8,8 (oito e oito)

Departamento de Engenharia Mecatrônica

PMC 581 - Projeto Mecânico II- Relatório Final

Automação de um goniômetro de cinco graus de liberdade

Orientador: Prof. Dr. Celso Furukawa

Alunos:

Daniel Olioni Andersson **n°USP 2368345**

Rodrigo de Deus Reinaldo **n°USP 2368992**

Abstract

Not only factories or productive process can be automated. This work is about a particle accelerator goniometry automatization. The costs involved in the operation and maintenance of the particle accelerator are high enough to consider that any time saved during its operation is worth. Considering this, to make its operation automatic, and so saving time, can be a very good choice. But not only the economical factor is a reason enough to start this work. Another reason is the fact that with the automatic motion of the goniometry, experiments can be remade with more similarity comparing when it was done by the first time. It is a very important thing when you think that the work made in a particle accelerator is concerned with research and science.

To make the goniometry motion automatic, stepping motors were used and a man machine interface has been made to control them. A proper input language has been made as an alternative to control it.

1.Introdução

O Laboratório de Análise de Materiais por Feixes Iônicos (LAMFI) localiza-se no Instituto de Física da Universidade de São Paulo e dedica-se a desenvolver e aplicar técnicas de feixes de íons para análise de materiais e filmes finos. É resultado do esforço conjunto de pesquisadores da Universidade de São Paulo que desde os anos setenta usam técnicas nucleares para análise de poluição do ar e amostras ambientais, semicondutores e filmes finos. O laboratório tem como principal equipamento um acelerador de partículas Pelletron 1.7 MV conectado a duas câmaras de análise, tendo como principais técnicas de análise RBS (Rutherford Backscattering Spectrometry) e PIXE (Particle Induced X-ray Emission). O LAMFI é um laboratório aberto supervisionado por um comitê científico formado por cinco representantes do Instituto de Física da USP e dois representantes da Escola Politécnica da USP. O LAMFI conta com uma equipe técnica cujo coordenador é indicado pelo comitê científico. Para a utilização do laboratório basta fazer a solicitação diretamente ao comitê científico ou ao coordenador técnico.

Figura 1- Laboratório de Análise de Materiais por Feixes Iônicos

O LAMFI possui o respaldo técnico e científico do Laboratório Pelletron do Departamento de Física Nuclear, do Laboratório do Acelerador Linear, ambos do Instituto de Física, contanto também com grande colaboração do Laboratório Van der Graaff da Pontifícia Universidade Católica do Rio de Janeiro e do Departamento de Física da Universidade Federal do Rio Grande do Sul.

O LAMFI foi primeiramente instalado em 1992 no nono andar do prédio do acelerador Pelletron, aguardando o término das obras das instalações atuais, sendo essas localizadas no edifício Van der Graaff.

O acelerador é do tipo Pelletron 1.7MV NEC 5SDH com stripper gasoso por nitrogênio. Este tipo de equipamento mantém sempre sua região central com um potencial positivo elevado, em torno de 1.7 milhões de Volt e suas extremidades aterradas. Este potencial é gerado através de um gerador Van der Graaff.

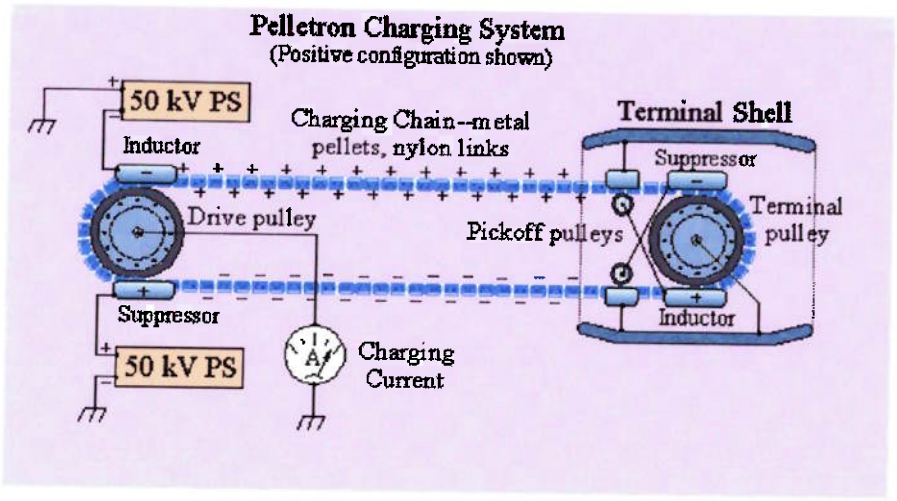


figura 2 – Funcionamento do acelerador

Duas fontes de íons são responsáveis pela geração do feixe de íons negativo:

RF -(Radio Frequency) Alphatroz

Este tipo de fonte produz íons positivos. Ao longo de uma linha de vácuo, hélio é injetado em uma garrafa de quartzo; um oscilador de rádio freqüência da ordem de 100KHz conectado a garrafa provoca a ionização do gás ($He \rightarrow He^{+} + e^{-}$). Uma

diferença de potencial (2 - 6 KV) é utilizada para expulsar os íons positivos da garrafa de quartzo forçando-os a passar por uma pequena abertura formando um feixe contínuo. Para produzir o feixe negativo, o feixe positivo é imediatamente injetado em uma câmara de troca de carga de vapor de rubídio.

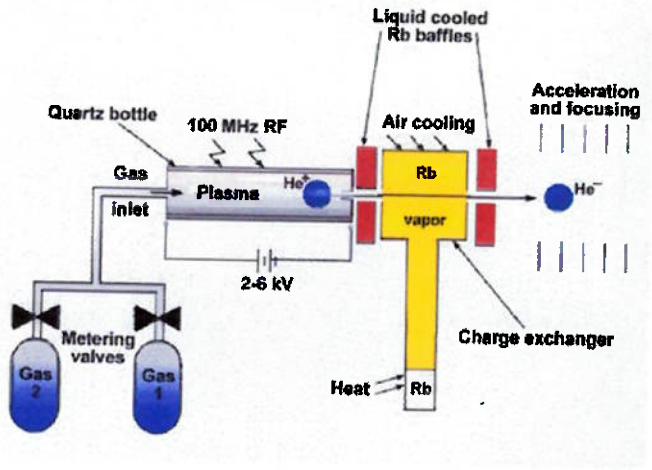


figura 3 – Fonte de íons RF - Alphatroz

Source of Negative Ions by Cesiun Sputtering (SNICS):

Esta fonte produz feixes negativos. Vapor de césio proveniente de um forno é injetado em uma câmara fechada entre um catodo resfriado e um filamento ionizador de alta temperatura. Parte do vapor de césio condensa no catodo e outra parte é ionizada pela alta temperatura do ionizador. O césio ionizado é acelerado contra o catodo provocando espalhamento de partículas da camada do catodo com césio condensado. Alguns materiais irão espalhar preferencialmente partículas neutras ou positivas; ao passarem pela camada de césio condensado absorverão os elétrons disponíveis no césio, produzindo um feixe negativo. Na instalação atual é utilizada para fornecer feixes de prótons e outros elementos(Silício, Carbono) com intensidade de 2μA a 20μA.

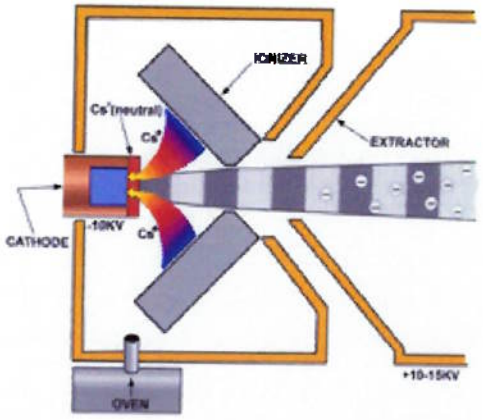


figura 4 – Fonte Snics

O feixe negativo gerado é atraído pelo potencial positivo do acelerador. Na região central do acelerador, onde há o máximo potencial, os íons negativos atravessam uma nuvem de nitrogênio e perdem elétron. Desta forma, os íons se tornam positivos e são repelidos pelo mesmo potencial que os atraiu. Com esse procedimento é possível gerar feixes com uma energia de mais de 3 MeV. Um eletroímã é utilizado qual linha de análise será utilizada ($\pm$ 30° e $\pm$ 15°).

Atualmente estão disponíveis duas estações completas de análise, cada uma com uma câmara de alto vácuo, bombas de vácuo, detectores e eletrônica de aquisição de dados. A estação conectada na linha -15° é usada principalmente para análise de filmes finos através da técnica RBS sendo possível determinar espessura de filmes, difusão de materiais , camadas de oxidação e identificação de contaminantes . A estação RBS (figura 2) é composta por uma câmara de alto vácuo NEC (43 cm de diâmetro interno e 15 cm de altura) com um goniômetro (figura 3) com cinco graus de liberdade (x,y,z,ϕ,θ) e dois detectores de barreira de superfície montados a 170° e 110° em relação ao alvo. Os detectores são montados com colimadores de 3mm de diâmetro e recebem o feixe espalhado que se chocou com a amostra . A aquisição de dados é feita por um PC e a análise e manipulação do espectro é feita através do software Rump.

A estação conectada na linha +30° é utilizada para análises com a técnica PIXE onde são analisadas amostras ambientais (estudo de poluição do ar) e biológicas sendo possível à identificação dos elementos químicos presentes na amostra. A estação PIXE contém uma câmara de alto vácuo (15cm de diâmetro interno e 25cm de altura) de fabricação própria, utilizada desde 1976 para análise de poluição do ar no Instituto de

Física da USP. Utiliza dois detectores de silício/lítio que detectam os raios-X gerado quando o feixe atravessa a amostra. A aquisição de dados é feita por um conversor analógico-digital, ADC ORTEC Spectrum Master, e um multicanal controlado por um PC Pentium. A análise desses espectros de raios-X é feita com o software Axil.

Figura 5 – Câmara RBS

A divisão do tempo de utilização é feita entre os maiores usuários:

- GEPA - Grupo de estudos de poluição do ar (IFUSP), com 50% do uso.
- LSI- Laboratório de Sistemas Integráveis (EPUSP), com 20% do uso.
- LMM - Laboratório de Materiais Magnéticos(IFUSP)., com 20% do uso.
- Outros: pesquisas do IFUSP, principalmente do Professor Dr. Manfredo H. Tabacniks, com 10% do tempo de uso.

Dentre todas as instituições que usam ou usaram o LAMFI, destacam-se:

- Instituto de Física da USP;
- Laboratório de Sistemas Integráveis;
- Núcleo de Pesquisa e Desenvolvimento de Instrumentação Agrícola do Embrapa;
- Instituto de Pesquisa Energéticas e Nucleares(IPEN);

- Universidade de Campinas(UNICAMP);
- Universidade Federal do Rio Grande do Sul (UFRG);
- Pontifícia Universidade Católica do Rio de Janeiro;

2.Objetivos e Descrição do Problema

Nem sempre a automação de um processo traz vantagens que a justifique. Existem situações em que o custo de implantação, manutenção ou até mesmo o baixo custo de se manter um processo não automatizado inviabilize a empreitada.

No caso que é o objeto desse estudo, a automação viria dar uma maior flexibilidade ao processo, tornando a análise de amostras muito mais rápida.

O acelerador fica permanentemente em alto vácuo (10^{-8} mmTorr). Para a inserção de uma amostra para análise na linha RBS, seria necessária a abertura da câmara de análise e conseqüentemente, a perda do vácuo. Após o posicionamento da amostra, seria necessário evacuar o sistema até que ele atingisse o vácuo necessário. Além demorado, esse processo seria impreciso, uma vez que esse posicionamento seria feito manualmente. Para dar agilidade e reprodutibilidade ao processo de análise, implementou-se um goniômetro com cinco graus de liberdade, podendo conter até seis amostras em um único carregamento. Desta forma o usuário montaria seis amostras de uma única vez e para que o feixe atinja a amostra desejada, basta ajustar o posicionamento do goniômetro.

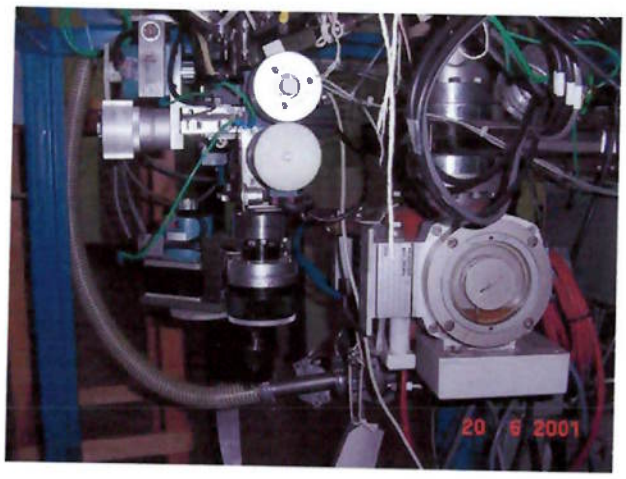


Figura 5 – Manipuladores do gonoômetro.

Existe atualmente implementado um sistema de manipulação dos graus de liberdade do goniômetro que não está funcional.

O objetivo deste trabalho é a automação deste goniômetro possibilitando maior confiabilidade e reprodutibilidade das análises.

Atualmente o laboratório analisa cerca de três mil amostra por ano e existe uma expectativa para um aumento de demanda. Um goniômetro com movimentação automática viria eliminar um gargalo do processo que é representado pelo tempo de posicionamento manual da amostra.

Existe também a questão do custo de se manter um corpo técnico especializado na operação do laboratório. O aumento de demanda traz a necessidade de um número maior de técnicos e que por sua vez traz um custo maior para o laboratório. Tem-se também que lembrar que dentro desse laboratório os espaços físicos são limitados e que existe apenas um acelerador de partículas, isto é, mesmo com o aumento de mão-de-obra em um mesmo período, uma maior produtividade não é alcançada. Isto é bem descrito pela lei dos rendimentos decrescentes da economia.

Em suma, o que se quer dizer é que sendo o acelerador de partículas um equipamento extremamente caro, é necessário que este seja aproveitado da melhor maneira possível e pelo maior tempo possível para que seus custos sejam amortizados justificando sua existência. O que esse trabalho se propõe é tentar extrair o máximo desse acelerador sem que haja um aumento de mão de obra. Uma outra conseqüência

benéfica seria a possibilidade de se expandir à oferta de tempo de máquina a novos usuários.

Quando se aborda o tema de automação industrial sempre se fala, além do aumento de produtividade e redução de custos, que existe um aumento de qualidade e surgimento de novas possibilidades. No caso do laboratório também existe uma situação análoga. No estudo de filmes finos por feixes iônicos existe uma modalidade de análise chamada varredura. A varredura é nada mais do que a movimentação da amostra em intervalos de tempo de tal maneira que o feixe percorra uma certa área da superfície da amostra. O fato é que esse tipo de experimento não é feito atualmente por falta de mecanismo que permitam que isso seja feito de maneira correta. Tendo isso em vista, um segundo e não menos importante objetivo desse trabalho é permitir que esse tipo de análise seja feito.

Serão discutidos a seguir os detalhes técnicos que serão alvo desse trabalho, os objetivos sejam cumpridos e as vantagens de um posicionamento automático frente ao manual.

3.Requisitos do Projeto

- Movimentação do goniômetro com uma precisão de mínimo 0.1mm na translação e 0.5° na rotação.
- Software de controle e monitoração.
- Movimentação do goniômetro através de um joystick.

3.1.Requisitos do software de controle:

- Interface amigável com o usuário.
- Monitoração da posição atual.
- Possibilidade de programação dos deslocamentos e do tempo entre eles (varredura).
- Possibilidade de programação de uma seqüência de irradiação
- Gerenciamento da interface com o joystick
- Definição de uma posição de carga/descarga do porta-amostras.
- Definição de um sistema relativo de coordenadas.

- Indicação de que o atuador está em funcionamento.
- Indicação de que a posição desejada foi atingida.

4.Soluções proposta

4.1.Implementação de um driver de controle/potência.

Nesta solução o comando para movimentação do goniômetro viria do PC sendo este também responsável pela monitoração. Seriam implementados um controlador microprocessado e uma interface de potência. O projeto seria completo com o dimensionamento dos atuadores e projeto do driver incluindo dimensionamento e especificação dos componentes. As placas de circuito impresso também seriam confeccionadas e montadas no decorrer do projeto. Esta é uma solução barata e atenderia especificamente as necessidades do projeto.

4.2.Implementação de um driver de potência e utilização PC como controlador.

Nesta solução o PC seria responsável pela monitoração e controle do movimento. Seria utilizada a saída serial ou paralela para comunicação com o driver de potência. No driver seriam utilizados componentes comerciais. Com esta solução o controle do atuador é feito via software.

4.3.Implementação de um driver de potência e utilização de uma placa de aquisição de dados comercial.

Neste caso a lógica de controle seria feita via software e na interface do PC com o driver de potência que seria implementado utilizar-se-ia uma placa de aquisição de dados de entrada e saída analógicas/digitais programáveis.

4.4.Utilização de uma placa controladora de motores de passo comercial

Placas deste gênero englobam o controle de acionamento de fase e a interface de potência. Esta solução utiliza somente componentes comerciais deixando o projeto com um aspecto mais profissional inclusive no que diz respeito à manutenção e confiabilidade. Estes são aspectos relevantes visto que o projeto será aplicado em um laboratório de prestação de serviços.

5.Solução Escolhida

A solução escolhida foi a que opta pela utilização de uma placa controladora de motores de passo comercial, com interface de potência incorporada. O motivo da escolha desta solução foi a necessidade de utilização de um equipamento com confiabilidade que se adequasse as necessidades de um laboratório. Os custo envolvidos na manutenção e operação do laboratório justificam um investimento em um equipamento de maior confiabilidade.

O software de controle e monitoração, além de atender as necessidades atuais do laboratório, deverá também possuir um módulo de programação que permita ao usuário programar rotinas de maneira que atenda as necessidades do laboratório no futuro

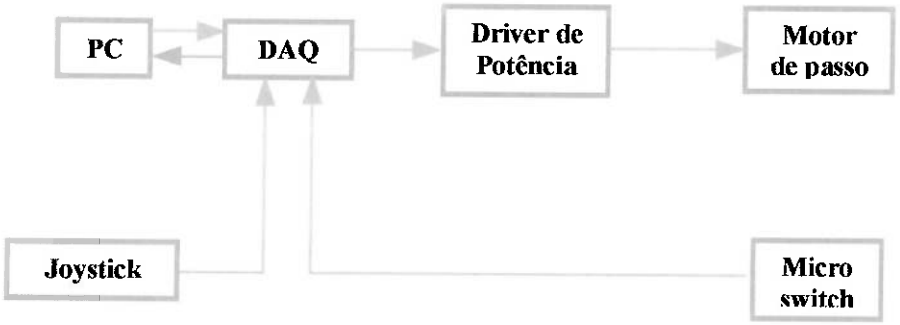


Figura 6 - Diagrama da solução escolhida

6.Especificação do sistema

6.1.Especificações do goniômetro

O goniômetro, como dito anteriormente, possui cinco graus de liberdade como esquematizado na figura 5. A translações x e y possuem deslocamentos máximos de

25mm com uma resolução de 0.005 mm a translação em z possui um deslocamento máximo 250mm e resolução de 0.005mm. Os ângulos ϕ e θ possuem rotação de 360° contínuo com uma resolução angular de 0.1°. Constituído em aço inox e ligas leves, podendo trabalhar entre 1000 bar e 10^{-11} bar a uma temperatura que pode variar de -20°C a 200°C.

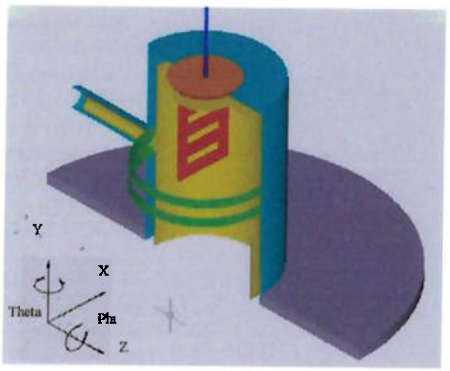


figura 7 –goniômetro e seus graus de liberdade

6.2.Especificação dos atuadores atuais.

- Motores de passo Superior Electric modelo MO61 FD 6008 E,
- Número de fases: 4;
- Corrente por fase: 3.8 A;
- Voltagem por fase: 1.4 V;
- Torque nominal máximo: 35N cm a 400 passos por passos;
- Full step: 1.8°;
- Half step: 0.9°;

Segue em anexo o catálogo do mesmo com mais informações.

Figura 8 – Motor de passo

6.3.Reduções acopladas aos eixos:

Eixo	Relação de transmissão
x, y, z	1:1
θ	90:1
ϕ	45:1

6.4.Velocidades máximas recomendadas:

Eixo	Velocidade	Passos por segundo (em modo half step)
x, y, z	7.5mm/s	3000
θ	40°/s	4000
ϕ	2.5°/s	3000

7.Metodologia de Trabalho

A primeira etapa do projeto foi utilizada para a coleta de todos os parâmetros técnicos do objeto de controle, no caso o goniômetro. Para o correto dimensionamento dos atuadores o torque estático de todos os graus de liberdade foi determinado experimentalmente. Construiu-se uma alavanca em alumínio e com auxílio o de massas

padrão determinou-se o torque estático. Os valores encontrados foram tratados estatisticamente e estão disponíveis na tabela a seguir:

Grau de Liberdade	**X**	**Y**	**Z**	**θ**	**ϕ**
Relação de transmissão	**1:1**	**1:1**	**1:1**	**90:1**	**45:1**
Torque estático(N.cm)	**1.4**	**4.9**	**0.3**	**0.1**	**2.0**

Tabela 1 torque estático

Também foram coletados dados para aferição do dimensionamento dos atuadores existentes. Para tanto, foram solicitadas junto ao fabricante as curvas de torque do motor. Concluiu-se que os atuadores estavam especificados corretamente.

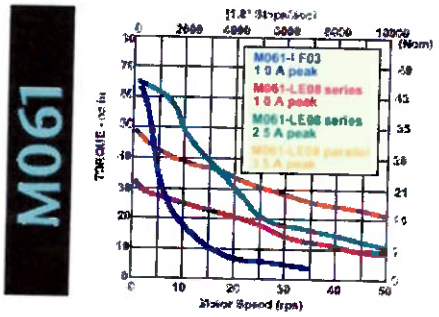


Figura 9 – Curvas de torque x rotação

Implementou-se uma placa controladora de motores de passo e uma interface de potência com componentes comerciais para teste dos motores. A seguir segue o circuito implementado:

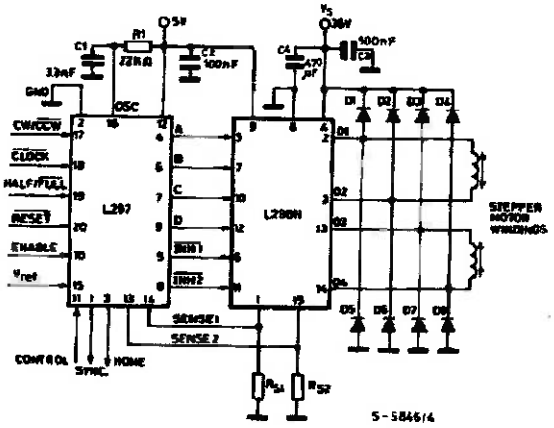


Figura 10 - Circuito de controle e potência

Pesquisou-se placas controladoras de motores de passo e optou-se pelo modelo 3535O Step Motor Driver fabricada pela Applied Motion. Características técnicas:

- Controle de corrente por fase de 0.4A a 3.5A.
- Seleção entre full e half step.
- Redução de 50% de corrente quando ocioso.

Segue em anexo o catálogo da mesma com mais informações.

A placa de aquisição a ser utilizada é fabricada pela Computer Boards, modelo PCI DAS 1001 tendo como principais características:

- Conversor A/D 12 Bits.
- Velocidade máxima de aquisição de 150kHz.
- 16 canais de entrada analógicos absolutos ou 8 canais diferenciais.
- 2 canais analógicos de saída.
- Corrente máxima de saída analógica de 5mA.
- 24 canais digitais configuráveis como entra/saída baseada no 8255 da Intel.
- Corrente máxima na entrada/saída digital de 3mA.

Segue em anexo o catálogo da mesma com mais informações.

8.Especificações de Software

Um importante ponto nesse trabalho é o software. Uma má especificação do mesmo viria a comprometer todo a funcionalidade do sistema e anular todo o aumento de aumento de produtividade que esse trabalho propõe. O software tem duas funcionalidades principais: controle dos motores de passo e do sistema em geral e interface homem máquina.

Dentro do controle do sistema, o software possui três aspectos principais de funcionamento: controle semi-automático, controle programável, e movimentação com o joystick.

8.1.Modo semi-automático de funcionamento

Nesse modo de funcionamento do sistema, o usuário encontrará uma série de funções que de certa forma imitam o funcionamento manual da máquina e além disso, também oferece algumas outras funções que tornam o uso da câmara de análise muito mais flexível. Mas de que forma isso de traduz na prática? Imagine que o usuário da câmara de análise queira fazer um deslocamento no grau de liberdade X. Até então esse deveria de deslocar até a câmara e rotacionar o micrometro do goniômetro referente a esse grau de liberdade e realizar o deslocamento necessário. Com a adoção da automação, o operador terá apenas que entrar através do computador com deslocamento necessário e os motores de passo ficam encarregados dessa movimentação. Perceba que isso não traz apenas uma maior rapidez e flexibilidade ao sistema, isso também traz a possibilidade de os experimentos possuírem uma repetibilidade muito maior.

Agora serão listadas as principais funcionalidades que o usuário encontrará nesse modo de funcionamento.

Deslocamento de um Grau de Liberdade

Essa é a função de utilização mais direta e que servirá de base para a construção das outras funções. Como parâmetros de entrada o usuário deverá fornecer o grau de

liberdade a ser movimentado x, y, z, θ ou ϕ. Em seguida deve-se entrar com o deslocamento de desejado, esse é dado em milímetros no caso dos deslocamentos lineares (x, y e z) e em graus nos deslocamentos angulares.

A movimentação do respectivo grau de liberdade fica condicionada ao curso disponível. Se o deslocamento pretendido pelo usuário atingir uma das chaves de fim de curso instaladas no goniômetro, o sistema pára a movimentação e avisa o usuário que o fim de curso foi atingido.

Estabelecimento de referência

O usuário, além de realizar um deslocamento arbitrário, talvez também queira que o sistema atinja uma determinada posição em relação a uma determinada referência. Sendo assim o software nesse modo de funcionamento deve permitir que o usuário estabeleça uma referência de funcionamento. Assim, com os graus de liberdades determinando uma certa posição do goniômetro, o usuário através de um comando, estabelece que esta posição será a referência de seu sistema.

Estabelecimento de uma posição padrão

Um outro aspecto que faz com que a automação desse goniômetro possua grande vantagem, é a possibilidade de se estabelecer posições padrões do goniômetro. Como já foi discutido anteriormente, já existe uma série de posições padrões de funcionamento do porta-mostra. Sendo assim, o usuário pode estabelecer um conjunto de posições que tornem o seu trabalho mais rápido. Para tal, como parâmetros de entrada, deve-se entrar a posição do porta-mostra em relação a uma determinada referência. Uma vez estabelecida uma determinada posição padrão, basta um simples comando para que essa seja atingida.

Como requisito do sistema, é desejável que haja ao menos a possibilidade de o sistema armazenar seis dessas posições.

Varredura

Como já comentado nesse trabalho, essa função é a que realmente traz algo de novo no sistema. Lembrando que a varredura nada mais é que a movimentação da amostra em intervalos de tempo de tal maneira que o feixe percorra uma certa área da superfície da amostra. Como parâmetro de entrada o usuário deve fornecer a posição inicial do porta-mostra em relação a uma referência em termos de coordenadas X e Y e uma posição final, também em termos de coordenadas X e Y em relação a essa mesma referência. Essa posição inicial (daqui para adiante chamada de Xi e Yi) e a posição final (Xf e Yf) determinam uma região retangular que será chamada de área de varredura. O usuário também deverá entrar com o incremento na direção X e Y entre uma posição e outra (tais incremento agora designados por, respectivamente, Xo e Yo), e o tempo (to) no qual o feixe iônico permanece em um mesmo ponto. Assim, após o usuário entrar com os parâmetros de entrada e iniciar a varredura, o sistema se desloca para a posição inicial (Xi, Yi). Após alvejar a amostra com o feixe iônico pelo tempo to, o sistema se desloca de Xo e inicia uma nova análise de tempo to. Tal procedimento se repete até que o feixe atinja a fronteira da área de varredura na direção X, daí então o sistema se desloca de Yo e recomeça uma nova seqüência de análises só que percorrendo a direção X em sentido contrário. Sempre após um incremento na direção Y o sentido da varredura na direção X se inverte. Isso tudo se repete até que toda área seja varrida.

Cancelamento de função

Eventualmente é necessário que o sistema pare imediatamente de realizar uma das funções das acima citadas por algum determinado motivo, seja uma emergência ou simplesmente porque houve algum erro na entrada dos parâmetros de uma determinada função e o usuário quer que tal procedimento seja abortado. Para isso é interessante que se crie uma função cancelamento que seja atuante em qualquer uma das funções do sistema e que seu efeito seja "congelar" as ações do mesmo.

Função Carga

A posição de carga do porta-amostra é sempre fixa e é definida pelo centro geométrico da câmara de análise. Devido aos aspectos construtivos do porta-amostra, o centro de irradiação não coincide a posição de carga. Sempre que for necessário carregar um novo porta-amostra, a função carga deslocará o posicionador até a posição de carga definida pelo usuário. Esta posição pode ser alterada pelo usuário se forem utilizados porta-amostra geometricamente diferentes.

Função Home

Esta função desloca o posicionador de sua posição atual até a posição de referência definida pela função zero.

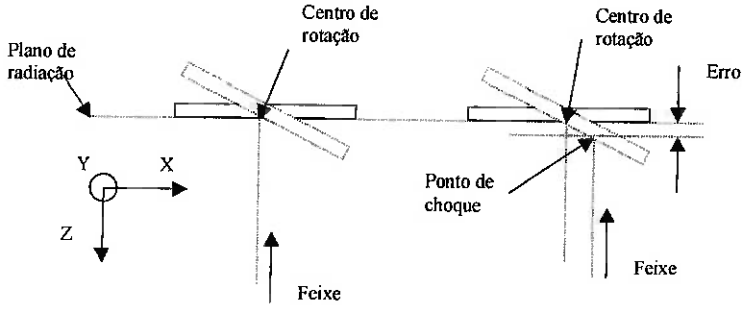
Função Calibração

Através desta função, o posicionador percorre todos os graus de liberdade até atingir os fins de curso que são determinados por micro-switchs. Com esta rotina sempre é possível calibrar o posicionador garantindo maior reprodutibilidade de posicionamento. Desse forma a principal utilidade dessa função é contar qual é o número total de passos realizado pelo motor de passo entre os micro-switchs em dada grau de liberdade. Assim é necessário que se entre com o deslocamento total entre os micro-switchs em milímetros,n o caso dos deslocamentos angulares, e em graus, no caso dos deslocamentos angulares. Como resultado é possível se estabelecer uma relação deslocamento/passo. Deve-se lembrar que o posicionador localiza-se em uma câmara de análise em que o arranjo experimental é constantemente alterado e as referências do sistema podem ser perdidas. Desta forma esta função deverá estar disponível para o usuário

Função Compensação

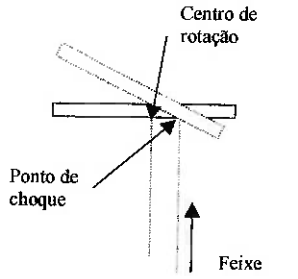
Na maioria das análises, o ponto de contato entre o feixe de íons e as amostras deve estar sempre contido no mesmo plano. Desta forma, as distâncias entre ponto de

choque e os detetores são sempre constantes. Pode-se dizer que o feixe é fixo e portanto o posicionamento da amostra deve garantir esta configuração. Quando a rotação em torno eixo Y é nula não há problemas, a distância é sempre a mesma. Porém, quando a rotação não é nula, o plano que contém o ponto de choque depende do ângulo de giro e da posição do eixo X. Se o ponto de choque coincidir com centro de rotação, o que significa que a posição X é numa, não há problemas. Se a posição X não for nula, quanto maior for o deslocamento no eixo X, maior será a distância entre o eixo Y e o feixe, alterando plano de choque.



Rotação com o ponto de choque coincidindo com o centro de rotação

Rotação com o centro de rotação deslocado em relação ao ponto de choque



Rotação com o centro de rotação deslocado em relação ao ponto de choque com a compensação ativada

A função compensação tem por finalidade corrigir o erro de posicionamento causado pelo deslocamento no sentido do eixo X de forma a garantir que o ponto de radiação esteja sempre no mesmo plano.

8.2.Modo Programável de Funcionamento

Durante a utilização da câmara de análise, verifica-se que o porta-mostra possui uma série de posições fixas e que são freqüentemente utilizadas pelo usuário. Além disso, verifica-se também que a seqüência de deslocamento a ser executada pelo sistema entre uma posição fixa e outra costuma variar pouco. Dessa maneira torna-se interessante criar dispositivos que permitam ao usuário estabelecer rotinas de funcionamento que possam ser realizadas automaticamente. A maneira pela qual o sistema a ser criado pretende atingir esse objetivo é através da criação de um modo de funcionamento em que o usuário possa criar "scripts" em formato texto. Esses scripts ou arquivos podem ser gravados e recuperados todas as vezes que o usuário desejar. Assim cria-se uma espécie de linguagem de programação do sistema em que as funções baseiam-se nas funções já definidas no modo semi-automático. Apesar dessa linguagem estar apoiada no modo semi-automático de funcionamento, algumas funções extras terão que ser definidas, sendo a principal dessas a criação de um timer para estabelecer o tempo que deverá ser transcorrido entre a execução de uma função e outra.

8.3.Modo Manual de Funcionamento(Joystick)

O software deverá permitir que o usuário possa realizar deslocamentos nos eixos através de um joystick. Esse funcionalidade tem como objetivo dar uma maior flexibilidade ao usuário em posicionamentos pouco usuais. Nesse caso, o papel do software seria de monitorar o sistema, informando a posição corrente do porta-mostra e realizar a interface do joystick com o driver.

Uma função de calibração foi introduzida de forma que qualquer joystick possa ser utilizado bastando que o usuário realiza a sua calibração que segue o mesmo modelo de calibração que existe em jogos para PC:

9-Implementação do Software

9.1 - Implementação do Modo Manual de Funcionamento

Nesse tópico será discutido o meio pelo qual o modo manual de funcionamento foi implementado. É importante notar que é através desse modo que o usuário se familiarizará com o sistema, uma vez que nele estão implementadas todas as funcionalidades básicas do funcionamento do sistema automatizado e o uso dessas se dá através da maneira mais direta possível.

9.1.1 - Ambiente de Desenvolvimento

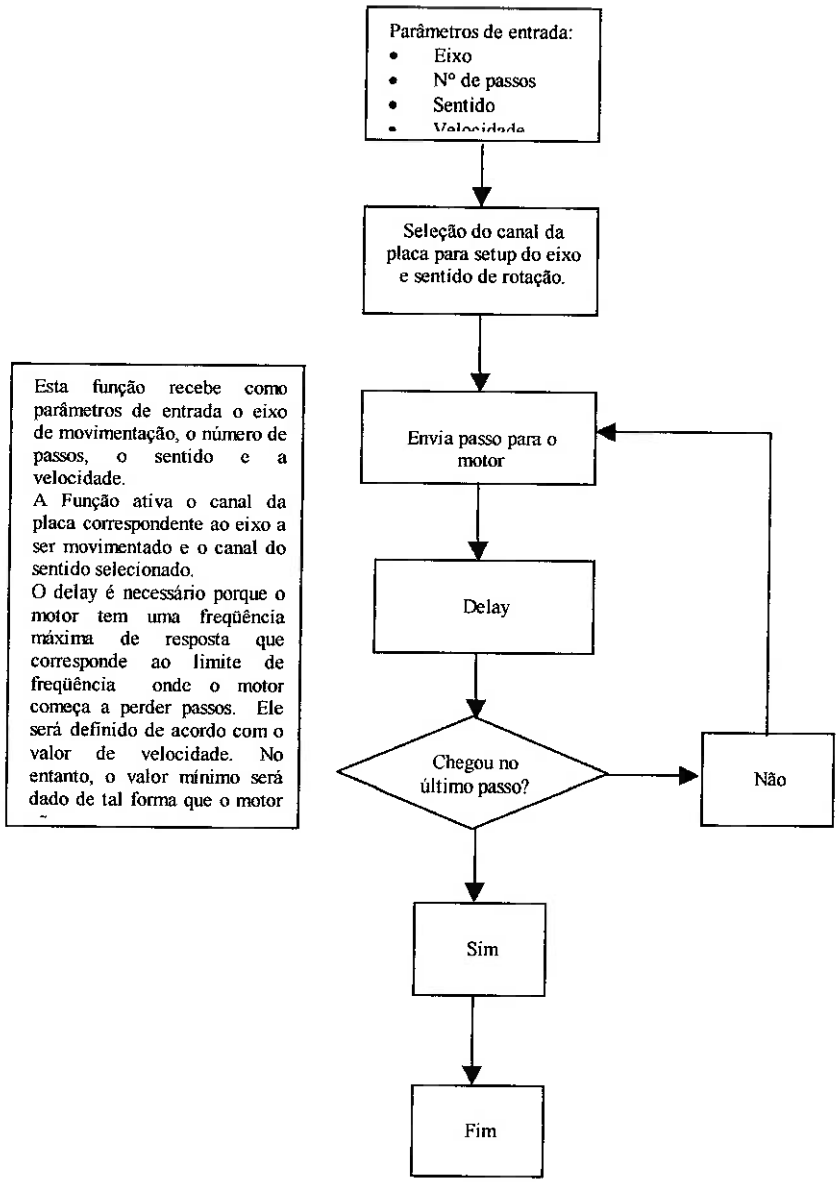
Um dos pontos importantes para a boa evolução do desenvolvimento de um software é a boa escolha da linguagem de programação e ferramentas correlatas. No caso em questão, a linguagem escolhida foi o Visual Basic em associação com uma ferramenta de software chamada Softwire.

O Softwire consiste em uma API para a placa de aquisição de dados de fácil utilização e que funciona integrada ao Visual Basic (VB). Dessa forma, tal fato tornou-se uma motivação muito forte para a utilização do VB como linguagem de desenvolvimento. Um segundo motivo que levou a adoção do VB como linguagem foi a facilidade com que se pode construir uma interface Windows através desse. Visto que esse software irá servir uma vasta gama de pessoas e que essas de modo geral não possuem o mínimo tempo para qualquer tipo de treinamento, torna-se muito interessante a construção de uma interface com o usuário extremamente amigável e de uso muito intuitivo. A melhor maneira para de obter isso foi através da construção do sistema em ambiente Windows e se aproveitando das vantagens que o VB pode oferecer nesse tipo de projeto.

9.1.2 – Implementação das principais funções utilizadas pelo software

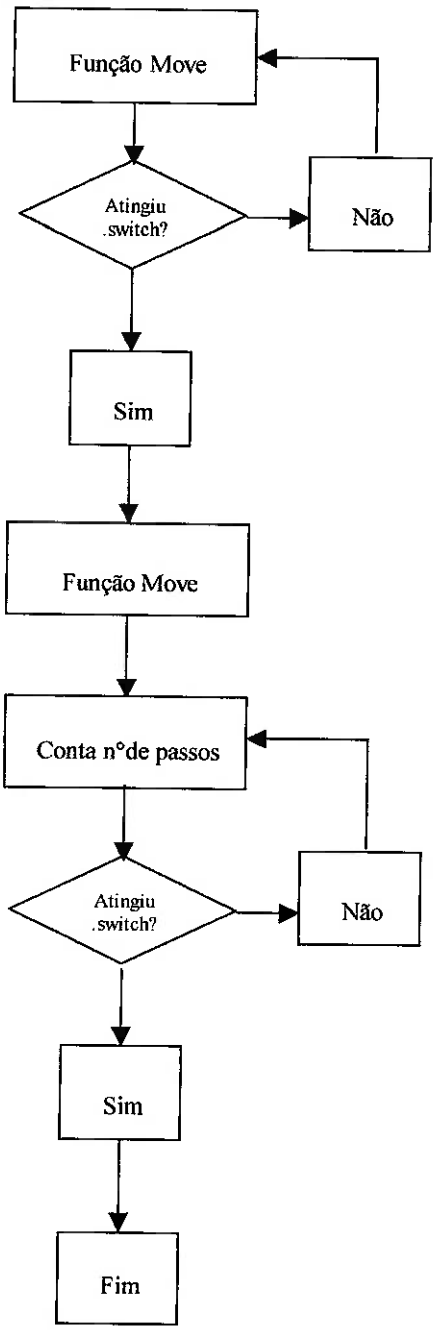
Apesar de o software possuir três modos de funcionamento, esses se utilizam das mesmas funções básicas para realizar o controle. Por exemplo, a função para realizar o deslocamento em um determinado grau de liberdade é utilizada pelos três modos de funcionamento do software. Abaixo está a descrição dessas funções no que se refere a seus algoritmos de funcionamento e suas relações com o hardware do sistema(placa de aquisição, micro-switchs, motores de passo). O objetivo é modularizar o trabalho, tornando-o mais eficiente e flexível no que diz respeito a modificações posteriores. O código das funções implementadas está em anexo.

Função Move

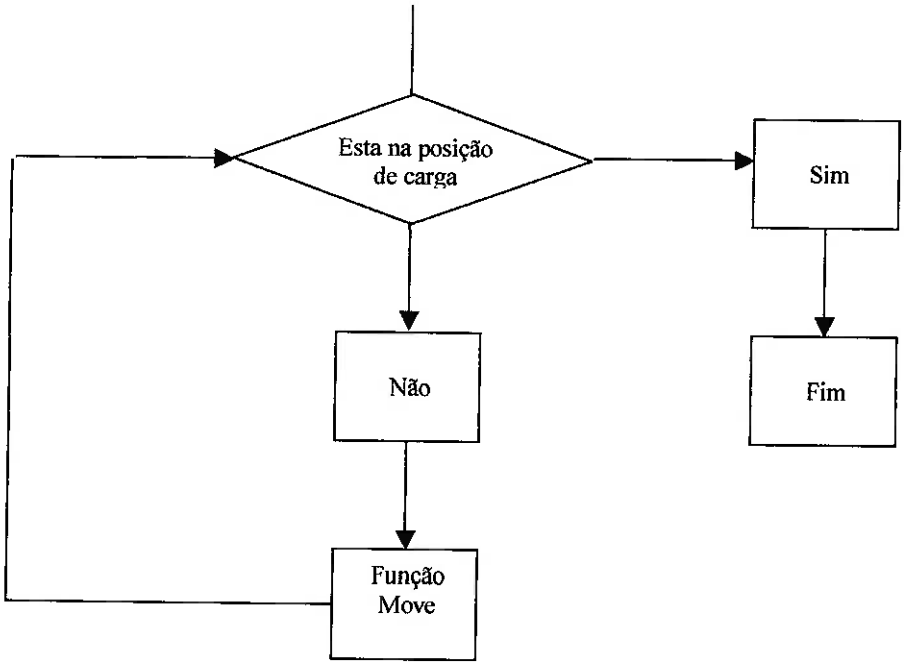


Função Calibração

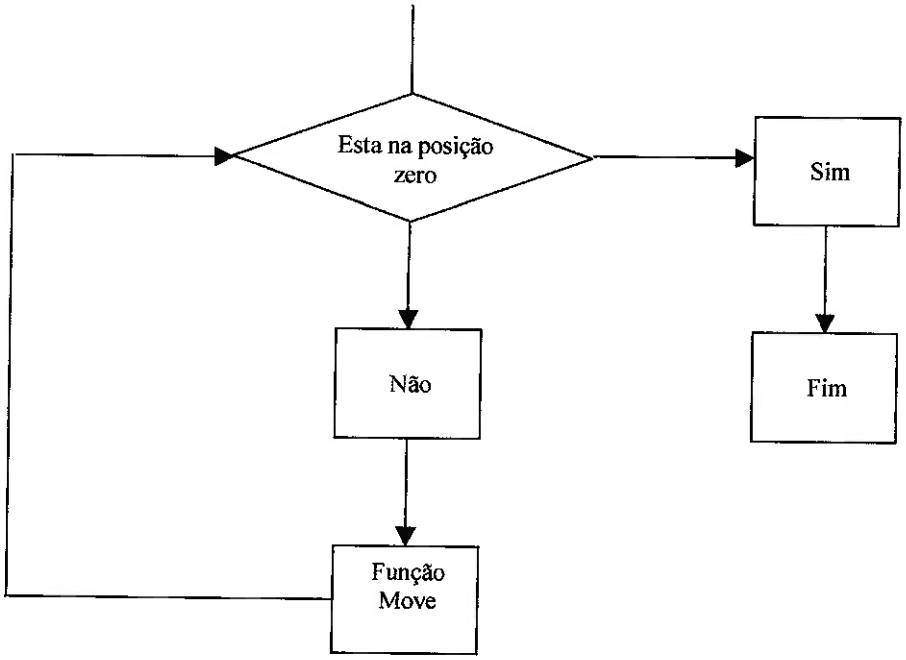
A Função Move recebe com parâmetros os eixos, o número de passos a serem movimentados e o sentido que motor irá se movimentar. No caso da Função Calibração este numero será unitário.
Após atingir o primeiro micro-switch, o sentido de movimentação se inverter e o número de passos é contado até que o micro-switch oposto seja atingido. Desta forma obtém-se a relação deslocamento /passo uma vez que a distância dos μ-switch é conhecida.
A sequência ao lado representa a rotina de calibraçào somente para um eixo. Ela é repetida para todos os cinco eixos.



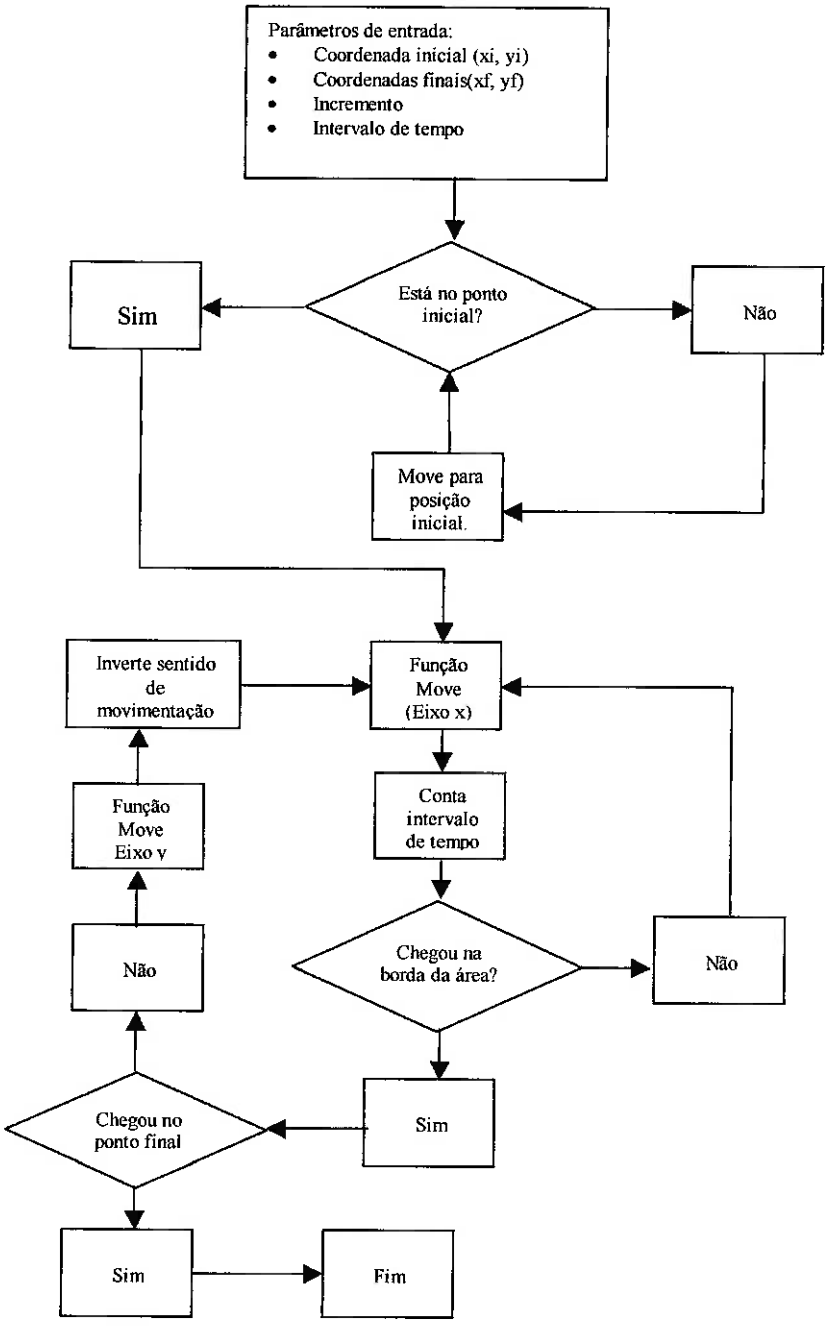
Função Carga



Função home



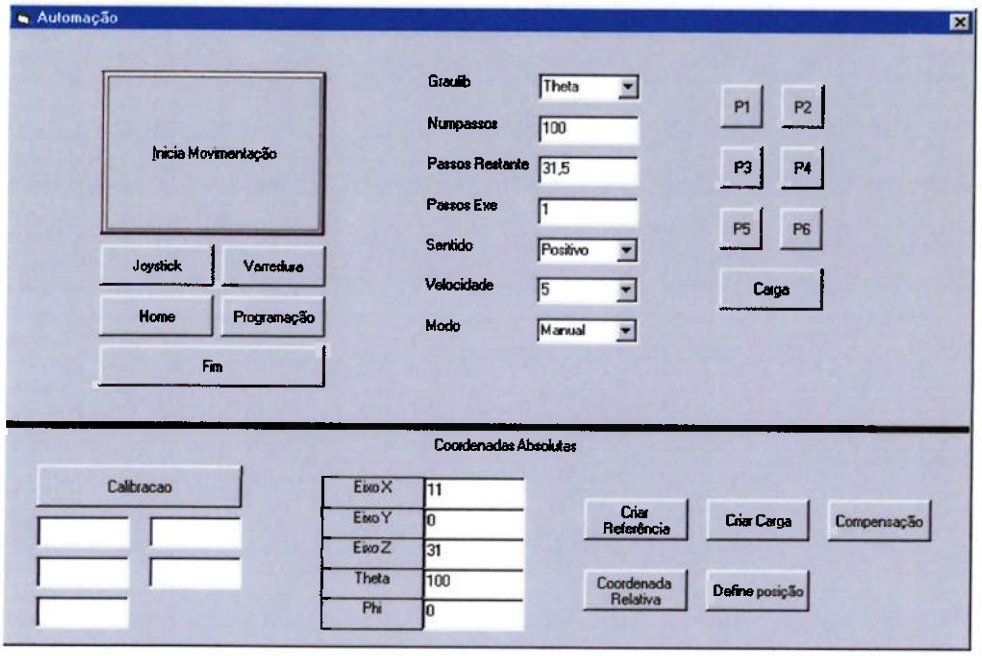
Função Varredura



9.2 - Interface Homem-Máquina

Como já citado no item anterior, a interface homem-máquina foi desenvolvida em ambiente Windows para a maior facilidade de uso. A entrada de dados do usuário se dá principalmente através de caixas de texto e de botões e de controles do tipo combobox. No caso do modo manual de funcionamento, sua interface permite o acesso aos modos de funcionamento através do joystick e programável.

Abaixo temos a tela principal de funcionamento do software. Em seguida haverá uma explicação da função de cada controle e caixa de texto.



Tela principal

Botões:

Inicia Movimentação - Esse botão dá início a movimentação do motor de passo definido pelo combobox Graulib, com sentido de rotação definido pelo combobox Sentido e com velocidade definida através do combobox Velocidade.

Joystick - Permite a calibração do joystick, tal recurso será melhor abordado no tópico sobre o modo de funcionamento através do joystick.

Calibração- Esse botão executa a calibração do sistema. Para cada grau de liberdade, o motor de passo correspondente é acionado até que chegue ao micro-switch do extremo do curso, logo em seguida o motor é acionado em sentido contrário até que chegue no micro-switch do outro extremo. Entre a movimentação entre um fim de curso e outro, o número de passos realizado é contado. Após isso o motor é acionado até que atinja o ponto médio do curso. Lembrando que essa operação é realizada para cada grau de liberdade.

Após a calibração, o sistema permanece em uma posição que é definida como a origem do sistema absoluto de coordenadas, isto é, todos os graus de liberdade possuem por definição deslocamento nulo em relação a posição em questão. Como se verá a seguir, é possível criar um sistema de coordenadas relativas, no entanto, após a execução da calibração, a origem do sistema de coordenadas relativas é deslocada para da origem do sistema de coordenadas absolutas.

O número total de passos realizados entre os fins de curso de cada grau de liberdade é mostrado ao final da execução do comando nas caixas de texto logo abaixo ao botão **Calibração**

Criar referência - Quando acionado, este botão cria uma referência relativa cuja origem é definida como sendo a posição que o goniômetro ocupa no instante de acionamento desse de botão. Esse recurso trás uma série de vantagens para o usuário, pois esse poderá definir o sistema de coordenadas que mais for interessante para seu trabalho. É importante notar que esse comando apenas cria a referência relativa. Essa será apenas efetivamente usada quando o botão **Coordenada Relativa** for acionado, como se verá logo a seguir.

Coordenada Relativa/Coordenada Absoluta - intercambia o sistema entre a utilização do sistema de coordenadas absolutas e relativo. Quando o sistema estiver se utilizando de coordenadas absolutas, no botão haverá a mensagem "Coordenada Relativa", e nas caixas de texto localizados no posição inferior central da tela estarão indicadas as

posições de cada grau de liberdade em relação a esse sistema absoluto. A situação se inverte quando se aciona o botão, isto é, o sistema migra para as coordenadas relativas e no botão haverá a mensagem "Coordenada Absoluta", e nas caixas de texto estarão indicadas a posição do goniômetro em relação ao sistema relativo.

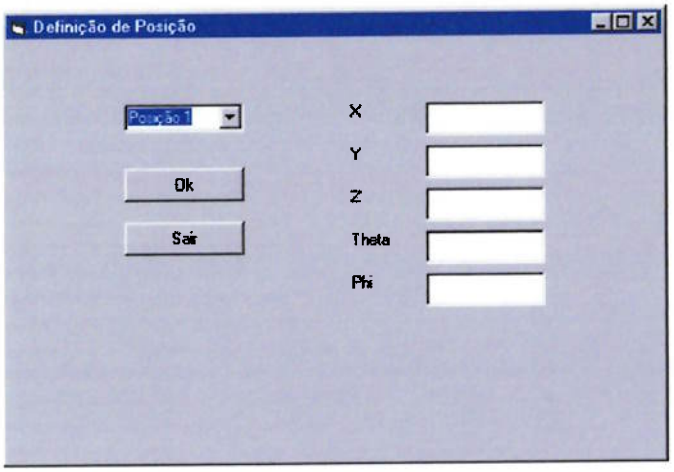
Home - Retorna o goniômetro para a posição de origem do sistema de coordenadas. Se o sistema estiver usando coordenadas absolutas, o sistema volta para a origem do sistema absoluto. Se estiver usando o relativo, retorna para a origem do sistema de coordenadas relativas.

Criar Carga – Uma importante função do software é permitir o rápido acesso do goniômetro a posições que são utilizadas com freqüência. A posição de carga do sistema é uma delas. A posição de carga é a posição que o goniômetro deve permanecer para que esse seja carregado e descarregado de um porta-mostra. O comando **Criar Carga** define a posição de carga. Para tal, basta mover os graus de liberdade do goniômetro até que cheguem na posição desejada de carga, então, bastão pressionar o botão **Criar Carga.**

Carga – Quando acionado movimenta o sistema para a posição definida pelo comando **Criar Carga.**

Define Posição – Não apenas a posição de carga pode ser importante para se necessitar um rápido acesso. Logo, o sistema permite a criação de outras posições padrões. Essas podem ser, por exemplo, posições que geralmente possuem uma amostra a ser analisada no porta-mostra. O software permite a criação de até seis posições pré-definidas.

Quando o usuário aciona o comando **Define Posição**, a tela abaixo é apresentada a ele.



Tela Define Posição

Através desse formulário, é possível definir quais serão as posições padrão. Usando o controle combobox, indica-se qual é a posição (de um a seis) que será definida. Após isso, o usuário entra com as coordenadas dessa posição padrão nas caixas de textos correspondentes ao graus de liberdade. Quando o botão **ok** é pressionado as informações são gravadas. Ao acionar o botão **Sair**, esse formulário sai da tela.

P1 - Movimenta o goniômetro para a primeira posição pré-definida.

P2 - Movimenta o goniômetro para a segunda posição pré-definida.

P3 - Movimenta o goniômetro para a terceira posição pré-definida.

P4 - Movimenta o goniômetro para a quarta posição pré-definida.

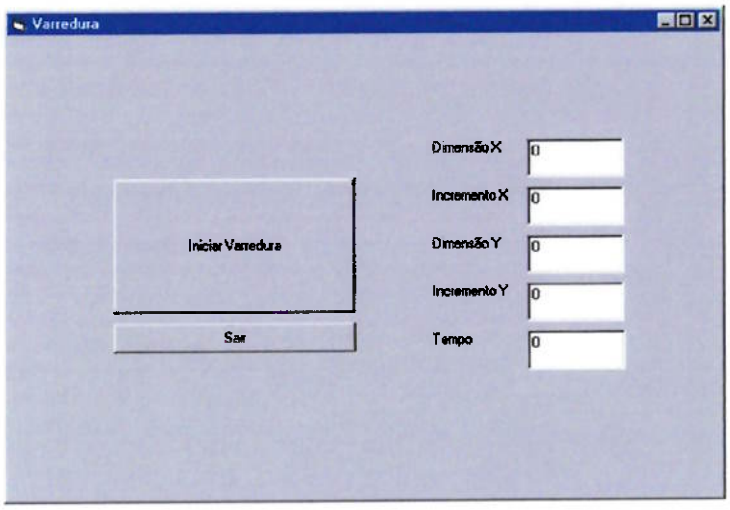
P5 - Movimenta o goniômetro para a quinta posição pré-definida.

P6 - Movimenta o goniômetro para a sexta posição pré-definida.

Varredura – Permite que o usuário faça uma operação de varredura conforme os moldes já vistos. A explicação mais detalhada de como o software realiza a operação de

varredura os parâmetros necessários e o que esses vem a ser será mais bem explicado no tópico da implementação do modo programável de funcionamento. Agora o texto se deterá mais em como o usuário pode realizar essa operação no modo manual.

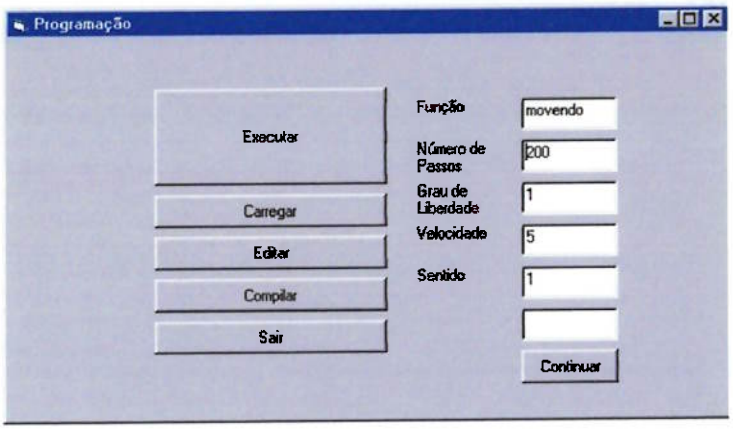
Quando a opção da varredura é executada, a seguinte formulário fica disponibilizado:



Tela Varredura

Nas caixas de texto é possível entrar com os parâmetros para se realizar a varredura, logo após isso, basta "clicar" no botão **Iniciar Varredura** para que a operação seja executada.

Programação – Ativa a tela de programação permitindo ao usuário selecionar o arquivo compilado com a programação de movimentação desejada (Através do botão *"Carregar"*) e iniciar a execução do mesmo (Através do botão *"Executar"*). Também permite que o usuário abra um arquivo texto, podendo editar e compilar este arquivo (Através dos botões *"Editar"* e *"Compilar"*). A seqüência de comando é acompanhada pelas caixas de texto que são atualizadas a cada novo comando.



Tela Programação

Compensação- Ativa e Desativa a função compensação. O nome do botão é atualizado de forma a informar ao usuário em qual modo está operando .

Fim – Finaliza a execução do software.

Caixas de Texto:

Numpassos- Neste campo, o usuário entra com o número de passos que o grau de liberdade deve se deslocar

Passos Restantes – Esta caixa de texto mostra o número de passos restantes durante a movimentação do motor.

Passos Exe - Fornece p número de passo que foram executados. Quando o movimento é completo, este valor é igual ao número de passos do campo NumPassos. Quando ocorre algum imprevisto como por exemplo o contato com um microswitch o valor é parcial.

Combobox:

Graulib - O usuário define qual grau de liberdade irá movimentar (X ,Y, Z, Theta, Phi).

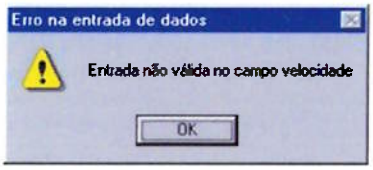
Sentido - O usuário define qual será o sentido de movimentação do grau de liberdade associado (Positivo e Negativo).

Velocidade - O usuário define qual será a velocidade de movimentação dos motores. Estão disponíveis cinco velocidades (1, 2, 3, 4, 5).

Modo- Neste combobox, o usuário define qual será o modo de funcionamento do sistema(Manual, Programável, Joystick)

9.2.1 - Validação de Entradas

Todos os campos de entrada de dados têm associados uma função de validação de entrada. Desta forma se o campo NumPassos que recebe somente inteiros positivos, tiver como parâmetro uma letra ou um número não inteiro, o sistema bloqueia a movimentação do goniômetro e uma mensagem de erro é mostrada ao usuário indicando que a um erro na entrada de dados. O mesmo ocorre com os outros campos de entrada de dados.



Erro na entrada de dados no campo velocidade

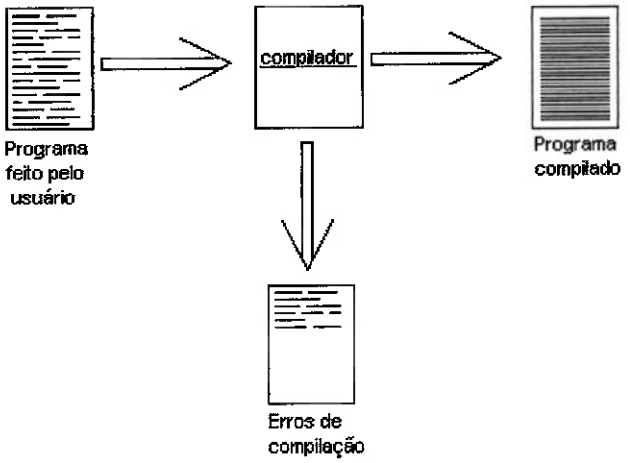
9.3- Implementação do modo programável de funcionamento

Para que o modo programável de funcionamento seja realmente efetivo proporcionando maior flexibilidade e facilidade de operação do sistema, é necessário que uma linguagem de uso simples e de utilização intuitiva seja desenvolvida. Sendo assim, essa linguagem foi concebida com a idéia de que fosse formada apenas por

alguns comandos com seus respectivos parâmetros que tornassem o uso do goniômetro bastante versátil. Esses comandos tentariam traduzir a essência do funcionamento do goniômetro. Para elaborar o programa, bastaria digitar um arquivo com a seqüência de operações desejadas e em seguida compilar o programa. Nas próximas seções será analisada passo a passo a maneira pela qual essa idéia foi implementada.

9.3.1 Funcionamento do modo programável

A figura abaixo ilustra em linhas gerais o funcionamento desse modo. Como já foi citado, o usuário edita um arquivo contendo os comandos da linguagem que irão executar as correspondentes ações de movimentação do goniômetro.



Após a elaboração desse arquivo de entrada, o usuário aciona um programa de compilação desse código fonte. Esse programa será chamado doravante de compilador. Esse compilador cria dois arquivos. Um contém as instruções do usuário em formato no qual o programa principal do software consegue interpretar e realizar os comandos. Um segundo arquivo é gerado contendo os possíveis erros encontrados ao longo do código fonte elaborado pelo usuário. Esses erros podem ser tanto de sintaxe, bem como a inserção incorreta de parâmetros nas funções. Juntos dessas indicações de erros estão as respectivas linhas do programa onde esses ocorrem.

Linguagem de programação.

Essa seção tem por objetivo descrever a sintaxe e comandos da linguagem criada.

Inicialmente serão descritos os comandos. Convenciona-se aqui que na descrição dos comandos, o que está em **negrito**, corresponde à forma sintática do comando, e que está em *itálico*, corresponde aos parâmetros de entrada do comando.

Comando move

Esse comando tem por finalidade a movimentação de um dos graus de liberdade do goniômetro. Possui a seguinte forma:

move (*número de passos* , *grau de liberdade* ,*sentido de movimentação* , *velocidade*);

número de passos- valor necessariamente inteiro e não negativo que corresponde ao número de passos a serem executados pelo motor de passo respectivo ao grau de liberdade escolhido para movimentação.

grau de liberdade- parâmetro que define qual grau de liberdade será movimentado. Esse pode assumir um valor inteiro maior ou igual a um ou menor ou igual a cinco, sendo que cada valor irá representar um grau de liberdade da seguinte forma:

1 - para movimentação do grau de liberdade X

2 - para movimentação do grau de liberdade Y

3 - para movimentação do grau de liberdade Z

4 - para movimentação do grau de liberdade θ

5 - para movimentação do grau de liberdade ϕ

sentido de movimentação- parâmetro que define qual será o sentido de movimentação do motor de passo especificado pelo parâmetro *grau de liberdade*. Pode assumir os seguintes valores com os seguintes efeitos:

0-rotação no sentido anti-horário.

1- rotação no sentido horário.

velocidade – define a velocidade de rotação do motor de passo. No sistema em questão a velocidade em cinco valores possíveis. Para cada velocidade existe um valor diferente para esse parâmetro, sendo que esses podem ser 1 ,2, 3, 4 ou5. Quanto maior o valor desse parâmetro maior é a velocidade.

Comando varre

Esse comando realiza a operação de varredura (nos mesmos moldes já citados). Possui a seguinte forma:

Varre(ΔX, ΔY, δx, δy, ΔT);

{

ΔX- número de passos a serem executados pelo motor de passo respectivo ao grau de liberdade X, que corresponderá a uma extensão total de deslocamento no eixo X da área de varredura. Esse parâmetro é um número inteiro sem restrições de ser negativo ou positivo.

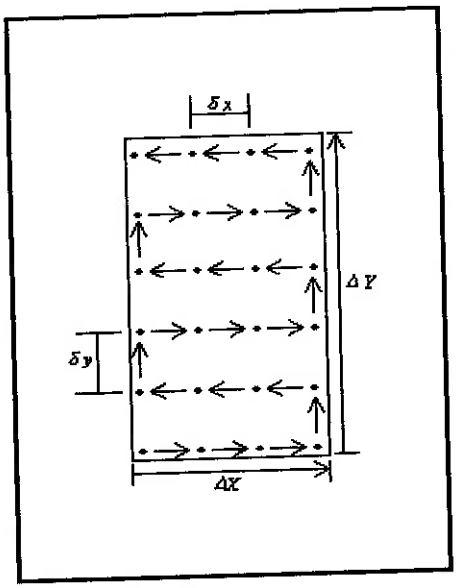
ΔY- número de passos a serem executados pelo motor de passo respectivo ao grau de liberdade Y que corresponderá a uma extensão total de deslocamento no eixo Y da área de varredura. Esse parâmetro é um número inteiro sem restrições de ser negativo ou positivo.

δx- número de passos a serem executados pelo motor de passo respectivo ao grau de liberdade X, de tal maneira que esse corresponda à distância desejada entre os pontos que serão alvos do feixe iônico, na direção X.

δy- número de passos a serem executados pelo motor de passo respectivo ao grau de liberdade Y, de tal maneira que esse corresponda à distância desejada entre os pontos que serão alvos do feixe iônico, na direção Y.

ΔT – tempo total em que um determinado ponto será atingido pelo feixe iônico, isto é, o tempo entre o deslocamento entre um ponto e outro.

Sendo assim, a partir de uma certa posição inicial do goniômetro, o sistema executa uma movimentação permitindo que o feixe iônico realize uma varredura numa certa área do plano XY de dimensões ΔX na direção X e ΔY na direção Y, de forma que o feixe irá atingir um conjunto de pontos na mostra espaçados de forma igual de δx na direção X e de δy na direção Y . Cada ponto será irradiado por um período de ΔT. Para um melhor entendimento, se imaginássemos que o feixe estivesse incidindo constantemente sobre a placa do porta-mostra que o goniômetro movimenta teríamos uma trajetória "marcada" nessa placa conforme mostra figura abaixo.



Seqüência de movimentação na varredura

Cada ponto na figura acima corresponde a um ponto no qual o feixe iônico incidirá pelo período ΔT. A flechas indicam o sentido de movimentação. Nessa figura, as distâncias estão representadas pelo número de passos realizados pelos atuadores que gerarão as distâncias reais. Para se saber essas ditâncias reais é necessário saber qual é o deslocamento real (linear ou angular) correspondente a um passo. Isso vai depender da precisão do motor e do sistema de reduções. É importante salientar que o goniômetro possui uma certa posição nos outros graus de liberdade (Z,θ e Φ) que permanece inalterada ao longo da operação de varredura.

Comando tempo

Comando que proporciona atraso entre operações consecutivas. Sem esse comando, uma certa ação determinada pelos outro comandos é executada sempre logo em seguida a uma outra. Eventualmente essa situação pode não ser conveniente. Possui a seguinte sintaxe:

tempo(ΔT **);**

ΔT- representa o tempo de atraso em segundos. Esse parâmetro apenas assume valores inteiros de tempo.

Comando calibra

Esse comado que realiza a calibração do sistema. Essa função não aceita parâmetros uma vez que sua ação já é pré-definida conforme os mesmos moldes da função calibra do modo automático de funcionamento. Sua sintaxe é dada da seguinte forma:

calibra;

Comando home

Retorna o sistema para a origem das coordenadas definido pelo comando **calibra**. Essa função também não aceita parâmetros. Possui a seguinte sintaxe:

home;

Comando botao

Comando que para a execução do programa até que um botão definido seja acionado na tela do modo programável de funcionamento do software principal. Sua sintaxe é:

botao;

Comando irradia

Comando que aciona o feixe iônico para que este incida sobre a amostra. Possui como parâmetro o tempo em segundos que o feixe permanecerá acionado.Sua sintaxe é:

irradia(*Δ T*);

Δ T- representa o tempo que o feixe iônico permanecerá acionado.

Comentários

Também é possível fazer comentários para que haja maior clareza e facilidade para se realizar modificações posteriores da programação. Para se fazer um comentário no programa, a seguinte sintaxe deve ser obedecida:

```
/*comentário*/
```

Operações aritméticas

Caso o usuário deseje, é possível realizar contas aritméticas dentro dos parâmetros das funções. As operações de soma e subtração possuem precedência sobre as operações de divisão e multiplicação. Também é possível o uso de parênteses nas operações aritméticas, nesse caso, as expressões dentro dos parênteses terão precedência sobre as demais. Exemplo de uso:

```
move((100-5)*3,3,0,2);
```

Exemplo de programação

Agora será fornecido um exemplo de programação usando a linguagem desenvolvida.

```
/*inicio da programação*/
calibra;
move(100,2,1,1);
tempo(10);
varre(100+10,100+20,10,10,35);
home;
/*final da programação*/
```

Elaboração do Compilador

O ponto principal na elaboração da linguagem é a construção de um compilador. Como já descrito anteriormente, é o compilador quem vai transformar o programa do usuário em um texto padrão interpretável pelo programa principal. Para tanto, duas

ferramentas importantes foram usadas nessa tarefa: uma delas é o criador do analisador léxico, *Flex* , e a outra, o criador de análise de gramática de linguagens *Bison.* Abaixo segue uma breve explicação do que vem a ser cada uma dessas ferramentas e de como é possível criar uma linguagem com esses instrumentos.

Flex

O Flex é um programa que gera um código escrito em linguagem C que quando compilado cria um executável que funciona como um analisador léxico, ou seja, realiza uma busca de padrões léxicos em um certo texto desejado pelo usuário.

Para utilizar o Flex, primeiramente, deve-se criar um texto de entrada que contenha as informações correspondentes aos padrões léxicos que o analisador léxico irá procurar. Essas informações são fornecidas por meio de uma linguagem própria do *Flex*. Cada vez que o analisador léxico encontra um certo padrão, esse realiza uma certa ação correspondente, que de modo geral, é definida em linguagem C. Abaixo temos um exemplo de texto de entrada para o Flex que irá procurar padrões léxicos de números inteiros:

```
-?[0-9]+
```

Em uma breve explicação do texto acima, pode–se dizer que 0-9 entre colchetes significa que o analisador léxico irá procurar um ou mais caracteres no código ASCII entre 0 e 9 consecutivos e podendo ser antecedidos de um sinal negativo. É o operador `+'que define que serão um ou mais caracteres consecutivos. A possibilidade de o sinal de menos ser facultatico provem do fato de haver um operador `?' . Caso o operador `?' não existisse, apenas números inteiros negativos se encaixariam no padrao léxico.

Criando uma linguagem e a ferramenta Bison

No tópico anterior, explicou-se como a ferramenta Flex funciona, mas não se explicou como esse age na hora de se criar uma linguagem de programação. Perceba que o Flex pode ser usado não apenas para se criar uma linguagem mas também, por exemplo, para fazer uma análise estatística de palavras de um texto.

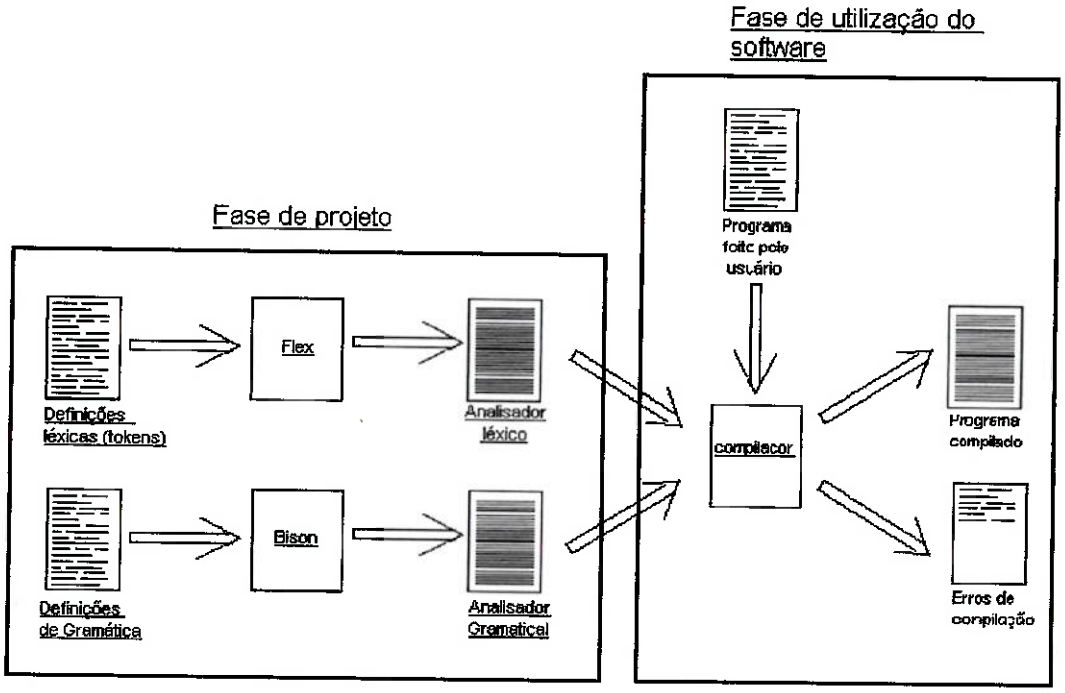
Antes de mais nada é interessante definir como funciona a mecânica da análise de uma linguagem. Para se definir uma linguagem, primeiramente é necessário se definir palavras e uma gramática. A gramática seria a forma pela qual essas palavras se relacionam. Essas palavras serão chamadas de agora em diante de token. Um token também pode ser definido como o menor elemento da linguagem. Da mesma maneira que em uma linguagem humana as palavras possuem um certo significado, computacionalmente, um token também possui um significado, chamado de valor semântico. Percebe-se que dessa maneira o primeiro passo para realizar um analisador de uma linguagem é identificar tokens. É nesse ponto que o Flex é utilizado. É ele quem vai construir a função que irá buscar os tokens num texto e também devolver seu valor semântico.

Com os tokens identificados, é necessário verificar se o seu agrupamento respeita as regras definidas. Como fazer isso? Utiliza-se uma segunda ferramenta: o Bison.

Da mesma maneira que no Flex precisamos definir os padrões léxicos que identificarão os tokens e a partir disso uma função C é definida, o mesmo acontece com o bison; é preciso definir uma gramática e a partir disso essa ferramenta cria uma função C que realizará a análise gramatical e sintática. A gramática é definida de uma maneira formal por meio de uma linguagem própria do Bison. Essa função em C criada pelo Bison, o analisador gramatical, funciona como uma máquina de estados dotada de uma pilha. De uma maneira mais genérica pode-se dizer que em cada estado dessa máquina uma determinada ação é tomada. Por exemplo, supomos que a primeira ação seja requisitar um token do analisador léxico. O valor semântico do token é armazenado na pilha. A partir dessa ação e das regras gramaticais definidas uma ação seguinte é tomada. Essa ação pode ser o requerimento de uma próximo token, ou, caso um certo padrão gramatical seja reconhecido, a redução da pilha, isto é, os tokens correspondentes a esse padrão são retirados da pilha e essa se reduz. Essa ação é muito importante para que a pilha não cresça indefinidamente.

Um outro ponto, é que toda vez que um certo padrão é reconhecido, uma ação definida pelo usuário em linguagem C é realizada. Cada regra gramatical pode ter uma ação correspondente.

O analisador gramatical, construído pelo bison, trabalhando em conjunto com o analisador léxico, construído pelo flex, formam o compilador. A figura abaixo mostra como tudo se passa.



Em anexo, nos apêndice 2 e 3, seguem como foram definidos os padrões léxicos identificados na linguagem que é alvo desse trabalho, e também suas correspondentes regras gramaticais.

Formato do Texto de Saída do Compilador

Como já dito anteriormente, o compilador transformará o programa de entrada do usuário em um novo arquivo no qual o programa principal conseguirá facilmente interpreta-lo. Agora se dará uma explicação de como é o formato desse arquivo. Primeiramente, cada comando da linguagem corresponde a uma linha no arquivo de saída. E cada comando possui um código que corresponde a uma letra. Em seguida a

essa letra código, existem os parâmetros correspondentes a esse comando com um número fixo de caracteres. Abaixo está uma lista de como cada comando aparece no arquivo de saída.

Comando move

m(cinco caracteres para deslocamento)(caractere para grau de liberdade)(caractere para sentido) (caractere para velocidade)

Então para a linha de comando,

```
move(100,2,1,2);
```

Teríamos no arquivo de saída,

```
m000100212
```

Comando varre

v(cinco caracteres para ΔX)(cinco caracteres para ΔY)(cinco caracteres para δx)(cinco caracteres para δy)(quatro caracteres para ΔT)

Então para a linha de comando,

```
varre(100,200,10,20,3);
```

Teríamos no arquivo de saída,

```
v0001000002000001000020000 3
```

Comando tempo

t(quatro caracteres para ΔT)

Então para a linha de comando,

```
t(10);
```

Teríamos no arquivo de saída,

```
t0020
```

Comando calibra

```
c
```

Então para a linha de comando,

```
calibra;
```

Teríamos no arquivo de saída,

```
c
```

Comando home

```
h
```

Então para a linha de comando,

```
home;
```

Teríamos no arquivo de saída,

```
h
```

Comando botao

```
b
```

Então para a linha de comando,

```
botao;
```

Teríamos no arquivo de saída,

```
b
```

Comando irradia

i(quatro caracteres para ΔT)

Então para a linha de comando,

```
Irradia(120);
```

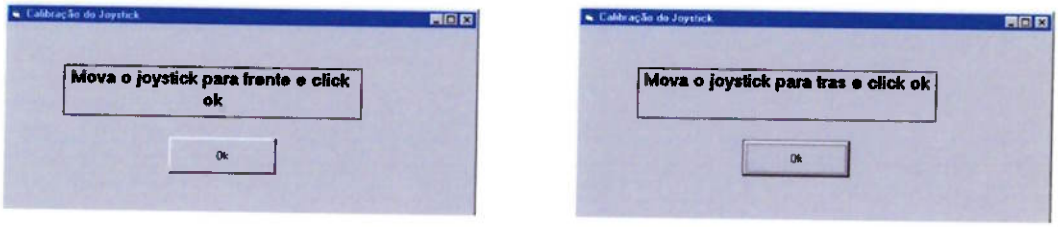
Teríamos no arquivo de saída,

```
h0120
```

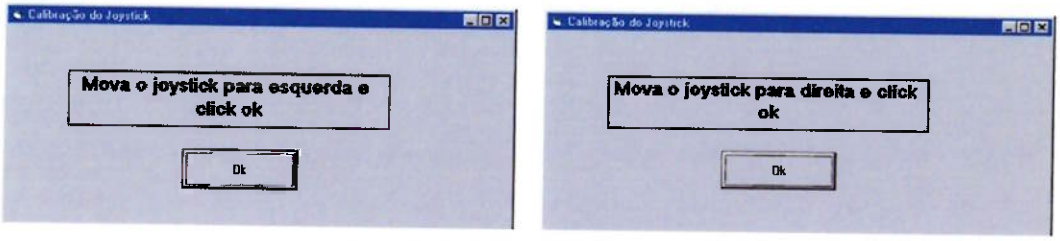
9.4 - Implementação do modo de funcionamento com joystick

A movimentação do goniômetro com o joystick visa dar agilidade de posicionamento. São utilizadas duas entrada analógicas da placa de aquisição que fazem uma leitura de tensão. Através de um conversor A/D incorporado a placa, o software recebe o sinal digitalizado. Para cada posição do joystick tem-se um valor digitalizado.

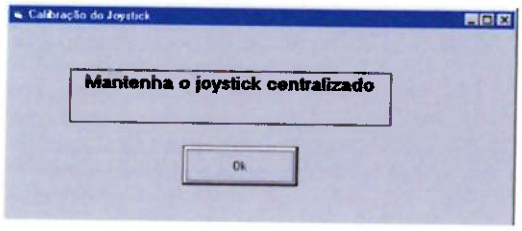
Para evitar que o software seja suscetível a variações de tensões e a alterações no modelo do joystick, uma função de calibração foi implementada de modo que os limites de deslocamento em todas as direções bem como o ponto central sejam identificados e fiquem armazenados. A função de calibração deve ser utilizada sempre que o usuário mudar de joystick.



Exemplo de calibração de um joystick



Exemplo de calibração de um joystick



Exemplo de calibração de um joystick

A movimentação do goniômetro através joystick utiliza a mesmas funções dos outros modos, ou seja, está limitada pelos fins de curso correspondente de cada grau de liberdade.

10.Conclusão

Até o presente momento os objetivos desse projeto foram alcançados. Todas as principais funcionalidades esperadas foram implementadas. O próximo passo é realizar testes do sistema em loco para verificar sua real eficiência. Tal experimento ainda não

foi possível devido a problemas técnicos no acelerador. É muito provável que problemas no sistema apareçam numa primeira fase de testes, mas como todo projeto, esses devem ser corrigidos e fazendo com que essa automação seja realmente efetiva em seu objetivo de aumentar a produtividade de laboratório.

11.Bibliografia

[1] Kenjo,Takashi e Sugawara,Akira . Stepping motors and their microprocessor controls. 2a ed., Oxford University Press, New York, 1994.

[2] Free Software Fundation, Inc. Binson.Info Informações sobre o Bison. 2001

[3] Free Software Fundation, Inc. Flex.Info Informações sobre o Flex. 2001

Apêndice 1-Código Fonte do Programa

```
Dim PosicaoRel(5) As Integer          'vetor que armazena a origem
relativa em relacao a absoluta

Dim PosicaoAbs(5) As Integer          'vetor que armazena a posição
absoluta

Dim PassosCalib(5) As Integer         'vetor que armazena as
posições de calibração

Dim PosCarga(5) As Integer            'vetor que armazena a posição
de carga

Dim Compensacao As Boolean

Public loopss As Integer              'variavel de calibração do
tempo

Public emergencia As Boolean          'variavel de emergencia
utilizada para parar o programa


Public Function PosicaoRelativa(graulib As Integer) As Integer



PosicaoRelativa = PosicaoAbs(graulib) - PosicaoRel(graulib)



End Function



Public Function DefineCarga()


PosCarga(1) = PosicaoAbs(1)

PosCarga(2) = PosicaoAbs(2)

PosCarga(3) = PosicaoAbs(3)

PosCarga(4) = PosicaoAbs(4)
```

```
PosCarga(5) = PosicaoAbs(5)

End Function

Public Function Carga()

Dim Deslocamento As Integer

Do While (graulib <= 5)

    Deslocamento = PosicaoAbs(graulib) - PosCarga(graulib)

        If (Deslocamento >= 0) Then

        PassosExec = MoveMotor(Abs(Deslocamento), graulib, 1,
False)

    Else

        PassosExec = MoveMotor(Abs(Deslocamento), graulib, 1,
True)

    End If

graulib = graulib + 1

Loop
```

```
End Function

Public Function Home()

Dim graulib, PassosExec As Integer
Dim Deslocamento As Integer
graulib = 1

Do While (graulib <= 5)

    If Label12.Caption = ("Coordenada Absoluta") Then

    Deslocamento = PosicaoAbs(graulib)

    Else

    Deslocamento = PosicaoAbs(graulib) - PosicaoRel(graulib)

    End If

        If (Deslocamento >= 0) Then

        PassosExec = MoveMotor(Abs(Deslocamento), graulib, 1,
False)

    Else
```

```
        PassosExec = MoveMotor(Abs(Deslocamento), graulib, 1,
True)

    End If

graulib = graulib + 1

Loop

End Function
Public Function Relativo()

PosicaoRel(1) = PosicaoAbs(1)
PosicaoRel(2) = PosicaoAbs(2)
PosicaoRel(3) = PosicaoAbs(3)
PosicaoRel(4) = PosicaoAbs(4)
PosicaoRel(5) = PosicaoAbs(5)

If (Label12.Caption = "Coordenada Relativa") Then

TextBox6.Text = 0
TextBox7.Text = 0
TextBox8.Text = 0
TextBox9.Text = 0
TextBox10.Text = 0

End If

End Function
```

```
'*****************************************************************
**********************

' Função CalibracaoDoTempo()

'Descrição: esta função realiza a calibraçào da base de tempo do
computador.

'Com ela é possível obter reprodutibilidade de desempenho em
computadores diferentes

'

'*****************************************************************
**********************

Public Function CalibracaoDoTempo()


Dim tempo, tempoa As String          'variável utilizada na
variação do tempo

Dim segundos As Integer              'variável que armazena


segundos = 2

loopss = 0


tempo = Now()                        'a função Now() retorna a hora
do computador

tempo = Right(tempo, 1)              'a variável tempo armazena o
conteudo dos segundos

tempoa = tempo


Do While (segundos > 0)


    tempo = Now()

    tempo = Right(tempo, 1)

    tempoa = tempo

    Do While (tempoa = tempo)

        tempo = Now()

        tempo = Right(tempo, 1)
```

```
        loopss = loopss + 1
    Loop

    segundos = segundos - 1

Loop

End Function

'*****************************************************************
**********************
'Função Delay(microsegundos)
'variáveis de entrada: microsegundos
'Descrição

'*****************************************************************
**********************
Public Function delay(ByVal microsegundos As Integer)

Dim tempo As String
Dim delay1 As Double

'loopss = 3000
delay1 = loopss * 0.001

Do While (microsegundos > 0)

    Do While (delay1 > 0)

        tempo = Now()
        delay1 = delay1 - 1
```

```
Loop


End Function


'*********************************************************************
**********************

'Função Calibração

'Variáveis de entrada:-

'Descrição: Esta função realiza a rotina de calibração do
sistema, deslocando todos os

'graus de liberdade até atingir os fins de curso. Desta forma, é
possível obter o número

'de passo a serem percorridos em cada grau de liberdade

'*********************************************************************
**********************


Public Function Calibracao() As Boolean


Dim PassosExec As Integer          'armazena o número de passo em
cada intervalo

Dim NumPassos As Integer           'armazena o número de passos
executados

Dim graulib As Integer             'armazena o grau de liberdade
que está sendo calibrado


PassosCalib(1) = 0                 'inicia o vetor como as
posicoes de calibracao

PassosCalib(2) = 0

PassosCalib(3) = 0
```

```
Loop

delay1 = loopss * 0.001

microsegundos = microsegundos - 1
```

```
PassosCalib(4) = 0
PassosCalib(5) = 0

graulib = 1

Do While (graulib <= 5)

    PassosExec = 100

    Do While (PassosExec = 100)

        PassosExec = MoveMotor(100, graulib, 10, False)
        TextBox1.Text = "01"
        DoEvents

    Loop

    PassosExec = 100

    Do While (PassosExec = 100)

        PassosExec = MoveMotor(100, graulib, 10, True)
        PassosCalib(graulib) = PassosCalib(graulib) + PassosExec
        TextBox1.Text = "o2"
        DoEvents

    Loop

    TextBox1.Text = PassosCalib1
    graulib = graulib + 1
```

```
Loop

PassosExec = MoveMotor(PassosCalib(1) / 2, 1, 1, False)
PassosExec = MoveMotor(PassosCalib(2) / 2, 2, 1, False)
PassosExec = MoveMotor(PassosCalib(3) / 2, 3, 1, False)
PassosExec = MoveMotor(PassosCalib(4) / 2, 4, 1, False)
PassosExec = MoveMotor(PassosCalib(5) / 2, 5, 1, False)

PosicaoAbs(1) = 0
PosicaoAbs(2) = 0
PosicaoAbs(3) = 0
PosicaoAbs(4) = 0
PosicaoAbs(5) = 0

TextBox6.Text = PosicaoAbs(1)
TextBox7.Text = PosicaoAbs(2)
TextBox8.Text = PosicaoAbs(3)
TextBox9.Text = PosicaoAbs(4)
TextBox10.Text = PosicaoAbs(5)

TextBox1.Text = PassosCalib(1)
TextBox2.Text = PassosCalib(2)
TextBox3.Text = PassosCalib(3)
TextBox4.Text = PassosCalib(4)
TextBox5.Text = PassosCalib(5)

Calibracao = True

End Function
```

```
'*****************************************************************
**********************

'Função FimDeCurso

'Variáveis de entrada:

'Graulib = representa qual grau de liberdade sera monitorado.

'Sentido = representa o sentido de movimentação e
consequentemente o switch correspondente.

'Variáveis de saída: retorna se o switch foi atingido ou não.

'Descrição : a medida que o motor se desloca, a função verifica
se o fim de curso não é

'atingido.


'*****************************************************************
**********************


Public Function FimDeCurso(ByVal graulib As Integer, ByVal
Sentido As Boolean) As Boolean



Dim fdcurso1, fdcurso2 As Boolean 'variáveis que armazenam os
estados dos switchs


fdcurso1 = False                    'inicialização das variáveis

fdcurso2 = False


Select Case graulib


      Case 1:


      If (Sentido = False) Then


       DigitalBitInA.ControlIn = False
```

```
DigitalBitInA.ControlIn = True
DigitalBitInA.BitNumber = 0
fdcurso1 = DigitalBitInA.Value

Else

DigitalBitInA.ControlIn = False
DigitalBitInA.ControlIn = True
DigitalBitInA.BitNumber = 5
fdcurso2 = DigitalBitInA.Value

End If

Case 2:

If (Sentido = False) Then

DigitalBitInA.ControlIn = False
DigitalBitInA.ControlIn = True
DigitalBitInA.BitNumber = 1
fdcurso1 = DigitalBitInA.Value

Else

DigitalBitInA.ControlIn = False
DigitalBitInA.ControlIn = True
DigitalBitInA.BitNumber = 6
fdcurso2 = DigitalBitInA.Value

End If
```

```
Case 3:

If (Sentido = False) Then

DigitalBitInA.ControlIn = False
DigitalBitInA.ControlIn = True
DigitalBitInA.BitNumber = 2
fdcurso1 = DigitalBitInA.Value

Else

DigitalBitInA.ControlIn = False
DigitalBitInA.ControlIn = True
DigitalBitInA.BitNumber = 7
fdcurso2 = DigitalBitInA.Value

End If

Case 4:

If (Sentido = False) Then

DigitalBitInA.ControlIn = False
DigitalBitInA.ControlIn = True
DigitalBitInA.BitNumber = 3
fdcurso1 = DigitalBitInA.Value

Else
```

```
        DigitalBitInA.ControlIn = False
        DigitalBitInA.ControlIn = True
        DigitalBitInA.BitNumber = 8
        fdcurso2 = DigitalBitInA.Value

        End If

        Case 5:

        If (Sentido = False) Then

        DigitalBitInA.ControlIn = False
        DigitalBitInA.ControlIn = True
        DigitalBitInA.BitNumber = 4
        fdcurso1 = DigitalBitInA.Value

        Else

        DigitalBitInA.ControlIn = False
        DigitalBitInA.ControlIn = True
        DigitalBitInA.BitNumber = 9
        fdcurso2 = DigitalBitInA.Value

        End If

End Select

FimDeCurso = fdcurso1 Or fdcurso2 'retorna o estado dos swtchi
```

```
End Function


'*****************************************************************
**********************

'Função MoveMotor

'Variáveis de entrada:

'Numpassos: armazena o número de passos a serem movimentados

'Graulib: armazena o grau de liberdade a ser movido

'Velocidade: armazena a velocidade de movimentação

'Sentido: armazena o sentido de movimentação do motor

'Variável de saída: retorna o número de passos que foram
executados

'Descrição: esta função realiza a movimentação dos motores
enviando pulsos de acordo

'com as variáveis: Sentido, Velocidade e Graulib.


'*****************************************************************
**********************


Public Function MoveMotor(ByVal NumPassos As Integer, ByVal
graulib As Integer, ByVal Velocidade As Integer, ByVal Sentido
As Boolean) As Integer


Dim velocidade2 As Integer          'armazena o valor incial da
variável

Dim NumPassoInicial As Integer      '

Dim i As Boolean

Dim switch As Boolean


NumPassos = NumPassos * 2

NumPassoInicial = NumPassos

velocidade2 = Velocidade

i = True
```

```
switch = False


Do While ((NumPassos > 0) And (emergencia = False) And (switch =
False))
        Do While (velocidade2 > 0)

        velocidade2 = velocidade2 - 1

        Loop

velociade2 = Velocidade
NumericText4.InputText = NumPassos / 2
DoEvents
delay (Velocidade)

NumPassos = NumPassos - 1


If (CommandButton7.Caption = "Coordenada Absoluta") Then

TextBox6.Text = PosicaoAbs(1) - PosicaoRel(1)
TextBox7.Text = PosicaoAbs(2) - PosicaoRel(2)
TextBox8.Text = PosicaoAbs(3) - PosicaoRel(3)
TextBox9.Text = PosicaoAbs(4) - PosicaoRel(4)
TextBox10.Text = PosicaoAbs(5) - PosicaoRel(5)

End If

If (CommandButton7.Caption = "Coordenada Relativa") Then
```

```
TextBox6.Text = PosicaoAbs(1)
TextBox7.Text = PosicaoAbs(2)
TextBox8.Text = PosicaoAbs(3)
TextBox9.Text = PosicaoAbs(4)
TextBox10.Text = PosicaoAbs(5)

End If

If (i = True) Then

    i = False


Else
    If (i = False) Then

    i = True
    switch = FimDeCurso(graulib, Sentido)
    End If

End If

If (Sentido = True And i = False) Then          'atualizacao da
variavel global de posicao absoluta

    PosicaoAbs(graulib) = PosicaoAbs(graulib) + 1

End If
```

```
If (Sentido = False And i = False) Then

    PosicaoAbs(graulib) = PosicaoAbs(graulib) - 1

End If

Select Case graulib

Case 1: DigitalBitOut2.BitNumber = 10
        DigitalBitOut1.BitNumber = 11

        If (Sentido = True) Then

            DigitalBitOut2.BitValue = True

        Else

            DigitalBitOut2.BitValue = False

        End If

           DigitalBitOut1.BitValue = i

Case 2: DigitalBitOut2.BitNumber = 12
        DigitalBitOut1.BitNumber = 13

        If (Sentido = True) Then
```

```
            DigitalBitOut2.BitValue = True

        Else

            DigitalBitOut2.BitValue = False

        End If

          DigitalBitOut1.BitValue = i


Case 3: DigitalBitOut2.BitNumber = 14
        DigitalBitOut1.BitNumber = 15


        If (Sentido = True) Then

            DigitalBitOut2.BitValue = True

        Else

            DigitalBitOut2.BitValue = False

        End If

        DigitalBitOut1.BitValue = i
```

```
Case 4: DigitalBitOut2.BitNumber = 16  'bit do sentido
        DigitalBitOut1.BitNumber = 17  'bit do clock

        If (Sentido = True) Then

            DigitalBitOut2.BitValue = True

        Else

            DigitalBitOut2.BitValue = False

        End If

        DigitalBitOut1.BitValue = i


Case 5: DigitalBitOut2.BitNumber = 18
        DigitalBitOut1.BitNumber = 19

        If (Sentido = True) Then

            DigitalBitOut2.BitValue = True

        Else

            DigitalBitOut2.BitValue = False

        End If

       DigitalBitOut1.BitValue = i
```

```
End Select

Loop

MoveMotor = (NumPassoInicial - NumPassos) / 2

End Function

Public Function ValidaEntrada(Dado As String, Caixa As Integer)
As Boolean

Select Case Caixa

    Case 1:
    Case 2:
    Case 3:
    Case 13: ValidaEntrada = Dado Like String$(Len(Dado), "#")

End Select

End Function


Private Sub ComboBox4_Click()

Select Case ComboBox4.Text

Case "Manual"
```

```
CommandButton1.Enabled = True
CommandButton3.Enabled = True
CommandButton4.Enabled = True
CommandButton5.Enabled = True
CommandButton8.Enabled = False
CommandButton9.Enabled = True

Case "Programável"

CommandButton1.Enabled = False
CommandButton3.Enabled = False
CommandButton4.Enabled = False
CommandButton5.Enabled = False
CommandButton8.Enabled = True
CommandButton9.Enabled = False

Case "Joystick"

CommandButton1.Enabled = False
CommandButton3.Enabled = False
CommandButton4.Enabled = False
CommandButton5.Enabled = False
CommandButton8.Enabled = False
CommandButton9.Enabled = True

Do While (ComboBox4.Text = "Joystick")

Joystick
DoEvents
```

```
Loop




End Select



End Sub

Private Sub CommandButton1_Click()

Dim graulib As Integer          'armazena o grau de liberdade que
ira mover
Dim Passos As String
Dim NumPassos As Integer      'armazena o número de passos que o
motor ira andar
Dim Sentido As Boolean          'armazena o sentido de
movimentaçào do motor
Dim Velocidade  As Integer      'armazena a velocidade de
movimentação do motor
Dim Mensagem As Integer
Dim Valida As Boolean
Dim Entrada As Boolean
Dim Analogico As Variant
Dim Compensa As Double
Dim CompensaInt As Integer
Dim Angulo As Double
Dim PosAbs(5) As Long
Dim PosRel(5) As Long
```

```
Entrada = True

Passos = TextBox13.Text

Valida = ValidaEntrada(Passos, 13)


If (Valida = False) Then


    Mensagem = MsgBox("Entrada não válida no campo NumPassos",
48 + 0, "Erro na entrada de dados")

Else


    NumPassos = Passos


End If


Select Case ComboBox1.Text      'define o grau de liberdade


    Case "X":      graulib = 1

    Case "Y":      graulib = 2

    Case "Z":      graulib = 3

    Case "Theta": graulib = 4

    Case "Phi":    graulib = 5

    Case Else:     Mensagem = MsgBox("Entrada não válida no campo
Graulib", 48 + 0, "Erro na entrada de dados")

                   Entrada = False

End Select


Select Case ComboBox2.Text      'define a velocidade de
movimentação do motor


    Case "1":  Velocidade = 200
```

```
    Case "2":  Velocidade = 100

    Case "3":  Velocidade = 50

    Case "4":  Velocidade = 5

    Case "5":  Velocidade = 1

    Case Else: Mensagem = MsgBox("Entrada não válida no campo
velocidade", 48 + 0, "Erro na entrada de dados")

               Entrada = False


End Select


Select Case ComboBox3.Text      'define o sentido de movimentação
do motor


    Case "Positivo": Sentido = True

    Case "Negativo": Sentido = False

    Case Else:       Entrada = False

                     Mensagem = MsgBox("Entrada não válida no
campo Sentido", 48 + 0, "Erro na entrada de dados")

End Select


AnalogIn1.ControlIn = False

AnalogIn1.ControlIn = True

Analogico = AnalogIn1.Value

TextBox14.Text = Analogico


If (Entrada = True) Then


NumericText7.InputText = MoveMotor(NumPassos, graulib,
Velocidade, Sentido)


End If
```

```
If (graulib = 4 And Compensacao = True) Then


PosAbs(4) = CLng(PosicaoAbs(4))

PosRel(4) = CLng(PosicaoRel(4))

PosAbs(1) = CLng(PosicaoAbs(1))

PosRel(1) = CLng(PosicaoRel(1))


Angulo = ((CDbl(PosicaoAbs(4)) - CDbl(PosicaoRel(4))) * 0.02 *
3.1415) / 180

Compensa = Sin(Angulo) * (CDbl(PosicaoAbs(1)) -
CDbl(PosicaoRel(1)))

CompensaInt = CInt(Compensa) - (PosicaoAbs(3) - PosicaoRel(3))


If (CompensaInt > 0) Then


Sentido = True


Else


Sentido = False


End If


NumericText7.InputText = MoveMotor(Abs(CompensaInt), 3,
Velocidade, Sentido)


End If


End Sub


Public Function Joystick()
```

```
Dim Sentido As Boolean
Dim PassosExec As Integer
Dim Analog As Integer
Dim Velocidade As Integer
Dim Delta As Integer

AnalogIn1.ControlIn = False
AnalogIn1.ControlIn = True
Analog = AnalogIn1.Value
TextBox11.Text = Analog

DoEvents

If (Analog > Form4.Meio1) Then     'define qual será o sentido de
rotação do motor

    Sentido = True

Else

    Sentido = False

End If

Delta = Abs(Form4.Analogico1 - Form4.Analogico2) / 10
TextBox14.Text = Delta
Select Case Analog

Case (Form4.Meio1 - Delta) To (Form4.Meio1 + Delta)
```

```
'Joystick no centro. sem movimento

Case (Form4.Meio1 + Delta) To (Form4.Meio1 + Delta * 2)

    PassosExec = MoveMotor(1, 1, 100, Sentido)

Case (Form4.Meio1 - Delta * 2) To (Form4.Meio1 - Delta)

    PassosExec = MoveMotor(1, 1, 100, Sentido)

Case (Form4.Meio1 + Delta * 2) To (Form4.Meio1 + Delta * 3)

    PassosExec = MoveMotor(1, 1, 10, Sentido)

Case (Form4.Meio1 - Delta * 3) To (Form4.Meio1 - Delta * 2)

    PassosExec = MoveMotor(1, 1, 10, Sentido)

Case (Form4.Meio1 - Delta * 8) To (Form4.Meio1 - Delta * 3)

    PassosExec = MoveMotor(1, 1, 1, Sentido)

Case (Form4.Meio1 + Delta * 3) To (Form4.Meio1 + Delta * 8)

    PassosExec = MoveMotor(1, 1, 1, Sentido)

End Select
```

```
End Function



Private Sub CommandButton10_Click()

DefineCarga


End Sub

Private Sub CommandButton11_Click()

Carga

End Sub

Private Sub CommandButton12_Click()

Form5.Show

End Sub

Private Sub CommandButton13_Click()

 Dim Posicao As Integer
 Posicao = 0
```

```
 Form5.MovePosicao (Posicao)

End Sub

Private Sub CommandButton14_Click()
Dim Posicao As Integer
 Posicao = 5
 Form5.MovePosicao (Posicao)

End Sub

Private Sub CommandButton15_Click()
Dim Posicao As Integer
 Posicao = 10
 Form5.MovePosicao (Posicao)

End Sub

Private Sub CommandButton16_Click()
Dim Posicao As Integer
 Posicao = 15
 Form5.MovePosicao (Posicao)

End Sub

Private Sub CommandButton17_Click()
Dim Posicao As Integer
 Posicao = 20
 Form5.MovePosicao (Posicao)
```

```
End Sub

Private Sub CommandButton18_Click()
Dim Posicao As Integer
 Posicao = 25
 Form5.MovePosicao (Posicao)

End Sub

Private Sub CommandButton19_Click()

If (Compensacao = True) Then

Compensacao = False
CommandButton19.Caption = "Ativar Compensação"

Else

Compensacao = True
CommandButton19.Caption = "Desativar Compensação"

End If

End Sub

Private Sub CommandButton2_Click()

End

End Sub
```

```
Private Sub CommandButton3_Click()

Form2.Show


End Sub

Private Sub CommandButton4_Click()

Dim resultado As Boolean

resultado = Calibracao()


End Sub


Private Sub CommandButton5_Click()

Home

End Sub

Private Sub CommandButton6_Click()

Relativo


End Sub
```

```
Private Sub CommandButton7_Click()

If (CommandButton7.Caption = "Coordenada Relativa") Then

TextBox6.Text = PosicaoAbs(1) - PosicaoRel(1)
TextBox7.Text = PosicaoAbs(2) - PosicaoRel(2)
TextBox8.Text = PosicaoAbs(3) - PosicaoRel(3)
TextBox9.Text = PosicaoAbs(4) - PosicaoRel(4)
TextBox10.Text = PosicaoAbs(5) - PosicaoRel(5)
CommandButton7.Caption = "Coordenada Absoluta"
Label12.Caption = "Coordenada Relativa"

Else

TextBox6.Text = PosicaoAbs(1)
TextBox7.Text = PosicaoAbs(2)
TextBox8.Text = PosicaoAbs(3)
TextBox9.Text = PosicaoAbs(4)
TextBox10.Text = PosicaoAbs(5)
CommandButton7.Caption = "Coordenada Relativa"
Label12.Caption = "Coordenada Absoluta"

End If




End Sub
```

```
Private Sub CommandButton8_Click()

Form3.Show

End Sub

Private Sub CommandButton9_Click()

Form4.Show

End Sub

Private Sub Form_Load()

'##################################################################
##########LoadBegin
'
' SoftWIRE
' IMPORTANT - The following lines of Load code were added by
SoftWIRE.
'             Do not remove them or the program will not run
correctly.
Load Form1

Wire1.Initialize
'
```

```
'#################################################################
############LoadEnd

CalibracaoDoTempo

Label12.Caption = "Coordenadas Absolutas"

Compensacao = True


emergencia = False

DigitalBitInA.Port = 10

DigitalBitOut1.Port = 10

DigitalBitOut2.Port = 10


ComboBox1.AddItem "X"

ComboBox1.AddItem "Y"

ComboBox1.AddItem "Z"

ComboBox1.AddItem "Theta"

ComboBox1.AddItem "Phi"


ComboBox2.AddItem "1"

ComboBox2.AddItem "2"

ComboBox2.AddItem "3"

ComboBox2.AddItem "4"

ComboBox2.AddItem "5"


ComboBox3.AddItem "Positivo"

ComboBox3.AddItem "Negativo"


ComboBox4.AddItem "Manual"

ComboBox4.AddItem "Programável"

ComboBox4.AddItem "Joystick"
```

```
End Sub


Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)


'##############################################################
##########
'
' SoftWIRE
' IMPORTANT - The following lines of QueryUnload code were added
by SoftWIRE.
'             Do not remove them or the program will not run
correctly.
Dim Form As Form
Dim Count As Integer
On Error Resume Next


If UnloadMode = vbFormMDIForm Then
    Exit Sub
End If


Count = 0
For Each Form In Forms
    If Form.Visible Then
        Count = Count + 1
    End If
Next Form


If Count > 1 Then
```

```
    Cancel = True
    Me.Hide
Else
    For Each Form In Forms
        Unload Form
    Next Form
End If
'
'#################################################################
############

End Sub
```

```
Public Function TempoDeRadiacao(delay As Integer)

Dim tempo, tempoa As String
Dim segundos As Integer

segundos = delay

tempo = Now()
tempo = Right(tempo, 1)
tempoa = tempo
Do While (segundos > 0)
     tempo = Now()
     tempo = Right(tempo, 1)
     tempoa = tempo
      Do While (tempoa = tempo)
       tempo = Now()
       tempo = Right(tempo, 1)
```

```
        Loop
      segundos = segundos - 1

  Loop

  End Function


  Public Function Varredura(ByVal valorX As Integer, ByVal valorY
  As Integer, ByVal deltaX As Integer, ByVal deltaY As Integer,
  ByVal delay As Integer) As Boolean

  Dim sentidoX, sentidoY As Boolean
  Dim deltaXR, deltaYR As Integer
  Dim passoexe As Integer


  valorxr = Abs(valorX)
  valoryr = Abs(valorY)

  If (valorX > 0) Then
      sentidoX = True
      Else
          sentidoX = False
  End If

  If (valorY > 0) Then
      sentidoY = True
      Else
          sentidoY = False
  End If
```

```
Do While (valoryr > 0)

    Do While (valorxr > 0)

    TempoDeRadiacao (delay)
    passosexe = Form1.MoveMotor(deltaX, 1, 10, sentidoX)

    NumericText4.InputText = valorxr

    valorxr = valorxr - deltaX

    DoEvents

    Loop

    NumericText4.InputText = 1000
DoEvents
valorxr = valorX
TempoDeRadiacao (delay)
passosexe = Form1.MoveMotor(deltaY, 2, 10, sentidoY)
valoryr = valoryr - deltaY
Select Case sentidoX
Case True:  sentidoX = False
Case False: sentidoX = True

End Select
```

```
Loop

Varredura = False

End Function


Private Sub CommandButton1_Click()


Dim valorX, valorY, deltaX, deltaY, tempo As Integer

Dim FimDaVarredura As Boolean









valorX = NumericText11.Text

valorY = NumericText2.Text

deltaX = NumericText3.Text

deltaY = NumericText1.Text

tempo = NumericText5.Text

FimDaVarredura = Varredura(valorX, valorY, deltaX, deltaY,
tempo)









End Sub


Private Sub CommandButton2_Click()
```

```
Form2.Hide



End Sub


Private Sub Form_Load()


'##################################################################
##########LoadBegin
'
' SoftWIRE
' IMPORTANT - The following lines of Load code were added by
SoftWIRE.
'             Do not remove them or the program will not run
correctly.


Load Form2


'
'##################################################################
############LoadEnd



End Sub
```

```
Public botao As Boolean

Private Sub CommandButton1_Click()
Dim Entrada As String
Dim Funcaoid As String
Dim pmta, pmtb, pmtc As Integer
Dim paramet As Variant
Dim fim As Boolean
Dim arquivo As String
Dim PassosExec
Dim Fimdavarre As Boolean

CommandButton1.Enabled = False
paramet = Array(100)
arquivo = CommonDialog1.FileName

Open arquivo For Input As #1

Input #1, Entrada
```

```
Funcaoid = Left(Entrada, 1)


Do While (fim = False)


    Select Case Funcaoid


    Case "m"


    paramet1 = Mid(Entrada, 2, 5) 'numero de passos a serem
realizados
    paramet2 = Mid(Entrada, 7, 1) 'grau de liberdade
    paramet3 = Mid(Entrada, 8, 1) 'velocidade
    paramet4 = Mid(Entrada, 9, 1) 'sentido de movimento
    TextBox2.Text = paramet1
    TextBox3.Text = paramet2
    TextBox4.Text = paramet3
    TextBox5.Text = paramet4
    TextBox6.Text = ""
    TextBox1.Text = "movendo"
    Label2.Caption = "Número de Passos"
    Label3.Caption = "Grau de Liberdade"
    Label4.Caption = "Velocidade"
    Label5.Caption = "Sentido"
    Label6.Caption = ""
    DoEvents
    PassosExec = Form1.MoveMotor(paramet1, paramet2, paramet3,
paramet4)



   Case "v"
```

```
    paramet1 = Mid(Entrada, 2, 5)  'distância em x
    paramet2 = Mid(Entrada, 7, 5)  'distância em y
    paramet3 = Mid(Entrada, 12, 5) ' delta x
    paramet4 = Mid(Entrada, 17, 5) ' delta y
    paramet5 = Mid(Entrada, 22, 4) 'tempo de incidência em
segundos
    TextBox2.Text = paramet1
    TextBox3.Text = paramet2
    TextBox4.Text = paramet3
    TextBox5.Text = paramet4
    TextBox6.Text = paramet5
    TextBox1.Text = "varrendo"
    Label2.Caption = "Delta X"
    Label3.Caption = "Delta Y"
    Label4.Caption = "Incremento X"
    Label5.Caption = "Incremento Y"
    Label6.Caption = "Tempo"
    DoEvents
    Fimdavarre = Form2.Varredura(prametl, paramet2, paramet3,
paramet4, paramet5)


    Case "t"
    paramet1 = Mid(Entrada, 2, 4) 'tempo de atrazo em segundos
    TextBox2.Text = paramet1
    TextBox3.Text = ""
    TextBox4.Text = ""
    TextBox5.Text = ""
    TextBox6.Text = ""
    TextBox1.Text = "tempo"
    Label2.Caption = "Tempo(s)"
    Label3.Caption = ""
```

```
Label4.Caption = ""
Label5.Caption = ""
Label6.Caption = ""
DoEvents
Form2.TempoDeRadiacao (paramet1)

Case "h"

TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""
TextBox5.Text = ""
TextBox6.Text = ""
TextBox1.Text = "Home"
Label2.Caption = ""
Label3.Caption = ""
Label4.Caption = ""
Label5.Caption = ""
Label6.Caption = ""
DoEvents
Form1.Home

Case "c"

TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""
TextBox5.Text = ""
TextBox6.Text = ""
TextBox1.Text = "Calibração"
```

```
Label2.Caption = ""
Label3.Caption = ""
Label4.Caption = ""
Label5.Caption = ""
Label6.Caption = ""
DoEvents
Form1.Calibracao

Case "b"

botao = False
TextBox2.Text = ""
TextBox3.Text = ""
TextBox4.Text = ""
TextBox5.Text = ""
TextBox6.Text = ""
TextBox1.Text = "Botão"
Label2.Caption = ""
Label3.Caption = ""
Label4.Caption = ""
Label5.Caption = ""
Label6.Caption = ""
DoEvents
Do While (botao = False)
DoEvents
Loop
Form1.Home

Case "i"
```

```
        paramet1 = Mid(Entrada, 2, 4) 'tempo de irradiação
        TextBox2.Text = paramet1
        TextBox3.Text = ""
        TextBox4.Text = ""
        TextBox5.Text = ""
        TextBox6.Text = ""
        TextBox1.Text = "Irradia"
        Label2.Caption = ""
        Label3.Caption = ""
        Label4.Caption = ""
        Label5.Caption = ""
        Label6.Caption = ""
        DoEvents

        End Select

    fim = EOF(1)


    If (fim = False) Then

    Input #1, Entrada

    Funcaoid = Left(Entrada, 1)

    End If

    Loop

    Close #1
```

```
CommandButton1.Enabled = True

End Sub

Private Sub CommandButton2_Click()

CommonDialog1.ShowOpen

End Sub

Private Sub CommandButton3_Click()

CommonDialog1.ShowOpen

CBExecute1.ControlIn = False
CBExecute1.ControlIn = True
CBExecute1.CommandLine = "c:\windows\notepad.exe"




End Sub

Private Sub CommandButton5_Click()

Form3.Hide

End Sub
```

```
Private Sub CommandButton6_Click()


botao = True



End Sub


Private Sub Form_Load()


'##################################################################
##########LoadBegin
'
' SoftWIRE
' IMPORTANT - The following lines of Load code were added by
SoftWIRE.
'             Do not remove them or the program will not run
correctly.


Load Form3


'


'##################################################################
############LoadEnd




End Sub
```

```
Public Limite As Integer
Public Analogico1 As Integer
Public Analogico2 As Integer
Public Analogico3 As Integer
Public Analogico4 As Integer
Public Meio1 As Integer
Public Meio2 As Integer


Public Function CalibracaoJoystick(Limite As Integer)


Select Case Limite

Case 1:    Form1.AnalogIn1.ControlIn = False
           Form1.AnalogIn1.ControlIn = True
           Analogico1 = Form1.AnalogIn1.Value
```

```
            Label1.Caption = "Mova o joystick para tras e click
ok"

            TextBox1.Text = Analogico1


Case 2:     Form1.AnalogIn1.ControlIn = False

            Form1.AnalogIn1.ControlIn = True

            Analogico2 = Form1.AnalogIn1.Value

            Label1.Caption = "Mova o joystick para esquerda e
click ok"

            TextBox1.Text = Analogico2


Case 3:     Form1.AnalogIn1.ControlIn = False

            Form1.AnalogIn1.ControlIn = True

            Analogico3 = Form1.AnalogIn1.Value

            Label1.Caption = "Mova o joystick para direita e click
ok"

            TextBox1.Text = Analogico3


Case 4:     Form1.AnalogIn1.ControlIn = False

            Form1.AnalogIn1.ControlIn = True

            Analogico4 = Form1.AnalogIn1.Value

            Label1.Caption = "Mantenha o joystick centralizado"

            TextBox1.Text = Analogico4


Case 5:     Form1.AnalogIn1.ControlIn = False

            Form1.AnalogIn1.ControlIn = True

            Meio1 = Form1.AnalogIn1.Value

            Meio2 = Form1.AnalogIn1.Value

            Label1.Caption = "Calibração realizada com sucesso"

            TextBox1.Text = Meio1
```

```
End Select

End Function

Private Sub CommandButton1_Click()


CalibracaoJoystick (Limite)
Limite = Limite + 1

If (Limite = 6) Then

CommandButton1.Caption = "Sair"

End If

If (Limite > 6) Then

Limite = 1
Label1.Caption = "Mova o joystick para frente e click ok"
CommandButton1.Caption = "Ok"

Form4.Hide

End If

End Sub

Private Sub Form_Load()
```

```
'##################################################################
##########LoadBegin
'
' SoftWIRE
' IMPORTANT - The following lines of Load code were added by
SoftWIRE.
'             Do not remove them or the program will not run
correctly.

Load Form4
Limite = 1
'
'##################################################################
############LoadEnd

Label1.Caption = "Mova o joystick para frente e click ok"

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)

'##################################################################
##########
'
' SoftWIRE
' IMPORTANT - The following lines of QueryUnload code were added
by SoftWIRE.
'             Do not remove them or the program will not run
correctly.
Dim Form As Form
Dim Count As Integer
On Error Resume Next
```

```
If UnloadMode = vbFormMDIForm Then
    Exit Sub
End If

Count = 0
For Each Form In Forms
    If Form.Visible Then
        Count = Count + 1
    End If
Next Form

If Count > 1 Then
    Cancel = True
    Me.Hide
Else
    For Each Form In Forms
        Unload Form
    Next Form
End If
'
'#################################################################
############


End Sub

Private Sub Label1_Click()

End Sub
```

```
Dim PosUser(50) As Integer

Public Function DefinePosicao()

Dim i As Integer

Select Case ComboBox1.Text

Case "Posição 1"
i = 0
Case "Posição 2"
i = 5
Case "Posição 3"
i = 10
Case "Posição 4"
i = 15
```

```
Case "Posição 5"
i = 20
Case "Posição 6"
i = 25

End Select
PosUser(1 + i) = TextBox1.Text
PosUser(2 + i) = TextBox2.Text
PosUser(3 + i) = TextBox3.Text
PosUser(4 + i) = TextBox4.Text
PosUser(5 + i) = TextBox5.Text

End Function

Public Function MovePosicao(Posicao As Integer)

Dim Deslocamento As Integer
Dim graulib As Integer
graulib = 1

Do While (graulib <= 5)


    Deslocamento = Form1.PosicaoRelativa(graulib) -
PosUser(graulib + Posicao)


        If (Deslocamento >= 0) Then
```

```
        PassosExec = Form1.MoveMotor(Abs(Deslocamento), graulib,
1, False)


    Else


        PassosExec = Form1.MoveMotor(Abs(Deslocamento), graulib,
1, True)


    End If


graulib = graulib + 1


Loop



End Function
Private Sub ComboBox1_Click()


Dim i As Integer


Select Case ComboBox1.Text


Case "Posição 1"
i = 0
Case "Posição 2"
i = 5
Case "Posição 3"
i = 10
Case "Posição 4"
i = 15
Case "Posição 5"
```

```
i = 20
Case "Posição 6"
i = 25

End Select

TextBox1.Text = PosUser(1 + i)
TextBox2.Text = PosUser(2 + i)
TextBox3.Text = PosUser(3 + i)
TextBox4.Text = PosUser(4 + i)
TextBox5.Text = PosUser(5 + i)
DoEvents

End Sub


Private Sub CommandButton1_Click()

DefinePosicao

End Sub

Private Sub CommandButton2_Click()

Form5.Hide

End Sub

Private Sub Form_Load()
```

```
'##########################################################################
##########LoadBegin
'
' SoftWIRE
' IMPORTANT - The following lines of Load code were added by
SoftWIRE.
'             Do not remove them or the program will not run
correctly.

Load Form5

'##########################################################################
############LoadEnd

ComboBox1.AddItem "Posição 1"
ComboBox1.AddItem "Posição 2"
ComboBox1.AddItem "Posição 3"
ComboBox1.AddItem "Posição 4"
ComboBox1.AddItem "Posição 5"
ComboBox1.AddItem "Posição 6"

End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)

'##########################################################################
##########
'
' SoftWIRE
' IMPORTANT - The following lines of QueryUnload code were added
by SoftWIRE.
```

```
'             Do not remove them or the program will not run
correctly.

Dim Form As Form

Dim Count As Integer

On Error Resume Next


If UnloadMode = vbFormMDIForm Then

    Exit Sub

End If


Count = 0

For Each Form In Forms

    If Form.Visible Then

        Count = Count + 1

    End If

Next Form


If Count > 1 Then

    Cancel = True

    Me.Hide

Else

    For Each Form In Forms

        Unload Form

    Next Form

End If

'

'###############################################################
############


End Sub
```

Apêndice 2- Definição dos Padrões Léxicos Reconhecidos pela Linguagem Desenvolvida

```
%option noyywrap

DIGITO [0-9]    /*define que a expressão DIGITO é qualquer*/

                 /* caracter de 0 a 9*/
%%


/*define o padrão move e devolve seu token correspondente*/

"move" {

return MOVE;

}

/*define o padrão varre e devolve seu token correspondente*/

"varre" {

return VARRE;

}

/*define o padrão tempo e devolve seu token correspondente*/

"tempo" {

return TEMPO;


}

/*define o calibra move e devolve seu token correspondente*/

"calibra" {

return CALIBRA;


}


/*define o padrão irradia e devolve seu token correspondente*/
```

```
"irradia" {
return IRRADIA;

}
/*define o padrão botao e devolve seu token correspondente*/
"botao" {
return BOTAO;

}
/*define o padrão home e devolve seu token correspondente*/
"home" {
return HOME;
}
/*define o padrão ( e devolve seu token correspondente*/

"(" {
return ABREP;
}
/*define o padrão ) e devolve seu token correspondente*/
")" {
return FECHAP;
}




/*define o padrão + e devolve seu token correspondente*/

"+" {
return MAIS;
```

```
}

/*define o padrão - e devolve seu token correspondente*/

"-" {

return MENOS;

}

/*define o padrão * e devolve seu token correspondente*/


"*" {

return MULT;

}

/*define o padrão / e devolve seu token correspondente*/

"/" {

return DIVIDE;

}

/*a expressão abaixo reconhece um número do tipo ponto
/*flutuante*/


{DIGITO}+"."{DIGITO}*([eE][-+]?{DIGITO}+)? {
yylval.pontof=atof(yytext);
return NUM;
}

/*a expressão abaixo reconhece um número do tipo inteiro*/


{DIGITO}+ {
yylval.inteiro=atoi(yytext);
return INT;
}


/*define o padrão ; e devolve seu token correspondente*/


";" {
return FIM;

}
```

```
/*define o padrão , e devolve seu token correspondente*/


"," {
return VR;

}

/*a expressão abaixo elimina os comentários feitos pelo
/*usuário*/

"/*"[^\n]*"*/"

/*a expressão abaixo elimina os espaços em branco e as*/
/*tabulações feitas pelo usuário*/

[ \t]+

/*conta o numero de linhas do programa*/

[\n]  {locerro++;}


/*caso nenhum dos caracteres acima seja reconhecido, uma*/
/*mensagem de erro é enviada*/

. {printf("Caracter nao
reconhecido:%s,linha:%d\n",yytext,locerro);
   fprintf(erros,"Caracter nao
reconhecido:%s,linha:%d\n",yytext,locerro);}


/*ao final do arquivo, o valor 0 é retornado*/

<<EOF>> return 0;

%%
```

Apêndice 3- Definição da Gramática Reconhecidos pela Linguagem Desenvolvida

```
%{
/*
Escola Polite'cnica da USP
Projeto Mecanico II
Daniel Olioni Andersson   NUSP 2368345
Rodrigo de Deus Reinaldo  NUSP 2368992
*/

#include <math.h>
```

```
#include <ctype.h>
#include <string.h>
#include <stdio.h>
extern void yyerror(char*);
int yylex(void);
int paramet[100];
int conta;
int locerro;
FILE *fp;
FILE *erros;

%}

%union{
int    inteiro;
float  pontof;
char   str[1000];
}

%token <inteiro> INT
%token <pontof>  NUM
%token <inteiro> MOVE
%token <inteiro> VARRE
%token <inteiro> TEMPO
%token <inteiro> ABREP
%token <inteiro> FECHAP
%token <inteiro> VR
%token <inteiro> FIM
%token <inteiro> CALIBRA
%token <inteiro> HOME
%token <inteiro> IRRADIA
%token <inteiro> BOTAO
%type  <inteiro>  parametro
%type  <inteiro>  exp
/* a expressão left define que esses tokens têm precedência a
esquerda*/
%left MENOS MAIS
%left MULT  DIVIDE
%left NEG /*operador una'rio*/
%%


codigo:    /*nada*/
        | codigo  linha
        | error {yyerrok;}
;

linha : '\n'
      | comando
      | error
;
```

```
/*para cada padrão gramatical reconhecido de um comando, existe
uma ação que no caso é escrever no arquivo de saida esse comando
no formato reconhecível pelo programa principal*/


comando: MOVE ABREP parametro VR parametro VR parametro VR
parametro FECHAP FIM  {

                    printf("m%5d %1d %1d
%1d\n",paramet[0],paramet[1],paramet[2],paramet[3]);

fprintf(fp,"m%5d%1d%1d%1d\n",paramet[0],paramet[1],paramet[2],pa
ramet[3]);

                    conta=0;

                    }
        | VARRE ABREP parametro VR parametro VR parametro VR
parametro VR parametro FECHAP FIM  {

                    printf("v%5d %5d %5d %5d
%4d\n",paramet[0],paramet[1],paramet[2],paramet[3],paramet[4]);

fprintf(fp,"v%5d%5d%5d%5d%4d\n",paramet[0],paramet[1],paramet[2]
,paramet[3],paramet[4]);

                    conta=0;

                    }

        | TEMPO ABREP parametro FECHAP FIM  {

                    printf("t%d\n",paramet[0]);
                    fprintf(fp,"t%4d\n",paramet[0]);

                    conta=0;

                    }

        |CALIBRA FIM  {
                    printf("c\n");
                    fprintf(fp,"c\n");

                    }

        |HOME FIM     {
                    printf("h\n");
                    fprintf(fp,"h\n");

                    }
        |IRRADIA ABREP parametro FECHAP FIM     {

                    printf("i%d\n",paramet[0]);
                    fprintf(fp,"i%4d\n",paramet[0]);
```

```
                    conta=0;

                    }
        |BOTAO   FIM     {
                    printf("b\n");
                    fprintf(fp,"h\n");

                    }


;

parametro:     exp  {paramet[conta]=$1;conta++;}

;

/*reconhece uma expressão aritmética mantendo a precedência a
esquerda*/

exp:           INT
               |exp MAIS   exp {$$=$1+$3;}
               |exp MENOS  exp {$$=$1-$3;}
               |exp MULT   exp {$$=$1*$3;}
               |exp DIVIDE exp {$$=$1/$3;}
               |MAIS exp       {$$=$2;}
               |MENOS exp      {$$=-1*$2;}
               |ABREP exp FECHAP {$$=$2;}
;


%%


#include "lexyy.c"


int main (void) {

locerro=1;
conta=0;
fp=fopen("saida","w");

yyin=fopen("entrada","r");

erros=fopen("erros","w");


return(yyparse());

fclose(fp);
```

}