não foram utilizados durante o projeto, mas foram dispostos na placa para oferecer uma funcionalidade extra em trabalhos futuros. Os pinos apresentados em (10) são os pinos de programação que são ligados por USB a um computador utilizando o dispositivo iCP01-V2.0 (Lab, 2010).

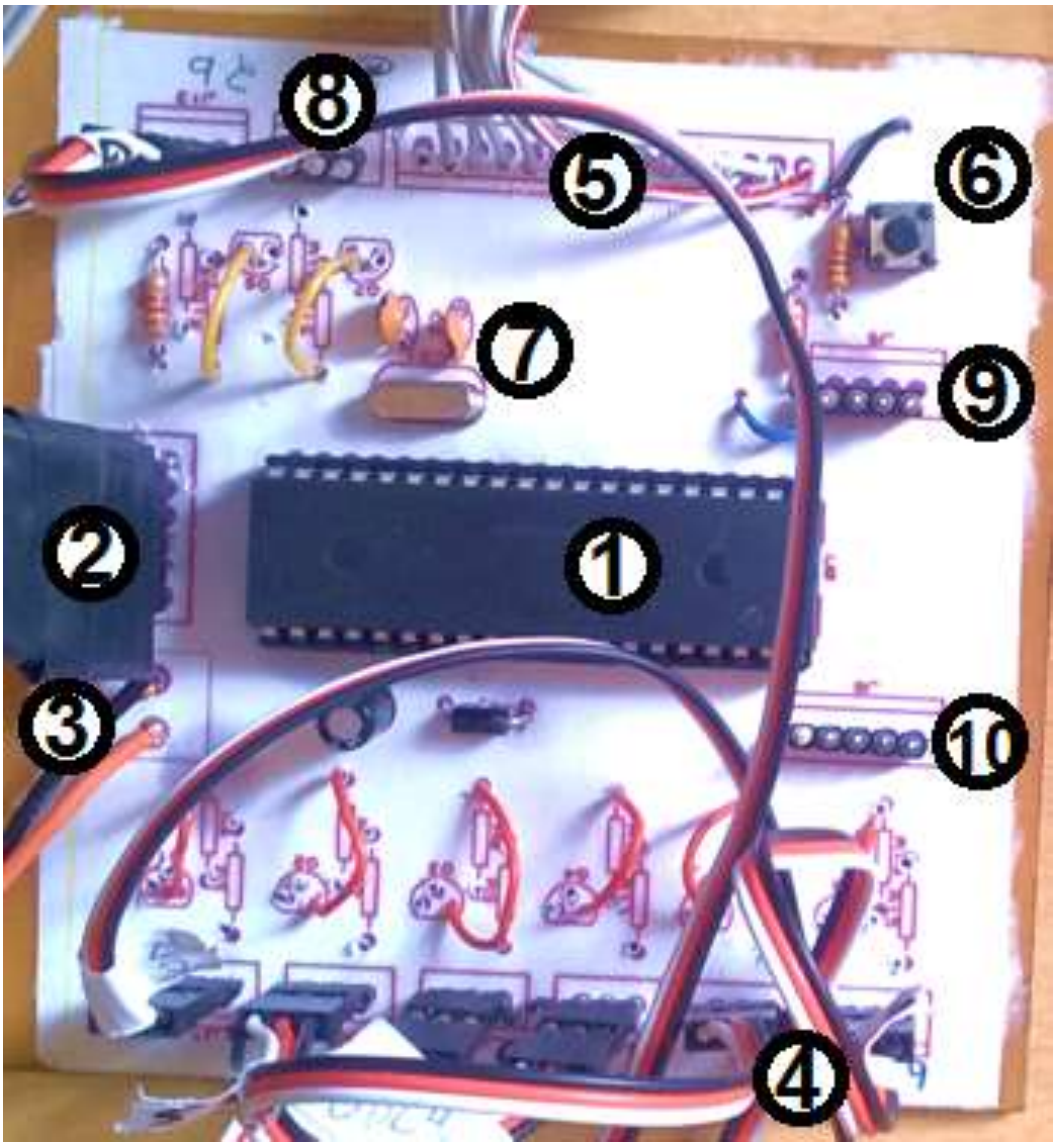

Figura 6.6: Resultado final da montagem eletrônica

O circuito de controle foi posicionado dentro de uma caixa de proteção mostrada, cujo objetivo foi tornar o projeto melhor organizado e apresentável.

Figura 6.7: Central de controle

A Figura 6.8 mostra uma sequência de imagens onde é possível observar o manipulador cumprindo uma trajetória que simula o posicionar de um objeto no espaço. Inicialmente, o robô se aproxima do objeto pretendido e, com uma velocidade maior, ele realiza um movimento de pegar o objeto, como é possível observar no quadro 4. A partir disso, o robô posiciona o objeto em uma outra posição, com pode ser visto até o quadro 8, e depois o robô retorna à primeira posição treinada como mostrado no quadro 12. Todos estes movimentos foram feitos utilizando equações de geração de trajetória de forma a anular movimentos bruscos ou descoordenados (com uma junta atingindo o ponto final antes da outra).

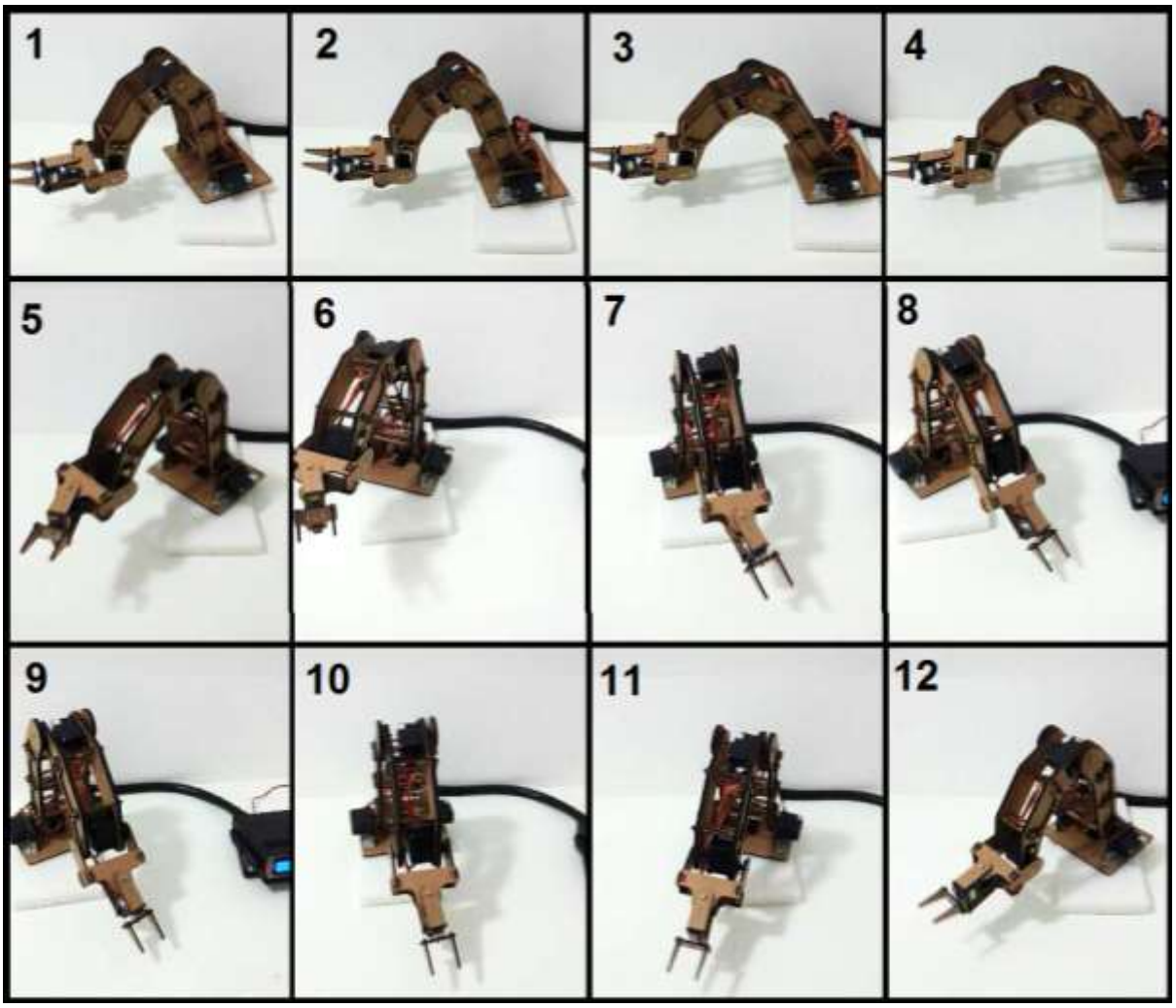

Figura 6.8: Sequência de imagens de trajetória do manipulador em modo automático

Para examinar fatores como repetitividade e precisão, foram desenvolvidos uma série de procedimentos de teste. Para o fator de precisão foi adotada uma metodologia de medição em que o robô foi posicionado sobre uma folha de papel branca. Em seguida, foi anotado sobre a folha de papel onde estava centralizada a base do robô, ao qual todas as medidas apresentadas no visor faziam referência. A partir disto foram realizados vinte e uma medições de posicionamentos diferentes, de forma que fosse possível comparar os valores apresentados no visor com os valores reais no plano XY. A Figura 6.9 representa cinco posicionamentos feitos pelo manipulador durante esta etapa de testes

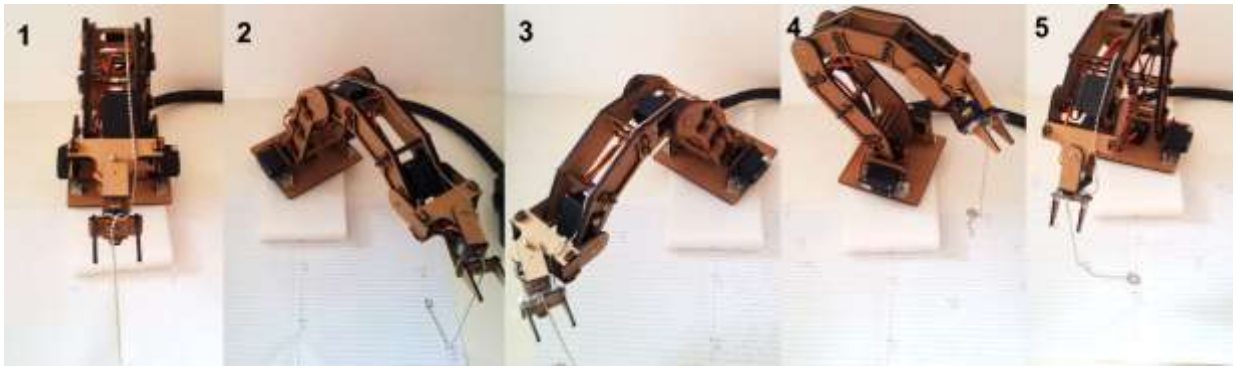

Figura 6.9: Procedimento de teste para o fator de precisão

A partir dos dados coletados é possível observar pelo gráfico da Figura 6.10, que os pontos marcados sobre a folha são bem próximos dos pontos que o manipulador pretendia alcançar. É possível analisar que os pontos mais distantes da origem costumam possuir uma divergência maior entre os valores reais e pretendidos. A partir deste fato é possível concluir que existe uma perda de precisão por parte dos atuadores em casos de carga muito grande, visto que para atingir os pontos mais distantes da origem o manipulador precisa se esticar mais, aplicando uma carga maior sobre os servo motores

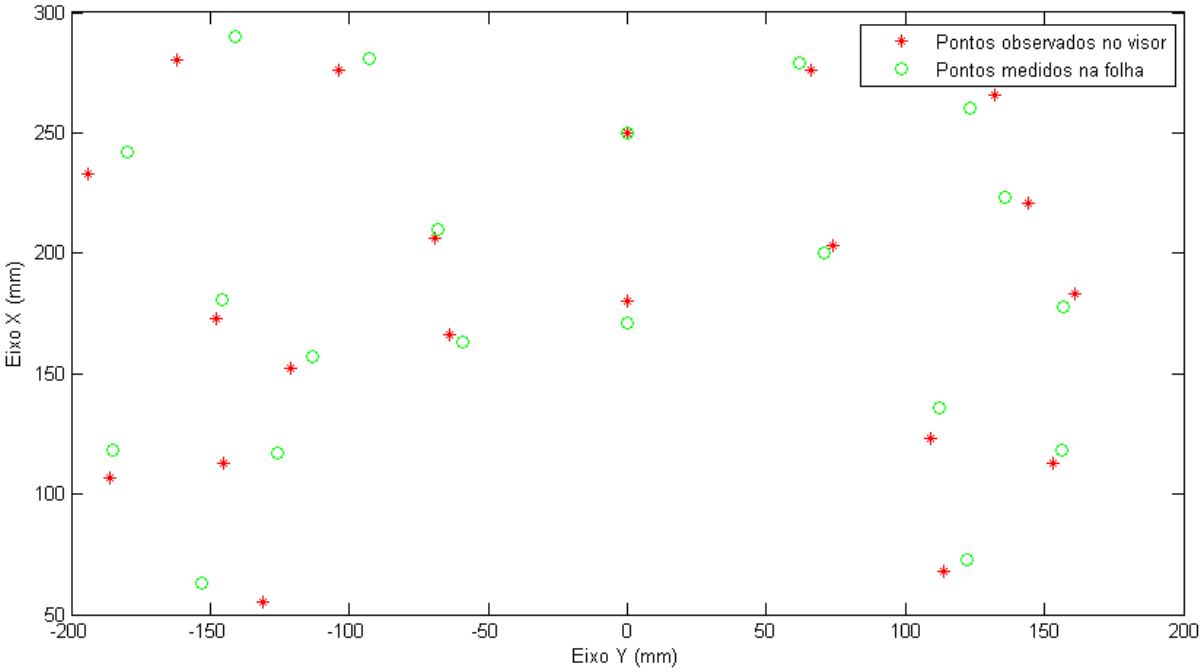

Figura 6.10: Dados plotados para pontos marcados sobre a folha e pontos mostrados pelo visor

Para medir a repetitividade do manipulador foi aplicado um marcador com tinta à ponta da ferramenta terminal e treinado o robô a repetir a mesma movimentação encostando o marcador ao papel. A Figura 6.11 representa o procedimento utilizado para medir o fator de repetitividade.

Figura 6.11: Procedimentos de teste para repetitividade

A partir do procedimento realizado pôde ser constatado que o manipulador tem pouca capacidade de manter uma repetitividade constante. Pode ser observado que o manipulador erra o ponto 0,5mm a cada iteração, resultando em uma falha de 5mm já na décima movimentação.

Outra habilidade do manipulador que também foi aferida é a sua capacidade de manter constante o posicionamento final da ferramenta se for alterada sua orientação. O esperado é que quando modificado somente o valor dos ângulos de orientação, os outros ângulos se rearranjem para manter o mesmo posicionamento. A Figura 6.12 mostra esta habilidade do manipulador em se reorganizar para manter o mesmo posicionamento. É possível observar que ele não mantem uma precisão, porém que os outros ângulos tendem a se reorientar para tentar manter este posicionamento.

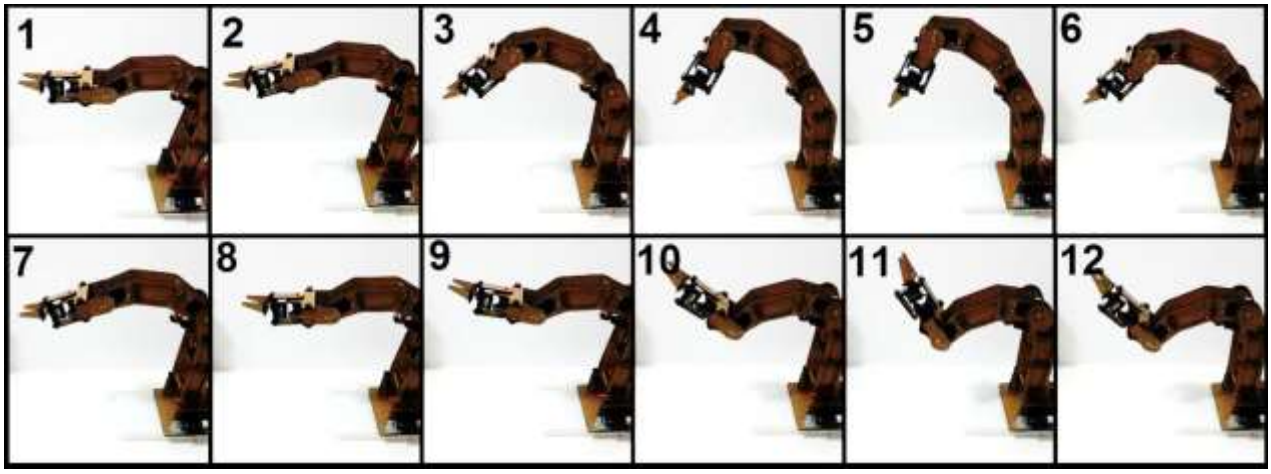

Figura 6.12: Manipulador na mesma posição modificando somente a orientação da ferramenta

Os resultados se mostraram parecidos, porém não exatos. Um possível fator que possa ter provocado esta diferença pode ser atribuído à precisão dos motores, visto que a etapa de modelagem foi feita com fidelidade ao projeto.

Na etapa seguinte serão feitas considerações em relação a este trabalho e a conclusão final.

7. CONSIDERAÇÕES FINAIS E CONCLUSÃO

Através dos testes pôde ser percebido que o manipulador articulado não possui ótimos fatores de precisão e repetitividade. Porém, satisfatórios para funções como auxiliar em kits educacionais e como ferramenta de aprendizagem.

O manipulador foi confeccionado com um baixo orçamento, o que o torna altamente viável para ser utilizado em escolas. Mesmo em universidades e instituições de ensino superior é possível observar aplicabilidade para um manipulador nestas características, visto que a aquisição de um manipulador industrial nem sempre é possível, dependendo de diversos fatores como custos e dificuldades burocráticas. A utilização deste manipulador em uma sala de aula de ensino superior de engenharia pode ser muito útil na apresentação de conceitos básicos de robótica industrial, tais como posicionamento e orientação, e também conceitos aplicados como modelagem cinemática e dinâmica, além de geração de trajetória, programação e controle.

O desenvolvimento do manipulador articulado deste trabalho foi feito com alguns incrementos para ser usado em trabalhos futuros, são estes:

- A possibilidade de integrar uma garra, ou qualquer outro tipo de ferramenta terminal, visto que a placa de circuito impresso confeccionada tem disponibilidade para mais uma saída para atuador.

- A possibilidade de obter dados via entrada serial do computador, visto que a placa foi confeccionada com entradas para dados seriais pelas portas do microcontrolador TX e RX.

- A possibilidade de modificar o programa facilmente, visto que existe o encaixe na placa de circuito impresso para tal feito.

Outra consideração importante a ser feita é referente à diferença observada entre o manipulador projetado neste trabalho e um manipulador de grande porte utilizado em indústrias. É certo que foram feitas diversas simplificações por se tratar de um manipulador de pequeno porte. Porém, pode-se afirmar que o princípio de funcionamento é muito semelhante ao de um robô industrial.

Uma possível pesquisa que poderia surgir a partir deste trabalho seria desenvolver um controle de posição em motores de corrente contínua de pequeno porte utilizando malha fechada com sensores de posição rotativos, tornando os motores servo controlados. A partir disto, aplicar o mesmo princípio de funcionamento utilizado neste trabalho em uma estrutura com um porte maior. Esta solução poderia se desenvolver em um produto para aplicações simples e que não necessite de muita precisão.

Outra iniciativa que poderia surgir a partir deste trabalho seria replicar este projeto e oferecê-lo a instituições de ensino que buscam uma alternativa viável de promover o estudo de robótica. Por ser um projeto de baixo custo, seria possível desenvolver um produto a partir destes resultados e comercializá-lo. A Tabela 4 mostra os preços dos principais materiais utilizados neste trabalho.

Tabela 4: Relação de preço dos principais materiais

Material	**Preço**
Estrutura (corte e material)	R$70,00
Atuadores	R$175,52
Microcontrolador (peça + frete)	R$30.36
Visor LCD	R$17,00
Base de Nylon	R$25,00
Caixa para circuito	R$20,00
Controle para *Play Station*	R$30,00
Acessórios (Componentes eletrônicos, parafusos, entre outros)	R$50,00
Total	**R$417,88**

Todos os programas, desenhos e circuitos estarão à disposição para qualquer aluno ou professor que estiver disposto a refazer ou dar continuidade ao projeto. Assim como eu também me disponibilizo a auxiliar.

8. REFERÊNCIAS

ABB, 2000. *International Report.* Zurich: s.n.

Architects, E., 2010. *EMBEDDED ARCHITECTS.* Disponível em: <http://www.embarc.com.br/p1600.aspx>. Acesso em: 05 jan. 2014.

Corke, P. I., 2002. *Robotic Toolbox for MATLAB.* Pullenvale, Australia: CSIRO.

Craig, J. J., 1986. *Introduction to Robotics: Mechanics and Control.* Third Edition ed. s.l.:Pearson Education International.

D´Ignazio, F., 1982. *Working Robots.* s.l.:Lodestar Books.

Denavit, J., 1955. *Description and displacement analysis of mechanics based on 2x2 dual matrices,* Evaston, IL: Universidade de Northwestern.

ErmicroBlog, s.d. *TowerPro SG90 - Micro Servo.* Disponível em: http://www.servodatabase.com/servo/towerpro/sg90. Acesso em: 10 jun. 2012.

G1, 2011. *Robotica educacional pega carona em sustentabilidade para atrair alunos.* Disponível em: http://g1.globo.com/educacao/noticia/2011/05/robotica-educacional-pega-carona-em-sustentabilidade-para-atrair-alunos.html. Acesso em: 15 out. 2012.

GrabCAD, 2012. *Servo Robot ARM.* Disponível em: http://grabcad.com/library/servo-robot-arm. Acesso em: 20 set. 2013.

Health, S., 2003. *Embedded Systems Design.* 2nd ed. s.l.:Newnes.

Inventor, C., s.d. *Curious Inventor.* Disponível em: http://store.curiousinventor.com/guides/PS2/#hardware. Acesso em: 21 nov. 2013.

Lab, E., 2010. *iCP01 USB PIC Programmer.* Disponível em: http://embedded-lab.com/blog/?p=641. Acesso em: 21 fev. 2014.

MakeItFrom, 2009. *Medium Density Fiberboard (MDF).* Disponível em: http://www.makeitfrom.com/material-data/?for=Medium-Density-Fiberboard-MDF. Acesso em: 05 dez. 2013.

Microship, 2012. *PIC18(L)F2X/4XK22 Data sheet.* s.l.:s.n.

Nof, S. Y., 1985. *Handbook of Industrial Robotics.* West Lafayette: John Wiley & Sons.

Onwubolu, G., 2005. *Mechatronics Principles and Aplications.* Burlington, MA: Elsevier Butterworth Heinemann.

Paul, R. P., B, S. & Mayer, G., 1981. *Kinematic Control Equations for Simple Manipulators.* s.l.:IEEE Transactions on Aerospace and Electronic Systems.

Rosário, J. M., 2005. *Principios de Mecatrônica.* São Paulo: Pearson Prentice Hall.

Santos, R. R., Saramago, S. d. F. & Jr., S. V., 2004. Especificação de Trajetória de Robôs em Tempo Ótimo. *14° POSMEC - Simpósio do Programa de Pós-Graduação em Engenharia Mecânica,* p. 9.

SKF, 2013. *Rolamentos rígidos de esferas, uma carreira.* Disponível em: http://www.skf.com/br/products/bearings-units-housings/ball-bearings/deep-groove-ball-bearings/single-row/index.html?prodid=1010016230&imperial=false. Acesso em: 15 jan. 2013.

Spong, M. W., Seth, H. & Vidyasagar, M., 1989. *Robot Modeling and Control.* Nova York: John Wiley & Sons, INC..

Tanenbaum, A. S., 1992. *Organização Estruturada de Computadores.* Quinta edição ed. s.l.:Pearson.

Technologies, C., 2011. *How RC Servo Works?.* Disponível em: http://tutorial.cytron.com.my/2011/09/19/how-rc-servo-works/. Acesso em: 20 dez. 2013.

UOL, 2011. *Robos em sala de aula aumentam motivação dos alunos. .* Disponível em: http://educacao.uol.com.br/noticias/2011/05/19/robos-em-sala-de-aula-aumentam-motivacao-dos-alunos-diz-educadora.htm. Acesso em: 15 out. 2012.

Veja, 2012. *A importancia do incentivo ao raciocinio logico e ao gosto pela investigacao cientifica.* Disponível em: http://veja.abril.com.br/noticia/educacao/a-importancia-do-incentivo-ao-raciocinio-logico-e-ao-gosto-pela-investigacao-cientifica. Acesso em: 15 out. 2012.

ANEXO A: PROGRAMA DO MICROCONTROLADOR

```
// TCC.C

#include <tcc.h>
#include <math.h>
#include "mod_lcd.c"
#include "funcoes.c"

#define x_ini 250
#define y_ini 0
#define z_ini 180

unsigned int16 ton[8];
int16 t=0,var_timer = 10000;
int c=0,inicia=0;
int1 flag_motor=0;


/*
ton[0] GDL 1 Posicionamento 1
ton[1] GDL 2 Posicionamento 2 Motor 1
ton[2] GDL 3 Posicionamento 3
ton[3] GDL 4 Orientação 1
ton[4] GDL 5 Orientação 2
ton[5] GDL 6 Orientação 3
ton[6] Garra
ton[7] GDL 2 Posicionamento 2 Motor 2
*/

#INT_TIMER1          // Timer1 interrupt
void timer1()
{
  if(flag_motor)
  {
    /*================== Atualiza Motores ======================*/
    switch(c){
    case 0:
    output_high(motor1);
    delay_us(ton[0]);
    output_low(motor1);

    output_high(motor2);
    delay_us(ton[1]);
    output_low(motor2);
    c++;
    break;
    case 1:
```

```
    output_high(motor3);
    delay_us(ton[2]);
    output_low(motor3);

    output_high(motor4);
    delay_us(ton[3]);
    output_low(motor4);
    c++;
    break;
    case 2:
    output_high(motor5);
    delay_us(ton[4]);
    output_low(motor5);

    output_high(motor6);
    delay_us(ton[5]);
    output_low(motor6);
    c++;
    break;
    case 3:
    output_high(motor7);
    delay_us(ton[6]);
    output_low(motor7);

    output_high(motor8);
    delay_us(ton[7]);
    output_low(motor8);
    c=0;
    break;
    }
  }

  set_timer1(get_timer1()+var_timer);
}


/*====================== Programa Principal ====================== */

void main()
{

  /*====================== Declarações iniciais ====================== */

  enable_interrupts(global);
  enable_interrupts(int_timer1);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_2);
  set_timer1(var_timer);
  lcd_ini();  // Inicia LCD

  /*=================== Declaração de Variáveis ====================== */
```

```
  //min[7]={-190,-190,-190,-190,-190,-190,-190},max[7]={190,190,190,190,190,190,190};
  int i;
  signed int32 ang[7],def[7]={0,0,0,0,0,0,0},min[7]={-90,-90,-90,-90,-90,-90,-
90},max[7]={90,90,90,90,90,90,90};// Variaveis de angulo
  int analog=0, ps_dig1=0, ps_dig2=0,ps_anRx=0,ps_anRy=0,ps_anLx=0,ps_anLy=0,ps_in; // Variaveis
do controle de play station
  float32 angulo[6];
  float32 passo,y_passo, x_passo, z_passo;
  float32  K0,K1,K2,K3,K4;
  signed int16 quad_max_x=350,quad_min_x=0,quad_max_y=350,quad_min_y=-
350,quad_max_z=350,quad_min_z=-200;
  signed int32 x,y,z; // Variaveis de posição global
  float32 a = 180,b = 150, c = 53, d = 47,rot1,rot2,rot3;
  char modo[5];
  int vel=3,n_mgrav=1;
  int1
flagO=0,flagTR=0,flagXg=0,flagX1=0,flagX2=0,flagR1=0,flagR2=0,flag_prim_l=0,flag_rpt=0,flag_pause=
0, flag_reset=0,flag_carregando = 0;
  unsigned int tf,n_mem=1,n_mem_f;
  signed int ang_m[7],ang_mf[7];
  signed int32 angI[6],angF[7];//Variaveis de geração de tragetória
  float32 a2[6],a3[6],c1,s1,c3,s3,c4,s4,c5,s5;

  /*========== Inicialização de variáveis e declarações ============== */

  x = x_ini;
  y = y_ini;
  z = z_ini;
  rot1 = 0;
  rot2 = 0;
  rot3 = 0;

  output_high(clk_ps); // Inicializa Play Station
  output_high(att_ps);

  modo[0] = 'T';
  modo[1] = 'r';
  modo[2] = 'e';
  modo[3] = 'i';
  modo[4] = 'n';

  x = x_ini;
  y = y_ini;
  z = z_ini;

  passo = 10; //Variavel que controla velocinade no modo treinamento

  for(i=0;i<7;i++)
  {
    ang[i] = def[i];
```

```
}

/*======================== Loop Principal ========================== */

while(true)
{
  if(inicia<110)
  {
    inicia++;
    flag_motor=0;
    if(inicia>50) {
    flag_motor=1;
    }
  }
  if (inicia==110)
  {
    inicia++;
    setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
    var_timer = 40536;
    flag_carregando = 1;
  }
  /*================== Atualiza controle ==================== */

  output_low(att_ps);
  ps_in = play(0x01);
  analog = play(0x42);
  ps_in = play(0x00);
  ps_dig1 = play(0x00);
  ps_dig2 = play(0x00);
  if(analog==0x73)
  {
    ps_anRx = play(0x00);
    ps_anRy = play(0x00);
    ps_anLx = play(0x00);
    ps_anLy = play(0x00);
  }
  output_high(att_ps);

  /*================ Botão L2 ==================== */
  if(!bit_test(ps_dig2,0)){passo=1;} // Diminui passo para ajuste fino
  else passo = 10;

  /*========= Tratamento de butões e Joypads ================ */
  if(analog==0x73&&n_mgrav==1)
  {
    /*============ Joypad Esquerdo X = Y ==================== */

    if(ps_anLx>=0  && ps_anLx<=255)
    {
      y_passo = (2*passo/256)*ps_anLx - passo; // 10 em 256; 0 em 128; -10 em 0
```

```
  y = y + y_passo;
  if(y>=quad_max_y)
  {
    y=quad_max_y;
  }
  if(y<=quad_min_y)
  {
    y=quad_min_y;
  }
}

/*=========== Joypad Esquerdo Y = X ==================== */

if(ps_anLy>=0  && ps_anLy<=255)
{
  x_passo = (2*passo/256)*ps_anLy - passo; // 10 em 0; 0 em 128; -10 em 256
  x = x + x_passo;
  if(x>=quad_max_x)
  {
    x=quad_max_x;
  }
  if(x<=quad_min_x)
  {
    x=quad_min_x;
  }
}

/*=========== Joypad Direito Y = Z ==================== */

if(ps_anRy>=0  && ps_anRy<=255)
{
  z_passo = -(2*passo/256)*ps_anRy + passo; // 10 em 0; 0 em 128; -10 em 256
  z = z + z_passo;
  if(z>=quad_max_z)
  {
    z=quad_max_z;
  }
  if(z<=quad_min_z)
  {
    z=quad_min_z;
  }
}
/*=========== Joypad Direito X = quinto GDL ==================== */

if(ps_anRx>=0  && ps_anRx<=255)
{
  if(ps_anRx>128)
  {
    rot2=rot2+passo;
  }
```

```
  if(ps_anRx<128)
  {
    rot2=rot2-passo;
  }
  if(rot2>=90)
  {
    rot2=90;
  }
  if(rot2<=-90)
  {
    rot2=-90;
  }
  angulo[4]=rot2/57.3;
}
/*=========== Cima/baixo = quarto GDL ================== */

if(!bit_test(ps_dig1,4))//Cima
{
  rot1=rot1+passo;
}
if(!bit_test(ps_dig1,6))//Cima
{
  rot1=rot1-passo;
}
if(rot1>=90)
{
  rot1=90;
}
if(rot1<=-90)
{
  rot1=-90;
}
angulo[3]=rot1/57.3;
/*=========== Direito/esquerdo = sexto GDL ================== */

if(!bit_test(ps_dig1,5))//Direito
{
  rot3=rot3+passo;
}
if(!bit_test(ps_dig1,7))//Esquerdo
{
  rot3=rot3-passo;
}
if(rot3>=90)
{
  rot3=90;
}
if(rot3<=-90)
{
  rot3=-90;
```

```
  }
}

/*=============== Botão Select ==================== */
if(!bit_test(ps_dig1,0)) // Reseta tudo
{
  flag_reset = 1;
  n_mgrav = 0; // Impossibilita treinar o robô durante reiniciação
  angF[0] =   0; // Valores para robô "sentado"
  angF[1] =  65;
  angF[2] = -58;
  angF[3] = -45;
  angF[4] =   0;
  angF[5] =   0;
  angF[6] =   0;
  tf = 300; //Velocidade baixa
  for (i=0;i<6;i++)
  {
    angI[i]=ang[i];
  }
  for (i=0;i<6;i++)
  {
    a2[i] = 3*(angF[i]-angI[i])*pow(tf,-2);
    a3[i] = -2*(angF[i]-angI[i])*pow(tf,-3);
  }
  t=0;
}
if(bit_test(ps_dig1,0)&&flag_reset==1) // Reseta tudo
{
  t++;
  for (i=0;i<6;i++)//Atualiza valores de angulo
  {
    angulo[i]=angI[i]+a2[i]*pow(t,2)+a3[i]*pow(t,3);
  }

  if(t==tf)//Chegou na posição final
  {
    reset_cpu();
  }

}
/*=============== Botão Start ==================== */
if(!bit_test(ps_dig1,3)) // Reseta tudo
{
  flag_pause = 1;
  n_mgrav = 0; // Impossibilita treinar o robô durante reiniciação
  angF[0] =   0; // Valores para robô "sentado"
  angF[1] =  65;
  angF[2] = -58;
  angF[3] = -45;
```

```
  angF[4] =   0;
  angF[5] =   0;
  angF[6] =   0;
  tf = 300; //Velocidade baixa
  for (i=0;i<6;i++)
  {
    angI[i]=ang[i];
  }
  for (i=0;i<6;i++)
  {
    a2[i] = 3*(angF[i]-angI[i])*pow(tf,-2);
    a3[i] = -2*(angF[i]-angI[i])*pow(tf,-3);
  }
  t=0;
}
if(bit_test(ps_dig1,3)&&flag_pause==1) // Reseta tudo
{
  t++;
  for (i=0;i<6;i++)//Atualiza valores de angulo
  {
    angulo[i]=angI[i]+a2[i]*pow(t,2)+a3[i]*pow(t,3);
  }

  if(t==tf)//Chegou na posição final
  {
    n_mgrav = 1; // retorna para modo treinamento
    x = 70;
    y = 0;
    z = 20;
    angulo[3]=-25/57.3;
    angulo[4]=0;
    angulo[5]=0;
    flag_pause = 0;
  }

}
/*=============== Botão Triangulo ================== */
if(!bit_test(ps_dig2,4)){flagTR = 1;} // Botão Triangulo
if(bit_test(ps_dig2,4) && flagTR == 1)// Modifica o modo Repete ou não
{
  if(flag_rpt==0)flag_rpt = 1;
  else flag_rpt = 0;
  flagTR = 0;
}
/*=============== Botão R2 ================== */
if(!bit_test(ps_dig2,1)){flagR2=1;} // Diminui intervalo de tempo para realizar o movimento
if(bit_test(ps_dig2,1) && flagR2==1)
{
  flagR2=0;
  vel++;
```

```
  if(vel>5)vel=5;
}

/*=============== Botão R1 ==================== */
if(!bit_test(ps_dig2,3)){flagR1=1;} // Aumenta intervalo de tempo para realizar o movimento
if(bit_test(ps_dig2,3) && flagR1==1)
{
  flagR1=0;
  vel--;
  if(vel<1)vel=1;
}

/*==============  Botão O - Seleciona o modo ==================== */
if(!bit_test(ps_dig2,5)){flagO = 1;} // Botão O
if(bit_test(ps_dig2,5) && flagO == 1)// Modifica o modo Treinamento/Automático
{
  flagO=0;
  n_mgrav++;
  n_mem=1;
  if(n_mgrav>2){n_mgrav=1;}
}
switch(n_mgrav)
{
  case 1:
  modo[0] = 'T';
  modo[1] = 'r';
  modo[2] = 'e';
  modo[3] = 'i';
  modo[4] = 'n';
  break;//Treinamento
  case 2:
  modo[0] = 'A';
  modo[1] = 'u';
  modo[2] = 't';
  switch(flag_rpt)
  {
    case 0:
    modo[3] = 'o';
    modo[4] = 'm';
    break;
    case 1:
    modo[3] = 'R';
    modo[4] = 'p';
    break;
  }
  break;//Automático
}

/*==============  Botão X - Grava posição ==================== */
if(!bit_test(ps_dig2,6)&& n_mgrav==1){flagXg = 1;} // Botão X
```

```
    if(bit_test(ps_dig2,6) && n_mgrav==1 && flagXg == 1)// Grava posição no modo leitura
    {
      flagXg = 0;
      for(i=0;i<7;i++)
      {
        ang_m[i]= ang[i]+90;
      }
      for(i=0;i<7;i++)
      {
        write_eeprom((n_mem-1)*8+i+1,ang_m[i]);
      }
      write_eeprom(n_mem*8,vel);
      n_mem++;
      if(n_mem>128)n_mem=1;
      n_mem_f = n_mem;
    }

    /*==============  Botão X - Lê posição ==================== */
    if(!bit_test(ps_dig2,6) && n_mgrav==2){flagX1 = 1;n_mem=0;flag_prim_l = 1;} // Botão O
    if(bit_test(ps_dig2,6) && n_mgrav==2 && flagX1 == 1)// Modifica o modo
Treinamento/Automático
    {
      if(flag_prim_l == 1)
      {
        If(n_mem==0)
        {
          vel=5;
          for (i=0;i<7;i++)
          {
            angF[i]=def[i];
          }
        }
        If(n_mem>0)
        {
          ang_mf[0]=read_eeprom((n_mem-1)*8+1);
          ang_mf[1]=read_eeprom((n_mem-1)*8+2);
          ang_mf[2]=read_eeprom((n_mem-1)*8+3);
          ang_mf[3]=read_eeprom((n_mem-1)*8+4);
          ang_mf[4]=read_eeprom((n_mem-1)*8+5);
          ang_mf[5]=read_eeprom((n_mem-1)*8+6);
          ang_mf[6]=read_eeprom((n_mem-1)*8+7);
          vel  = read_eeprom(n_mem*8);

          for (i=0;i<6;i++)
          {
            angF[i]=ang_mf[i]-90;
          }
        }
        switch(vel)//Calcula tf com Velocidade
        {
```

```
      case 1: tf = 100;break;//Velocidade 1 - Mais lento
      case 2: tf = 80;break;//Velocidade 2
      case 3: tf = 60;break;//Velocidade 3
      case 4: tf = 40;break;//Velocidade 4
      case 5: tf = 20 ;break;//Velocidade 5 - Mais rápido
    }
    for (i=0;i<6;i++)//Do motor 1 ao 6 - Garra é operada diferente
    {
      angI[i]=ang[i];
    }
    for (i=0;i<6;i++)//Do motor 1 ao 6 - Garra é operada diferente
    {
      a2[i] = 3*(angF[i]-angI[i])*pow(tf,-2);
      a3[i] = -2*(angF[i]-angI[i])*pow(tf,-3);
    }
    t=0;
    flag_prim_l = 0;//flag da primeira interação de cada movimento
  }
  t++;

  for (i=0;i<6;i++)//Atualiza valores de angulo
  {
    angulo[i]=angI[i]+a2[i]*t*t+a3[i]*t*t*t;
  }

  if(t==tf)//Chegou na posição final
  {
    t=0;
    n_mem++;
    flag_prim_l=1;
    ang[6]=angF[6];
  }
  if(n_mem==n_mem_f)//Terminou a leitura
  {
    if(flag_rpt==0)
    {
      flagX1 = 0;
      n_mgrav=1;
    }
    n_mem=1;
    vel = read_eeprom(8);;
  }

}
/*=================== Cinemática Inversa =================== */

if(n_mgrav==1)//Somente no modo de Treinamento
{
  if (flagX1==1)
  {
```

```
        flagX2 = 1;
//!       n_mgrav = 1; // Impossibilita treinar o robô durante reiniciação
        angF[0] = 0; // Valores para robô "sentado"
        angF[1] = 0;
        angF[2] = 0;
        angF[3] = 0;
        angF[4] = 0;
        angF[5] = 0;
        angF[6] = 0;
        tf = 300; //Velocidade baixa
        for (i=0;i<6;i++)
        {
          angI[i]=ang[i];
        }
        for (i=0;i<6;i++)
        {
          a2[i] = 3*(angF[i]-angI[i])*pow(tf,-2);
          a3[i] = -2*(angF[i]-angI[i])*pow(tf,-3);
        }
        t=0;
        flagX1 = 0;
      }
      if(flagX2==1) // Reseta tudo
      {
        t++;
        for (i=0;i<6;i++)//Atualiza valores de angulo
        {
          angulo[i]=angI[i]+a2[i]*pow(t,2)+a3[i]*pow(t,3);
        }

        if(t==tf)//Chegou na posição final
        {
          flagX2 = 0;
          x = x_ini;
          y = y_ini;
          z = z_ini;
          angulo[3] = 0;
          angulo[4] = 0;
          angulo[5] = 0;
        }
      }
      else{
        c4 = cos(angulo[3]);
        s4 = sin(angulo[3]);
        c5 = cos(angulo[4]);
        s5 = sin(angulo[4]);

        K0 = sqrt(abs(x*x+y*y-d*d*s5*s5));

        angulo[0] = atan2(y,x)+atan2(d*s5,K0);
```

```
    K1 = (a*a+b*b+c*c+d*d-x*x-y*y-z*z+2*b*c*c4+2*c*d*c5+2*b*d*c4*c5)/(2*a);
    K2 = d*d*c5*c5+b*b+c*c+2*b*c*c4+2*c*d*c5+2*b*d*c4*c5;

    angulo[2] = -atan2(c*s4+d*s4*c5,b+c*c4+d*c4*c5) + atan2(-K1,sqrt(K2-K1*K1));

    c1 = cos(angulo[0]);
    s1 = sin(angulo[0]);
    c3 = cos(angulo[2]);
    s3 = sin(angulo[2]);

    K3 = z*(a*s3+b+c*c4+d*c4*c5)-(a*c3+s4*(c+d*c5))*(x*c1+y*s1);
    K4 = z*(a*c3+s4*(c+d*c5))+(a*s3+b+c*c4+d*c5*c4)*(x*c1+y*s1);


    angulo[1] = atan2(K3,K4)-angulo[2];

    angulo[0] = angulo[0]*57.3;//Converte de radianos para graus
    angulo[1] = angulo[1]*57.3;//57.295
    angulo[2] = angulo[2]*57.3;
    angulo[3] = angulo[3]*57.3;
    angulo[4] = angulo[4]*57.3;
    angulo[5] = rot3;
  }
}

/*=================== Trata valores ================== */

ang[0]=angulo[0];
if (ang[0]>=max[0])ang[0]=max[0];
if (ang[0]<=min[0])ang[0]=min[0];

ang[1] = angulo[1];
if (ang[1]>max[1])ang[1] = max[1];
if (ang[1]<min[1])ang[1] = min[1];

ang[2] = angulo[2];
if (ang[2]>max[2])ang[2] = max[2];
if (ang[2]<min[2])ang[2] = min[2];

ang[3] = angulo[3];
ang[4] = angulo[4];
ang[5] = angulo[5];


/*========== Atualiza valores para motores  ================== */

ton[0] =   8.744*ang[0]+1420; //MG995
ton[1] =   8.055*ang[1]+1375; //MG996R
ton[2] =   8.055*ang[2]+1500; //MG996R
```

```
    ton[3] =  -8.744*ang[3]+1444; //MG995
    ton[4] =  10.522*ang[4]+1447; //SG90
    ton[5] =  10.522*ang[5]+1447; //SG90
    ton[6] =  10.522*ang[6]+1447; //SG90
    ton[7] =  -8.055*ang[1]+1455; //MG995 Inverso

    if(flag_carregando==0)
    {
      lcd_envia_byte(0,0x0c);
      lcd_pos_xy (1, 1); // Posiciona cursor no LCD
      printf (lcd_escreve, " TCC - Vinicius ");
      lcd_pos_xy (1, 2); // xxxxxxxxxxxxxxxx   Posiciona cursor no LCD
      printf (lcd_escreve, "Carregando Robo ");
    }else {
      /*================= Atualiza display LCD  ==================== */
      if(bit_test(ps_dig2,2)) // Reseta tudo
      {
        lcd_pos_xy (1, 1); // Posiciona cursor no LCD
        printf (lcd_escreve,  "%c%c%c%c%c Vel%u
Mem%u",modo[0],modo[1],modo[2],modo[3],modo[4],vel,n_mem);
        lcd_pos_xy (1, 2); //  xxxxxxxxxxxxxxxx Posiciona cursor no LCD
        printf (lcd_escreve, " X%4ldY%4ldZ%4ld",x,y,z);
      }
      /*================ Botão L1 ==================== */
      if(!bit_test(ps_dig2,2)) // Reseta tudo
      {
        lcd_pos_xy (1, 1); // Posiciona cursor no LCD
        printf (lcd_escreve,  "Ang %4ld%4ld%4ld",ang[0],ang[1],ang[2]);
        lcd_pos_xy (1, 2); //  xxxxxxxxxxxxxxxx Posiciona cursor no LCD
        printf (lcd_escreve,  "%4ld%4ld%4ld%4ld",ang[3],ang[4],ang[5],ang[6]);
      }
    }
//!     delay_ms(10);
  }
}

// TCC.H

#include <18F46K22.h>

#FUSES NOWDT                 //No Watch Dog Timer
#FUSES WDT128                //Watch Dog Timer uses 1:128 Postscale
#FUSES HS                 //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for PCD)
#FUSES NOPLLEN
#FUSES FCMEN                //Fail-safe clock monitor enabled
#FUSES IESO                 //Internal External Switch Over mode enabled
#FUSES PUT                 //No Power Up Timer
#FUSES NOBROWNOUT              //No brownout reset
//!#FUSES WDT_NOSLEEP
#FUSES PBADEN                //PORTB pins are configured as analog input channels on RESET
```

```
#FUSES TIMER3C0
#FUSES CCP2D2
#FUSES MCLR                 //Master Clear pin enabled
#FUSES STVREN               //Stack full/underflow will cause reset
#FUSES NOLVP                //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOXINST              //Extended set extension and Indexed Addressing mode disabled (Legacy
mode)
#FUSES NODEBUG              //No Debug mode for ICD
#FUSES NOPROTECT            //Code not protected from reading
#FUSES NOCPB                //No Boot Block code protection
#FUSES NOCPD                //No EE protection
#FUSES NOWRT                //Program memory not write protected
#FUSES NOWRTC               //configuration not registers write protected
#FUSES NOWRTB               //Boot block not write protected
#FUSES NOWRTD               //Data EEPROM not write protected
#FUSES NOEBTR               //Memory not protected from table reads
#FUSES NOEBTRB              //Boot block not protected from table reads

#use delay(clock=20000000)

#define lcd_rs        pin_a0      // pino rs do LCD
#define lcd_enable    pin_a1      // pino enable do LCD
#define lcd_d4        pin_a2      // pino de dados d4 do LCD
#define lcd_d5        pin_a3      // pino de dados d5 do LCD
#define lcd_d6        pin_a4      // pino de dados d6 do LCD
#define lcd_d7        pin_a5      // pino de dados d7 do LCD

#define data_ps       pin_d0
#define comm_ps       pin_d1
#define att_ps        pin_d2
#define clk_ps        pin_d3

#define motor1        pin_b0
#define motor2        pin_b1
#define motor3        pin_b2
#define motor4        pin_b3
#define motor5        pin_b4
#define motor6        pin_b5
#define motor7        pin_c2
#define motor8        pin_c3

// FUNCOES.C


int play (int ps_out)  // lê controle de playstation
{
// bit_test(ps_dig1,0) select
// bit_test(ps_dig1,1) l3
// bit_test(ps_dig1,2) r3
// bit_test(ps_dig1,3) start
```

```
// bit_test(ps_dig1,4) up
// bit_test(ps_dig1,5) right
// bit_test(ps_dig1,6) down
// bit_test(ps_dig1,7) left
// bit_test(ps_dig2,0) L2
// bit_test(ps_dig2,1) R2
// bit_test(ps_dig2,2) L1
// bit_test(ps_dig2,3) R1
// bit_test(ps_dig2,4) triang
// bit_test(ps_dig2,5) O
// bit_test(ps_dig2,6) X
// bit_test(ps_dig2,7) quadr
  int ps_in = 0;

  delay_us(2);
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,0));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,0); } else {bit_clear(ps_in,0); } // lendo entrada

  delay_us(2);
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,1));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,1); } else {bit_clear(ps_in,1); } // lendo entrada

  delay_us(2);
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,2));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,2); } else {bit_clear(ps_in,2); } // lendo entrada

  delay_us(2);
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,3));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,3); } else {bit_clear(ps_in,3); } // lendo entrada

  delay_us(2);
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,4));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,4); } else {bit_clear(ps_in,4); } // lendo entrada

  delay_us(2);
```

```
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,5));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,5); } else {bit_clear(ps_in,5); } // lendo entrada

  delay_us(2);
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,6));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,6); } else {bit_clear(ps_in,6); } // lendo entrada

  delay_us(2);
  output_low(clk_ps);
  output_bit(comm_ps,bit_test(ps_out,7));  // envia saída
  delay_us(2);
  output_high(clk_ps);
  if(input(data_ps)){bit_set(ps_in,7); } else {bit_clear(ps_in,7); } // lendo entrada

  delay_us(200);

  return ps_in;
}
```

ANEXO B: CÓDIGO G PARA FRESAGEM DA BASE

```
%
O5000 (CHAPA BASE_MILLING)
N100 (COMPENSATION-WEAR)
N102 (REV-0.70)
N104 (FEB-12-2014-9:22:41PM)


N106 (TOOL 1 - DIA 10.)


N1 G90 G17 G40 G80 G00
N108 M06 T1 ()
N110 (PR-faces1)
N112 G00 G54 G90 X63.2502 Y60. S2500 M03
N114 G43 H1 Z10.
N116 Z10.
N118 Z2.
N120 G01 Z-3.1667 F75.
N122 G03 X63.2502 Y60. I-3.2502 J0. F250.
N124 G01 X68.2502
N126 G03 X68.2502 Y60. I-8.2502 J0.
N128 G01 X73.2502
N130 G03 X73.2502 Y60. I-13.2502 J0.
N132 G01 X78.2502
N134 G03 X78.2502 Y60. I-18.2502 J0.
N136 G01 X83.2502
N138 G03 X83.2502 Y60. I-23.2502 J0.
N140 G01 X88.2502
N142 G03 X88.2502 Y60. I-28.2502 J0.
N144 G01 X93.2502
N146 G03 X93.2502 Y60. I-33.2502 J0.
N148 G01 X98.2502
N150 G03 X98.2502 Y60. I-38.2502 J0.
N152 G01 X103.2502
N154 G03 X103.2502 Y60. I-43.2502 J0.
N156 G01 X108.2502
N158 G03 X108.2502 Y60. I-48.2502 J0.
N160 G00 Z10.
N162 X63.2502
N164 Z-1.1667
N166 G01 Z-6.3333 F75.
N168 G03 X63.2502 Y60. I-3.2502 J0. F250.
N170 G01 X68.2502
N172 G03 X68.2502 Y60. I-8.2502 J0.
N174 G01 X73.2502
N176 G03 X73.2502 Y60. I-13.2502 J0.
N178 G01 X78.2502
N180 G03 X78.2502 Y60. I-18.2502 J0.
N182 G01 X83.2502
N184 G03 X83.2502 Y60. I-23.2502 J0.
N186 G01 X88.2502
N188 G03 X88.2502 Y60. I-28.2502 J0.
N190 G01 X93.2502
N192 G03 X93.2502 Y60. I-33.2502 J0.
N194 G01 X98.2502
```

```
N196 G03 X98.2502 Y60. I-38.2502 J0.
N198 G01 X103.2502
N200 G03 X103.2502 Y60. I-43.2502 J0.
N202 G01 X108.2502
N204 G03 X108.2502 Y60. I-48.2502 J0.
N206 G00 Z10.
N208 X63.2502
N210 Z-4.3333
N212 G01 Z-9.5 F75.
N214 G03 X63.2502 Y60. I-3.2502 J0. F250.
N216 G01 X68.2502
N218 G03 X68.2502 Y60. I-8.2502 J0.
N220 G01 X73.2502
N222 G03 X73.2502 Y60. I-13.2502 J0.
N224 G01 X78.2502
N226 G03 X78.2502 Y60. I-18.2502 J0.
N228 G01 X83.2502
N230 G03 X83.2502 Y60. I-23.2502 J0.
N232 G01 X88.2502
N234 G03 X88.2502 Y60. I-28.2502 J0.
N236 G01 X93.2502
N238 G03 X93.2502 Y60. I-33.2502 J0.
N240 G01 X98.2502
N242 G03 X98.2502 Y60. I-38.2502 J0.
N244 G01 X103.2502
N246 G03 X103.2502 Y60. I-43.2502 J0.
N248 G01 X108.2502
N250 G03 X108.2502 Y60. I-48.2502 J0.
N252 G00 Z10.
N254 X95.0006
N256 Z-7.5
N258 G01 Z-10. F75.
N260 G03 X95.0006 Y60. I-35.0006 J0. F250.
N262 G01 X98.2502
N264 G03 X98.2502 Y60. I-38.2502 J0.
N266 G01 X103.2502
N268 G03 X103.2502 Y60. I-43.2502 J0.
N270 G01 X108.2502
N272 G03 X108.2502 Y60. I-48.2502 J0.
N274 G00 Z10.
N276 X95.0006
N278 Z-8.
N280 G01 Z-13. F75.
N282 G03 X95.0006 Y60. I-35.0006 J0. F250.
N284 G01 X97.4991
N286 G03 X97.4991 Y60. I-37.4991 J0.
N288 G01 X102.4991
N290 G03 X102.4991 Y60. I-42.4991 J0.
N292 G01 X107.4991
N294 G03 X107.4991 Y60. I-47.4991 J0.
N296 G00 Z10.
N298 X95.0006
N300 Z-11.
N302 G01 Z-16. F75.
N304 G03 X95.0006 Y60. I-35.0006 J0. F250.
N306 G01 X97.4991
N308 G03 X97.4991 Y60. I-37.4991 J0.
N310 G01 X102.4991
N312 G03 X102.4991 Y60. I-42.4991 J0.
N314 G01 X107.4991
```

```
N316 G03 X107.4991 Y60. I-47.4991 J0.
N318 G00 Z10.
N320 X56.3067 Y59.2674
N322 Z-7.5
N324 G01 Z-12. F75.
N326 X59.6819 Y58.5347 F250.
N328 G03 X60.3181 Y58.5347 I0.3181 J1.4653
N330 G01 X67.0686 Y60.
N332 X60.3181 Y61.4653
N334 G03 X59.6819 Y61.4653 I-0.3181 J-1.4653
N336 G01 X52.9314 Y60.
N338 X56.3067 Y59.2674
N340 G00 Z10.
N342 M05
N344 G00 G28 G91 Z0
N346 G00 G28 G91 X-15.0 Y0.
N348 G90
N350 M06 T1
N352 M30
%
```

ANEXO C: CIRCUITO DE SIMULAÇÃO

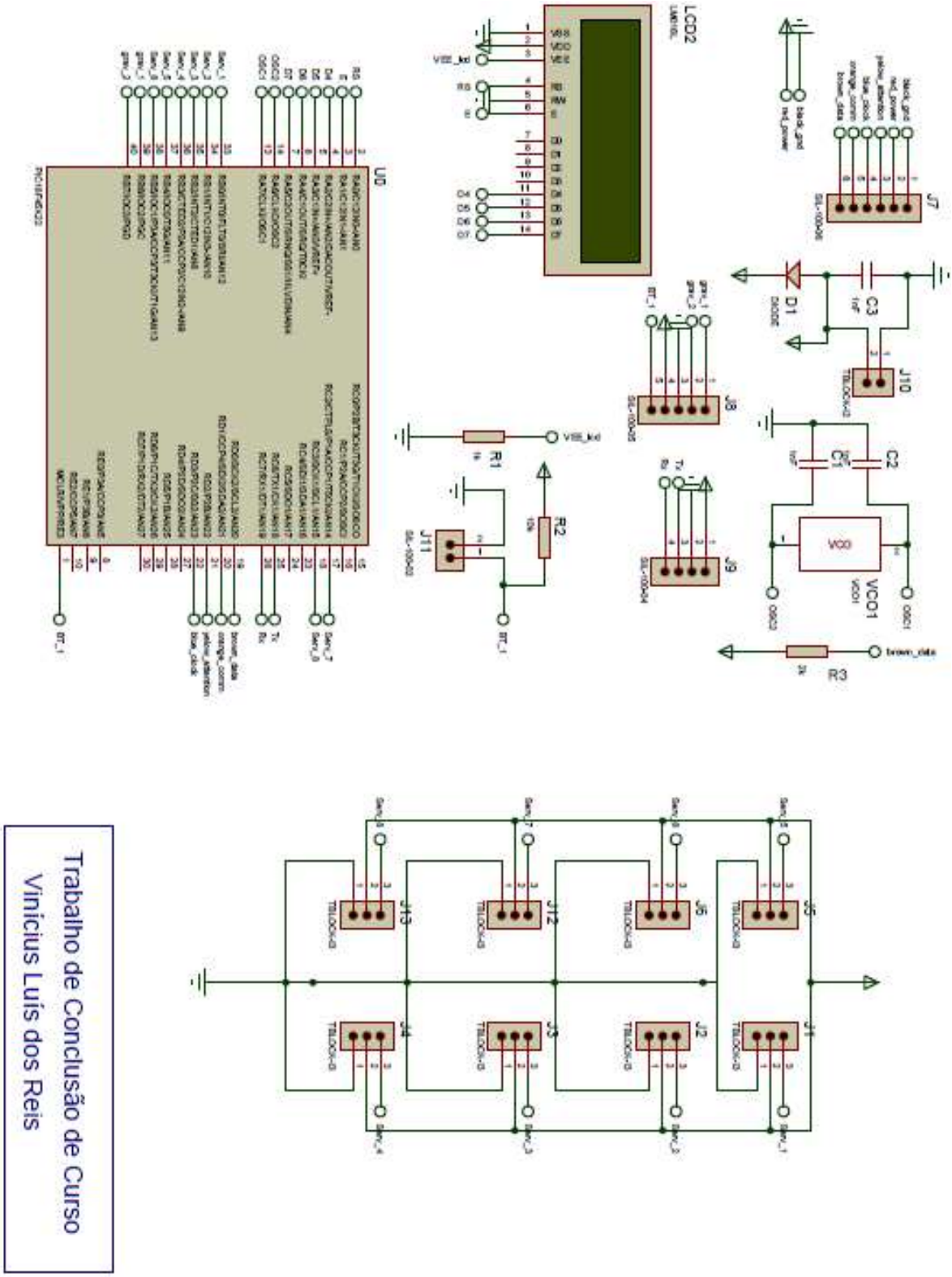

Figura A.0.1: Circuito montado para simulação