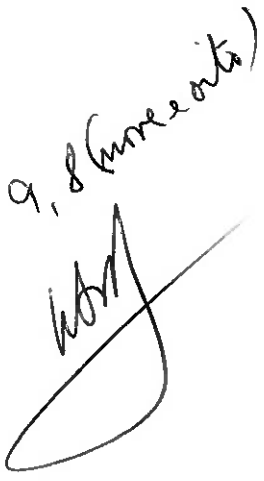
ALENO CHAVES BRAGA

BRENO TORDIN POLLI

JNC – JAVA NUMERICAL CONTROL

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduação em Engenharia.



São Paulo

1998

ALENO CHAVES BRAGA

BRENO TORDIN POLLI

JNC – JAVA NUMERICAL CONTROL

Trabalho de conclusão de curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduação em Engenharia.

Área de concentração:
Engenharia Mecânica – Automação e Sistemas

Orientador:
Marcos Ribeiro Pereira Barretto

São Paulo
1998

Aos nossos pais, pelo apoio e compreensão.

AGRADECIMENTOS

Ao Prof. Marcos R. P. Barretto, nosso orientador, pelo auxílio na arquitetura da solução do projeto.

Ao Marcos Hunold, aluno de pós-graduação da Escola Politécnica, pelo tempo despendido e, em especial, pelas idéias sobre a parte de hardware fornecidas.

Aos amigos Fernando, Érico, João e Júnior, pela ajuda na montagem dos componentes de hardware.

SUMÁRIO

Lista de equações

Lista de figuras

Lista de malhas

Lista de tabelas

Lista de telas

RESUMO

Comando numérico distribuído em células de manufatura têm sido largamente utilizados na indústria mecânica como uma alternativa para produção em larga escala, de modo a reduzir custos e agilizar o processo de produção.

Por outro lado, com o avanço da informação eletrônica através da Internet, software especializados em facilitar a comunicação em rede vêm ganhando espaço no mercado. Dentre eles destaca-se a linguagem de programação Java, largamente utilizada na confecção de provedores de informação.

Com o objetivo de aliar a necessidade das indústrias em relação ao comando numérico com as facilidades fornecidas pelo Java, desenvolveu-se um software baseado em linguagem Java capaz de realizar o posicionamento ao longo de um trilho de uma plataforma contendo um robô, cujo hardware encontra-se em uma célula no laboratório da Escola Politécnica da Universidade de São Paulo, onde ainda estão presentes um torno, uma fresadora e uma estação supervisora.

ABSTRACT

Distributed numerical control at manufacturing cells has been extensively used in mechanical industries as an alternative for large-scale production, so that costs can be reduced and production process can be optimized.

By the other hand, with the electronic information advance through Internet, software specialized in making net communication easier are getting stronger in market. Among them, Java programming language is the better known one and has been widely used in information servers development.

Having the objective of allying the need of industries to the numerical control with the facilities supplied by Java, the development of a Java language software able to realize the positioning of a platform with a robot on it along of a rail was done. The hardware used is in a cell at *Escola Politécnica da Universidade de São Paulo* laboratory, where there also are a lathe, a milling cutter and a supervisory unit.

1 Motivação

Antes de entrar no objetivo central deste projeto, é conveniente citar os conceitos que estão envolvidos no seu aspecto geral. O que este trabalho se propõe a descrever e solucionar é parte de um conjunto de outros projetos que visa colocar em funcionamento uma célula de manufatura existente num dos laboratórios da Escola Politécnica.

Tendências gerenciais recentes deram origem a novos conceitos de arranjo físico com o objetivo de aumentar a produtividade, juntamente com o aumento da qualidade e da redução de espaços ociosos. Em termos de lay-out, a idéia de células de produção é um dos mais importantes conceitos que vem com o intuito de atender estas necessidades. Uma célula é definida como o conjunto de máquinas posicionadas de tal maneira que obedecem à seqüência de operações a serem executadas no produto ou nas peças às quais são destinadas a produzir. Com isso, consegue-se minimizar ao máximo o tempo de "set-up" das máquinas, o transporte de materiais, os estoques intermediários, o espaço ocupado e os tempos de espera, requisitos importantes para uma resposta rápida às novas necessidades de mercado.

Variações deste modelo, como por exemplo as semicélulas, são utilizadas na produção de uma família ou grupo de peças que, embora diferentes, passam por processos de usinagem semelhantes. O uso de semicélulas é uma aplicação bastante interessante do conceito de tecnologia de grupos, que

consiste na idéia de criar códigos para as características e dimensões de produtos de fabricação similar. Depois de analisado, este código permite avaliar um grupo para o qual uma célula é aplicável. Cria-se então uma matriz de processos, especificando as operações para cada codificação.

Seguindo a idéia de diminuir o tempo de fabricação, material em processamento disponível para as esperas e transporte, além de contribuir para a racionalização do trabalho e qualidade do serviço, a automatização dos processos de fabricação foi a mais expressiva conseqüência desta tendência. No processo de desenvolvimento de robôs e máquinas operatrizes de usinagem controladas para atender um nível de automação desejada cada vez maior, surgiram os "Comandos Numéricos" (CN). A união dos conceitos de célula de manufatura e CN deram origem aos Sistemas Integrados de Fabricação.

2 Descrição do problema

A célula de manufatura existente no Departamento de Engenharia Mecânica da EPUSP é composta por um robô ABB com 6 graus de liberdade, um torno CNC (Comando Numérico Computadorizado), uma fresadora CNC e um trilho no qual se apoia o robô, concedendo a este mais um grau de liberdade - o sétimo.

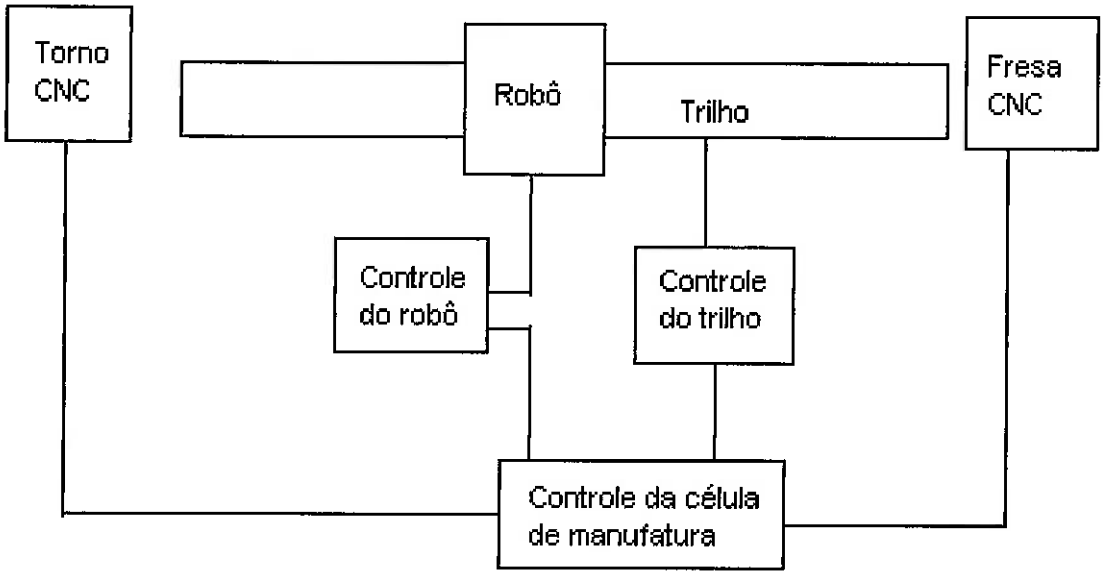


Figura 1 - Célula de manufatura

Com a idéia geral de contribuir para o funcionamento final desta célula como um sistema integrado de fabricação, este trabalho tem por objetivo implementar o controle de posição do trilho para a movimentação do robô, que terá

basicamente a função de transportar a peça em processo de usinagem entre o torno e a fresa.

A garantia de que o robô estará realmente na posição pretendida é muito importante para o perfeito funcionamento dos processos de retirada e fixação da peça a ser usinada nas máquinas operatrizes. Assim, a implementação de um controle com alto nível de precisão e baixa tolerância se torna necessário. Além disso, o controle da velocidade e da aceleração do movimento é necessário para garantir rápido transporte e sincronia de tempo entre as execuções do robô, do torno e da fresa.

Ao se analisar os processos inevitáveis de informatização e de globalização pelos quais as indústrias devem se submeter , é indiscutível que aspectos de acesso de informação são cada vez mais importantes neste final de século e a escolha da ferramenta ideal para o desenvolvimento do sistema é de fundamental importância na garantia de que este poderá ser facilmente integrado a outros sistemas existentes. Com o intuito de permitir o acesso da aplicação por qualquer tipo de plataforma ou máquina como PC's, computadores Macintosh ou servidores, foi escolhido a linguagem Java para a implementação do projeto.

Ponto importante deste trabalho é o engajamento que o projeto se compromete a ter com a evolução das indústrias de manufatura no que se diz respeito a processos, fabricação e tecnologia da informação.

3 Metodologia de solução

O projeto tem fundamentalmente o objetivo de colocar em funcionamento o posicionamento da plataforma onde se encontra o robô da célula de manufatura encontrada no Departamento de Engenharia Mecânica da EPUSP.

Similar ao que já foi pretendido anteriormente com relação a este objetivo, o resultado esperado depois de concluído este projeto é obter um valor de erro dentro do previsto para o posicionamento pré-definido da plataforma. O erro deve ter um valor máximo de modo a garantir o perfeito processo de colocação e retirada das peças a serem usinadas nas máquinas operatrizes pelo robô, resultando no adequado transporte de peças entre o torno e a fresa. Para se atingir esta meta, o projeto se dispõe a oferecer soluções de controle, hardware e software

3.1 Controle

Com relação ao projeto de controle, foi escolhido um modelo do tipo PID e seu equivalente digital para que o mesmo seja implementado, através de equações de diferenças, em um computador. Para se determinar este controlador, é necessário modelar o sistema trilho, motor, redutor e sensor de forma a obter sua função de transferência. O levantamento desta função é feito de forma experimental colocando-se no acionamento do motor um sinal de referência na

forma de degrau. Analisando a resposta do sistema para esta entrada pode-se determinar sua ordem, pólos e zeros.

Pretende-se, com o projeto do controlador, garantir a plataforma no local certo com rápido tempo de posicionamento. No entanto, quanto mais preciso o posicionamento, maior o tempo gasto para realizá-lo. Ao contrário, com um posicionamento muito rápido, o controle da posição fica prejudicado.

Assim define-se como requisito de projeto para o sistema controlador-trilho que o sobressinal deve ser zero, com isto a plataforma irá tender para a posição definida sem oscilar em torno da mesma. Além disto, a resposta do sistema, ou seja, o tempo de assentamento deve ser o mais rápido possível.

Para atender a estes requisitos, deve-se ter uma solução de compromisso, já que sistema muito rápidos apresentam sobressinal elevado. Como conseqüência, o sistema deve ter coeficiente de amortecimento igual a 1 para garantir resposta mais rápida quando sobressinal não é desejado. Além disso, o pólo de operação do sistema deverá ser escolhido de forma a tornar o controlador imune a possíveis variações no trilho. Isto é necessário já que a massa sobre a plataforma irá variar dependendo da peça que o robô estiver carregando.

3.2 Hardware

A solução de hardware seguiu o que se tinha disponível na célula. No entanto, alguns destes equipamentos estavam desligados, não funcionando ou desaparecidos. Portanto, foi parte deste projeto verificar os equipamentos em funcionamento e providenciar os necessários para o funcionamento final do trilho, tanto pela aquisição de soluções prontas no mercado quanto desenvolvimento e montagem de soluções dedicadas.

A solução consiste de uma unidade supervisora e controladora de nível global (estação de trabalho) que envia e recebe informações de todos os equipamentos que compõe a célula. Um resumo das características dos equipamentos utilizados se encontra relacionado abaixo:

1. Trilho de 259 mm projetado e construído com metal e madeira na própria faculdade, com uma guia para permitir deslocamento da plataforma;
2. Plataforma quadrada de madeira e base de metal de 52 mm onde será fixado o robô e que se movimenta sobre o trilho através de rolamentos que são fixados na parte superior e inferior da guia do trilho, permitindo somente o movimento em uma única direção. A plataforma possui uma guia para ser percebida pelos sensores de fim de curso e, além disso, é amarrada a um cabo de aço que se enrola à um tambor com 9 voltas;

3. Tambor de aço com 120 mm de diâmetro no qual é enrolado o cabo de aço que é ligado à plataforma. De um lado deste tambor é ligado o redutor e do outro o gerador de pulsos (encoder);
4. Redutor com relação de 1:10 com entrada e saída perpendiculares de marca CESTARI tipo K-40, com potência de 1,08 CV a 1750 rpm;
5. Gerador de pulsos óptico (encoder) de 5000 raias, marca Diadur, com dois discos graduados que permitem discernir se este está girando no sentido horário ou anti-horário;
6. Servo-motor de corrente contínua de marca VARIMOT S.A. modelo SD-2530 com as seguintes características:

    - Tensão máxima: 130 Vcc
    - Torque: 3 N.m
    - Rotação de saída em vazio: 3000 rpm
    - Tacogerador: 9,5V/1000 rpm.

    Este servo-motor é conectado à entrada do redutor por uma junta fixa;
7. Um servo-amplificador (parametrizável) de marca TRANSAXE série 700 modelo 710 (conversor estático para motores CC) com uma malha controlada por um circuito proporcional integral, realimentado por um tacogerador e que trabalha com pulsos (PWM - Pulse Width Modulator ou modulador de largura de pulsos). Tem saída para o motor e entrada do micro, que irá fornecer o sinal de referência;
8. Placa de interface microcomputador-hardware de marca SENSORAY modelo 421, contadora de pulsos que serão gerados pelo encoder, conversora digital/analógica para gerar sinal para o servo-amplificador e leitora de entradas digitais vindas das chaves fim-de-curso.

9. Dois microcomputadores com placa de comunicação (rede), sendo um para a parte da aplicação no local do trilho (workstation) e outro para a parte remota (server);

3.3 Software

De acordo com solução proposta, a estação de trabalho que coordena a célula enxerga o sistema como um comando numérico e envia comandos em linguagem G, reconhecida pelo controlador da célula.

Para isto, desenvolveu-se um compilador de linguagem G com a finalidade de controlar o trilho. A interface homem-máquina foi desenvolvida em Java, utilizando o aplicativo Visual Age for Java (IBM), e as funções que realizam a troca de sinais com a placa de interface colocada no micro (placa de entradas digitais, conversora digital-analógica e contadora de pulsos do encoder), em linguagem C/C++.

O software é ainda responsável pelo acionamento de um motor que movimenta a plataforma sobre o trilho e pelo controle de posição da plataforma. Sua implementação deve seguiu os critérios do projeto de controle, com o PID sendo realizado em malha semi-fechada, realimentado por um encoder.

4 Conceitos e fundamentos teóricos

4.1 Sistemas integrados de fabricação

Sistemas Integrados de Fabricação são constituídos por duas ou mais máquinas com "Comando Numérico" interligadas entre si com trocadores e transportadores automáticos de peça, máquinas estas que perfazem operações conseqüentes ou simultâneas.

A célula de fabricação normalmente é supervisionada por uma central de processamento de dados que coordena todos os CNC envolvidos, sistemas de trocas de peças, de programas e outros eventos.

Todo o conceito para definição de uma célula flexível de fabricação reside na teoria da Tecnologia de Grupo, que através do estudo de peças por família, estuda o ciclo de fabricação envolvendo processos, programas, ferramentas, dispositivos, trocadores e transportadores de peças, dispositivos e equipamentos de medição durante o processo de usinagem, de modo que todas as peças constituintes desta família tenha fluxo através da célula com a mínima intervenção do homem e com a maior flexibilidade possível com relação a tipos e quantidades. [MACHADO, 1987]

4.2 Comando numérico

O "Comando Numérico" é um equipamento eletrônico capaz de receber informações por meio de entrada própria, compilar estas informações e transmiti-las em forma de comando à máquina operatriz, de modo que esta, sem a intervenção do operador, realize as operações na seqüência programada.

4.2.1 Comandos numéricos com minicomputador ou microprocessadores (CNC)

A grande diferença do CNC é a existência de um minicomputador interno como comando. Portanto, se for necessário acrescer um recurso a mais no sistema, este recurso vem geralmente em forma de um programa. Para os comandos numéricos comuns, um aumento de recursos requerido implica em aumento de circuitos eletrônicos e componentes, ou seja, este recurso vem através de um aumento físico do comando. Por outro lado, outra característica essencial do CNC é a sua capacidade elevada de arquivo de programa.

A maioria dos CNC já não adotam mais os minicomputadores e sim os microprocessadores, que levam à diminuição do custo e redução do tamanho.

4.2.2 Comando numérico controlado por uma unidade central de computador (DNC)

É um sistema que consiste em se ter várias máquinas com CN, sendo que estes CN estão ligados a um computador central, o qual pode atuar de duas formas.

1. São os chamados sistemas de arquivos de programa, onde o computador central tem, em seu arquivo, todos os programas feitos para as máquinas, as quais estão a este interligadas. O programa vem à máquina e é arquivado na memória do CNC, o qual é executado bloco a bloco. O computador, desta maneira, pode controlar várias máquinas simultaneamente, operando com programas e peças diferentes. Cada máquina desta possui seu próprio CNC sem leitora, conectado à central.
2. Neste caso, um complexo sistema de máquinas está interligado a um computador central, que além de conter arquivado todos os programas, ainda controla diretamente cada máquina, englobando, portanto, a unidade de entrada de dados e a unidade de controle.

4.2.3 Vantagens do comando numérico

Pelas características citadas acima, pode-se destacar:

1. Redução da mão-de-obra direta e do tempo de execução;

2. Diminuição do custo do ferramental;
3. Liberdade para os projetos - versatilidade e flexibilidade;
4. Protótipos mais baratos e maior versatilidade de produção;
5. Fidelidade na especificação;
6. Tecnologia de fabricação armazenada pela empresa;
7. Controle de qualidade fica mais fácil e pode ser reduzido;
8. Redução das operações secundárias e compactação de ciclos;
9. Redução de tempo de montagem;
10. Redução de inventário da produção;
11. Redução de manuseio de material;
12. Redução de quantidade de máquinas e, consequentemente, da área utilizável;
13. Possibilidade de maior enquadramento para controle através do computador.

4.3 Tecnologia Java

A idéia defendida através de aplicações desenvolvida com o Java é simples: funcionar em qualquer sistema operacional que esteja rodando em algum hardware - dos menores dispositivos aos supercomputadores. A única condição necessária é ser um hardware que suporte a plataforma Java. O exemplo mais simples de plataforma Java é aquela onde se tem um Web browser.

4.3.1 JavaSpaces

Com o crescimento da Internet e das tecnologias de rede, a necessidade de se ter aplicações distribuídas de fácil desenvolvimento e manutenção se torna cada vez mais importante. A tecnologia JavaSpaces é um meio de construir aplicações distribuídas para intranets, extranets e para Web. Trata-se de um mecanismo pelo qual se consegue compartilhar, coordenar e permitir a comunicação de recursos, objetos e serviços em uma network.

4.3.2 Java RMI

A plataforma Java oferece um acesso central de linguagem que simplifica a programação de objetos distribuídos. Java Remote Method Invocation (RMI) permite que objetos Java sejam executados em diferentes Java Virtual Machines (JVM's), propiciando ambientes heterogêneos de comunicação. Como a tecnologia RMI torna possível toda a programação de aplicações distribuídas em pura linguagem Java, esta possui características como portabilidade de código, carregamento dinâmico, segurança e criação de grupos de refugo. A tecnologia RMI suporta também objetos de polimorfismo e pass-by-value distribuídos, características que sistemas tradicionais distribuídos não conseguiam prover.

4.4 Controlador PID

Os controladores PID são muito freqüentemente usados em sistemas de controles industriais. A função de transferência Gc(s) do controlador PID é:

$$Gc(s) = Kp.(1 + \frac{1}{Ti.s} + Td.s)$$

Kp: ganho proporcional

Ti: tempo integral

Td: tempo derivativo

Equação 1 - Função de transferência do controlador PID em s

Se e(t) for a entrada do controlador PID, a saída u(t) do controlador é dada por:

$$u(t) = Kp.[e(t) + \frac{1}{Ti}\int_{-\infty}^{t} e(t).dt + Td.\frac{de(t)}{dt}]$$

Equação 2 - Função de transferência do controlador PID no tempo

Se um modelo matemático da planta puder ser deduzido, então é possível aplicar várias técnicas de projeto para determinação dos parâmetros do controlador que satisfazem às especificações transitórias e de regime estacionário do sistema de malha fechada. No entanto, se a planta é tão complicada que seu modelo matemático não pode ser facilmente obtido, a abordagem analítica do projeto de um controlador PID não é possível. Recorre-se então à técnicas experimentais para o projeto de controladores [OGATA, 1990].

5 Arquitetura de hardware

Com base na descrição do problema apresentada, propõe-se a seguinte estrutura para o hardware do projeto:

5.1 Arquitetura de hardware proposta

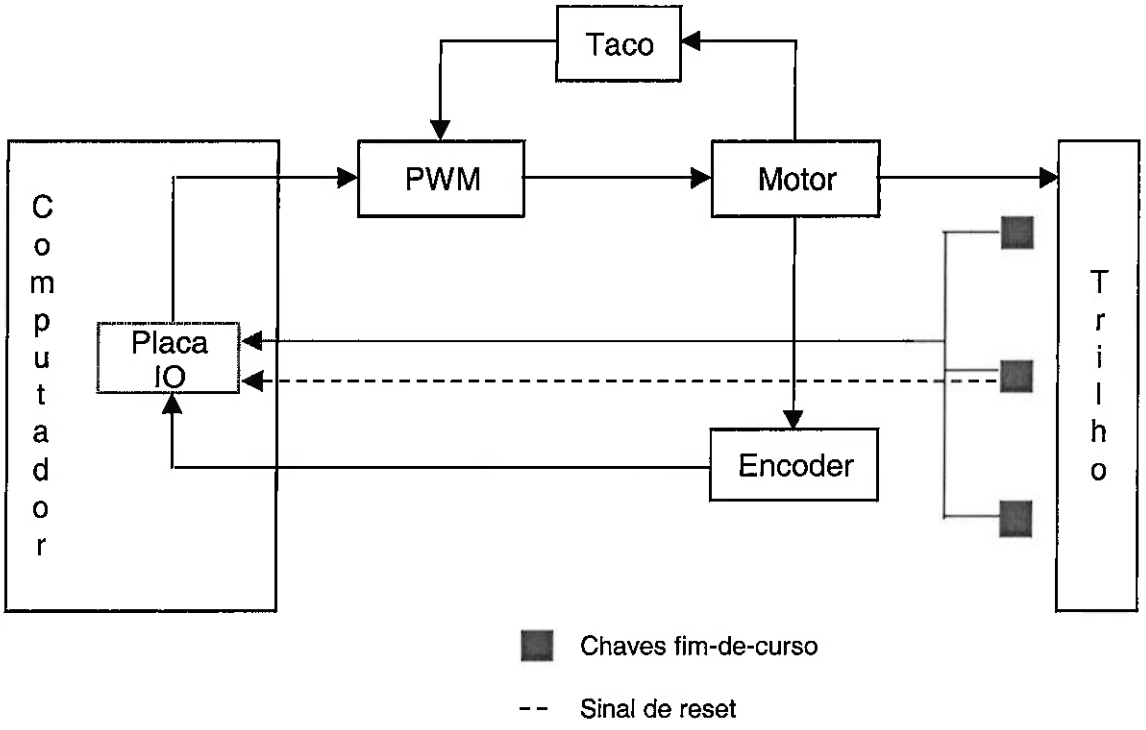


Figura 2 – Arquitetura de hardware

A parte mecânica proposta para a célula pode ser representada pelo esquema acima.

No microcomputador, onde está o controlador da planta e a interface homem-máquina, o sinal de controle é gerado e convertido pela placa de IO, que possui função de conversora D/A, chegando ao PWM, responsável pelo acionamento do motor e, consequentemente, da plataforma onde se encontra o robô. Este acionamento possui um controle proporcional integrativo para tensão e corrente em seu interior, necessitando da realimentação do tacogerador para um perfeito funcionamento.

A leitura da posição da plataforma é feita através da posição do motor, utilizando-se um encoder, que gera dois sinais defasados de 90º, permitindo à placa de IO, que possui função contadora de pulsos, saber em qual sentido está sendo o giro do motor. O valor da contagem de 16 bits é utilizado pelo microcomputador, que a utiliza para gerar o novo valor de controle. O sinal de reset proveniente da chave fim-de-curso localizada no centro do trilho é utilizado para zerar a contagem.

Além disso, o microcomputador recebe os sinais das chaves de fim-de-curso também através da placa de IO, que possui função de leitura de entradas digitais, para que possa parar a movimentação da plataforma e informar ao operador a anormalidade no posicionamento.

5.2 Placa controladora do servo-amplificador (PWM)

Utilizou-se um servo-amplificador Transaxe série 700 modelo 710 fabricado pela IPSO Automatização Ltda. O módulo é destinado ao comando de servo-motores de corrente contínua, possuindo um rack e um módulo de acionamento.

5.2.1 Características do módulo de acionamento

Tem-se uma placa formato dupla-europa, cujas principais características são:

- Corrente

  Regime: 12,5A;

  Máxima: 25A.
- Potência: 2000W.
- Máximo sinal de referência: ±10V.
- Máxima tensão de tacômetro: 100 V.
- Faixa de temperatura em operação: 0 – 45ºC.
- Tensão de alimentação: 50 – 160 V_{dc} ou 3 x 40 – 115V AC ±10% 50/60 Hz.
- Capacitor de filtro de 1000μF / 250V.
- Dissipação instantânea do excedente da energia de frenagem.
- Regulação de velocidade PI.
- Regulação de corrente PI.

- Controle em quatro quadrantes, com pulsos de potência modulados em PWM.
- Regulação de velocidade tanto por meio de tacômetro quanto por FCEM.
- Faixa de regulação: maior que 1:20.000 (com taco).
- Precisão da regulação: menor que 0,5% com velocidade máxima.
- Fator de forma: ff < 1,01 com corrente máxima.
- Proteções incorporadas:

  Corrente eficaz;

  Falta de taco;

  Curto-circuito;

  Termostato;

  Limitação automática da corrente em função da temperatura;

  Limitação da corrente em função da rotação;

  Supervisão sobre tensão no momento da frenagem do motor.

- Indicação do estado do servo-amplificador por meio de LED:

  Fonte;

  Falha de tacogerador;

  Sobre corrente;

  Stand-by;

  Sobre temperatura;

  Dispositivo de frenagem acionado.

- Regulagens no frontal do equipamento:

  Ganho;

  Corrente máxima;

  Corrente eficaz;

Tacômetro;

Offset.

- Funções auxiliares:

  Entrada de stand-by em 24V opto-acoplada;

  Duas entradas de fim de curso em 24V opto-acopladas;

  Entrada de tensão de referência nula em 24V opto-acoplada;

  Sinal de "servo ready" através de contato de relé;

- Saída do sinal analógico da corrente instantânea.
- Saída de +15V, terra e –15V (máx. 20mA).

5.2.2 Conexões dos sinais de comando

Na placa de conexão existem 16 bornes para a ligação dos sinais de comando:

- Bornes 1 e 2: saída dos contatos do relê "ready".

  Estas saídas, livres de potencial, podem indicar ao comando numérico que o servo-amplificador está em funcionamento, com o contato "stand-by" fechado e com o módulo sem alarme.

- Bornes 3, 4 e 5: saídas das tensões +15V, 0V e –15V.

  Estas saídas colocam à disposição do usuário as tensões de ±15V produzidas no servo-amplificador por uma fonte chaveada. A corrente nominal é de 20mA.

- Bornes 6 a 10: entrada dos sinais opto-acoplados.

Borne	*Descrição*
6	"Stand-by"
7	Velocidade nula
8	Fim de curso 1
9	Fim de curso 2
10	Entrada de referência da tensão opto-acoplada

Estas entradas funcionam com uma tensão de 24V contínua.

O servo-amplificador estará em "stand-by" se o contato SBY estiver aberto e estará liberado se o contato for fechado, sendo ainda indicado pelo LED SBY do frontal.

Uma tensão de referência nula será imposta com o contato OSW (borne 7) aberto, e estará livre para o comando externo com o contato fechado.

Uma tensão de referência externa positiva será anulada assim que o contato SW1 (borne 8) for aberto e estará respondendo ao comando de tensão negativa externa.

Uma tensão de referência externa negativa será anulada assim que o contato SW2 (borne 9) for aberto e estará respondendo ao comando de tensão positiva externa.

- Bornes 11, 12 e 13: entrada do tacogerador.

Borne	*Descrição*
11	Conectado ao + do tacogerador
12	Conectado ao - do tacogerador
13	Conectado à blindagem do cabo do tacogerador

- Bornes 14, 15 e 16: entrada da tensão de referência.

  Aos bornes 14 e 16 devem ser conectados o sinal de referência proveniente do controlador. Os níveis máximos de tensão são ±10V. Ao borne 15 deve ser conectado a blindagem do cabo da tensão de referência.

Para a aplicação, foram utilizados os seguintes sinais do PWM:

Pino	*Descrição*
3	Saída de tensão +15V*
4	Saída de tensão 0V*
6	Stand by*
10	Entrada de referência da tensão opto-acoplada*
11	Sinal + do tacogerador
12	Sinal - do tacogerador
13	Blindagem do cabo do tacogerador
14	Sinal de referência para girar em sentido anti-horário
15	Blindagem do cabo da tensão de referência
16	Sinal de referência para girar em sentido horário

Tabela 1 - Sinais do PWM utilizados

*Esses sinais não são efetivamente utilizados na operação da plataforma. Utilizou-se uma chave física, onde como entradas ligou-se a tensão de referência vinda do pino 10, que foi curto-circuitado com o pino 4, e a tensão de +15V vinda do pino 3, e como saída ligou-se o sinal de stand by, no pino 6. Com isso, ao se acionar a chave, o PWM não gera sinal para a saída e o motor fica parado. Isso é utilizado ao se ligar o PWM quando não se sabe qual sinal de referência se tem ou ainda em uma eventual emergência.

5.2.3 Configuração dos elementos variáveis

Como a placa pode ser utilizada em uma grande variedade de aplicações, alguns componentes devem ser dimensionados pelo usuário conforme a necessidade e soldados em seus respectivos lugares, indicados no diagrama da placa localizado no *Anexo A Esquema da placa de acionamento do servo-motor.*

Para o dimensionamento, foi considerada a regulação de velocidade com tacogerador. Seguindo [7] (MANUAL de instalação Transaxe série 700 Servo-amplificadores 705 e 710), tem-se:

- Colocar o jumper W1 na posição 1-2.

- Calcular e soldar a resistência R_2:

$$R_2 < \frac{250}{(V_{din} \cdot n_{max}) - 10} \text{ (kOhm)}$$

Onde:

V_{din}: tensão do dínamo;

n_{max}: rotação máxima.

Equação 3 - Dimensionamento do resistor R_2 da placa PWM

A tensão no dínamo, dada V/1000 rpm, e rotação máxima, dada em milhares de rpm, são características do servo-motor e, para o projeto, valem 9,5V/1000rpm e 3000 rpm. Assim:

$$R_2 < \frac{250}{(9{,}5 \cdot 3{,}0) - 10} = 13{,}5 \text{ kOhm}$$

Equação 4 - Cálculo do resistor R_2 da placa PWM

- Colocar o jumper W3 na posição "on", uma vez que será .utilizada a supervisão do tacogerador.
- Calcular e soldar as resistências R_{43} e R_{45}:

$$R_{43} = R_{45} = \frac{5400}{K_e \cdot n_{max}} \text{ (kOhm)}$$

Onde:

K_e: constante elétrica do motor;

n_{max}: rotação máxima.

Equação 5 - Dimensionamento dos resistores R_{43} e R_{45} da placa PWM

Para o projeto, a constante elétrica do motor, dada pelo fabricante, vale 45V/1000rpm e a rotação máxima, conforme apresentado, 3000 rpm. Assim:

$$R_{43} = R_{45} = \frac{5400}{45 \cdot 3{,}0} = \quad 40 \text{ kOhm}$$

Equação 6 - Cálculo dos resistores R43 e R45 da placa PWM

- Deixar o resistor R44 que vem de fábrica.
- Ajustar o trimpot P8 no mínimo (totalmente no sentido horário).
- Ajustar o trimpot P9 no mínimo (totalmente no sentido anti-horário).

5.3 Encoder

Conforme anteriormente mencionado, utilizou-se um encoder Diadur com 5000 raias. Os seguintes sinais presentes no cabo do encoder foram utilizados para interface com a placa de IO:

Pino	*Descrição*
1	Fase B-
5	Fase A
6	Fase A-
8	Fase B
10	GND (terra)
12	+5V (alimentação)

Tabela 2 - Sinais do encoder utilizados

5.4 Chaves de fim-de-curso ópticas

Para a montagem das chaves, comprou-se os sensores ópticos e montou-se o seguinte circuito:

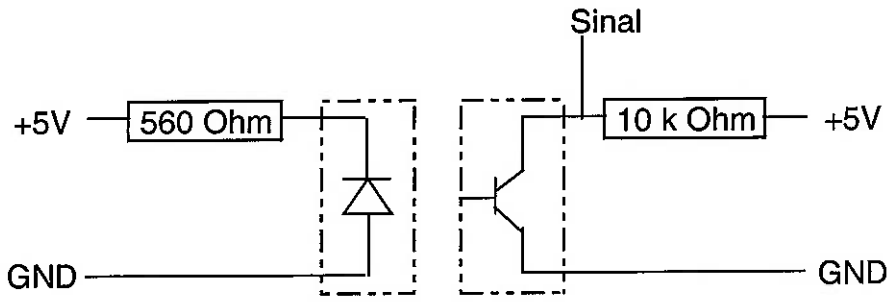


Figura 3 - Circuito do sensor fim-de-curso óptico

Pelo circuito, limita-se a corrente do diodo pelo resistor de 560Ω. Quando não há barreira entre as duas partes do sensor (linhas pontilhadas), o transistor fotoaclopado do lado direito trabalha em saturação e a corrente que flui pelo sinal é nula. Quando há uma barreira, o transistor fotoaclopado entra em corte, e passa a fluir uma corrente pelo sinal, no valor limitado pelo resistor de 10kΩ.

5.5 Placa de IO do microcomputador

A placa de IO utilizada possui várias funções, conforme mencionado, e seu endereço base de IO pode ser configurado por jumpers. No projeto original, este endereço foi colocado em 0x0F00.

5.5.1 Entradas digitais das chaves fim-de-curso

Utilizou-se os conectores P5 e P6 e os seguintes pinos:

Pino	*Descrição*
2	+5V (alimentação das 3 chaves)
3	GND da chave próxima ao torno (esquerda)
4	Sinal da chave próxima ao torno (esquerda)
5	GND da chave central
6	Sinal da chave central
7	GND da chave próxima à fresa (direita)
8	Sinal da chave próxima à fresa (direita)

Tabela 3 - Sinais da placa de IO relativos às chaves fim de curso

5.5.2 Contadora de pulsos do encoder

Utilizou-se o conector P3 e os seguintes pinos:

Pino	*Descrição*
5	+5V (alimentação)
6	Fase A
7	GND (terra)
8	Fase B
16	Fase A-
19	Fase B-

Tabela 4 - Sinais da placa de IO relativos ao encoder

5.5.3 Saídas analógicas para PWM

Utilizou-se o conector P2 e os seguintes pinos:

Pino	*Descrição*
1	Sinal de referência para girar no sentido horário
3	GND de referência do sentido horário
5	Sinal de referência para girar no sentido anti-horário
7	GND de referência do sentido anti-horário

Tabela 5 - Sinais da placa de IO relativos ao PWM

OBSERVAÇÃO: para que o PWM faça o motor girar nos dois sentidos, deve receber como sinais de referência valores variando de -10V a +10V. No entanto, a placa de IO utilizada não consegue gerar sinais analógicos negativos, podendo variar somente de 0 a +10V. A solução adotada foi:

- Para que o motor fique parado, coloca-se 0V nos dois sinais;
- Para que o motor gire no sentido horário, coloca-se +10V no sinal equivalente (pino 1) e 0V no outro sinal (pino 5);
- Para que o motor gire no sentido anti-horário, coloca-se +10V no sinal equivalente (pino 5) e 0V no outro sinal (pino 1);

6 Arquitetura de software

O parte de software do problema pode ser resumida nos seguintes pontos:

- Definição das funções de programação em linguagem G;
- Definição e projeto das classes a serem utilizadas no software;
- Implementação das classes projetadas;
- Projeto do controlador PID;
- Teste do software em conjunto com a parte de hardware.

Seguindo essa estrutura proposta, foram desenvolvidos os seguintes pontos:

6.1 Definição das funções de programação

Após o estudo das funções disponíveis na linguagem G, definimos aquelas que se aplicam ao projeto.

6.1.1 Conceito de função modal e não modal

Uma função é dita modal quando, uma vez programada, é válida para todo o programa, até que uma função que a cancele seja programada.

Uma função é dita não modal quando é válida apenas no bloco que a contém. Ou seja, toda vez que for necessária, deve-se reprogramá-la.

6.1.2 Função número de seqüência

A programação do número de seqüência é feita pela função 'n', cujo formato é n seguido de uma seqüência de quatro dígitos (ex.:n0010), que é usada para identificar o bloco de programação.

Deve estar presente em todas as linhas do programa para que o compilador possa saber a seqüência de execução em um eventual desvio no código.

6.1.3 Funções preparatórias

- g00 – Modo de Posicionamento

  Sob este modo, o posicionamento é feito de maneira precisa até a posição programada, em avanço rápido, definido como a maior velocidade permitida pelo sistema.
  É uma função modal, cancelada por g01.

- g01 – Interpolação Linear

  A interpolação linear faz com que a plataforma mova-se em velocidade programada de avanço. O vetor velocidade de avanço terá a direção do movimento.
  É uma função modal, cancelada por g00.

- g04 – Permanência

  Sob este modo, o programa permanece parado no tempo previsto e, esgotado este tempo, parte para a execução do próximo bloco.
  O tempo de permanência desejado, em segundos, é fornecido pela função 'x'.
  É uma função não modal.

- g90 – Programação com sistema de coordenadas absoluta

  A programação é feita de modo que todas as coordenadas se refiram a um sistema de coordenadas absolutas. A origem do eixo de movimentação no trilho corresponde ao centro do mesmo. No entanto, esta referência pode ser alterada pela função g92.
  É uma função modal, cancelada por g91.

- g91 – Programação com sistemas de coordenada incremental

  O posicionamento especificado em um bloco é feito em relação ao ponto anterior e não em relação ao zero do trilho.
  É uma função modal, cancelada por g90.

- g92 – Definição da origem do sistema

  Deve ser complementada pela função 'x', onde é especificado, em relação ao centro do trilho, o novo zero do eixo de movimentação.
  É uma função modal, cancelada por g93.

- g93 – Zera trilho

  Esta função retorna o zero de referência para seu valor padrão, ou seja, para o centro do trilho.
  É uma função modal, cancelada por g92.

6.1.4 Função de posicionamento

Esta função é designada por 'x', seguida de quatro dígitos com sinal (ex.:x+0010). Pode ser utilizada de várias maneiras, sendo sempre não modal:

- quando utilizada com as funções g00 e g01, especifica a posição a ser ocupada pelo robô, podendo ser em relação ao zero do sistema ou à posição anterior, dependendo do sistema de coordenadas escolhido (funções g90 e g91).
- quando utilizada com a função g04, especifica o tempo de espera do programa, em segundos.
- quando utilizada com a função g92, especifica a nova origem do sistema de coordenadas.
- quando utilizada com as funções m03 ou m05, define o número da linha para a qual o programa será desviado.

6.1.5 Função auxiliar de avanço

Esta função é designada por 'f', seguida de três dígitos (ex.:f010). É utilizada com a função g01 e define a velocidade de avanço na direção do movimento do trilho, em porcentagem de máxima velocidade permitida (variando entre 1% e 100%, ou seja, f001 a f100). Caso não seja especificada no início do programa, o compilador entende que se deseja a máxima velocidade (100%). É uma função modal.

6.1.6 Função posição do bit

Esta função é designada por 'b', seguida de dois dígitos (ex.:b17). Pode ser utilizada com as funções m04, m05 ou m06 e indica qual a posição do bit que se está esperando, testando ou setando na palavra de 32 bits. Logo, pode variar de 00 a 31. É uma função não modal

6.1.7 Função valor do bit

Esta função é designada por 'v', seguida de um dígito (v0 ou v1). Pode ser utilizada com as funções m04, m05 ou m06 e indica qual o valor que se está esperando, testando ou setando no bit que está sendo tratado dentro da palavra de 32 bits. É uma função não modal.

6.1.8 Função mensagem para o operador

Esta função é designada por 's', seguida de oitenta caracteres, sem aspas (“). Deve ser utilizada com a função m26, fornecendo a mensagem a ser colocada na tela para o operador do sistema.

Uma vez colocada na tela, a mensagem permanece até que outra seja colocada em seu lugar ou que o operador a reconheça.

6.1.9 Funções miscelânea

- m00 – Parada do programa

  Habilita uma tecla de "Continua" na tela para o operador e pára o programa, até que a tecla seja pressionada.
  É uma função não modal.

- m02 – Fim do programa

  É uma função não modal, obrigatória para se identificar o final do programa.

- m03 – Desvio do programa

  Desvia a execução do programa para um determinado bloco, cujo número de seqüência estará representado pela função 'x'.
  É uma função não modal.

- m04 – Espera sinal

  Aguarda até que um determinado bit de uma palavra de 32 bits, escrita através da rede por uma outra estação de trabalho, atinja o valor desejado (0 ou 1).
  Deve ser usada com a função 'b', que irá dizer qual bit está sendo esperado (0 a 31) e com a função 'v', que irá dizer qual valor está sendo esperado (0 ou 1).
  É uma função não modal.

- m05 – Testa sinal (Desvio Condicional)

  Testa se um determinado bit de uma palavra de 32 bits, escrita através da rede por uma outra estação de trabalho, possui o valor lógico desejado e, caso seja, desvia o programa para uma determinada linha.

Deve ser usada com a função 'b', que irá dizer qual bit está sendo testado (0 a 31), com a função 'v', que irá dizer qual o valor que está sendo testado (0 ou 1) e com a função 'x', que irá dizer qual o bloco para onde o programa será desviado, caso a condição seja verdadeira.
É uma função não modal.

- m06 – Seta sinal

  Seta um determinado bit de uma palavra de 32 bits e a envia pela rede até uma outra estação de trabalho.
  Deve ser usada com a função 'b', que irá dizer qual bit está sendo setado (0 a 31) e com a função 'v', que irá dizer qual o valor a ser colocado no bit (0 ou 1).
  É uma função não modal.

- m26 – Imprime mensagem na tela

  Imprime a mensagem desejada na tela, representada pela função 's'. É uma função modal, ou seja, a mensagem permanece na tela até que uma outra seja programada ou que o operador a reconheça.

6.1.10 Final de bloco

O interpretador de linguagem G considera o caracter ; (ponto-e-vírgula) como indicador de final de bloco.

6.1.11 Sintaxe da linguagem G

Dependendo do comando a ser executado, a sintaxe pode sofrer pequenas alterações. Para as diversas possibilidades, tem-se:

OBSERVAÇÃO: todos os zeros à esquerda nos números devem ser digitados para que o compilador reconheça o comando. Por exemplo, n10 deve ser escrito como n0010.

6.1.11.1 Modo de posicionamento

n4 (g00) *x±4;*

6.1.11.2 Interpolação linear

n4 (g01) *x±4 (f3);*

6.1.11.3 Permanência

n4 g04 *x±4;*

6.1.11.4 Programação com sistema de coordenadas absoluta

n4 g90;

6.1.11.5 Programação com sistema de coordenadas incremental

n4 g91;

6.1.11.6 Definição da origem do sistema

n4 g92 *x±4;*

6.1.11.7 Zera trilho

n4 g93;

6.1.11.8 Parada do programa

n4 m00;

6.1.11.9 Fim do programa

n4 m02;

6.1.11.10 Desvio do programa

n4 m03 *x±4;*

6.1.11.11 Espera sinal

n4 m04 *b2 v1;*

6.1.11.12 Testa sinal

n4 m05 *b2 v1 x±4;*

6.1.11.13 Seta sinal

n4 m06 *b2 v1;*

6.1.11.14 Imprime mensagem

n4 m26 *s80;*

6.2 Definição das classes

A solução adotada para o problema engloba duas aplicações, uma denominada de JNCW (Java Numerical Control Workstation), que será executada no computador que contem a placa de IO e realiza a interface com o hardware, e outra denominada JNCS (Java Numerical Control Server), que será executada em outro computador que estiver na mesma rede que aquele em que o JNCW estiver sendo executado.

Com base nessa divisão, serão definidas as classes que serão utilizadas na implementação do software responsável pela interface homem-máquina e pelo controle de posicionamento da plataforma que contém o robô.

6.2.1 JNCW (Java Numerical Control Wokstation)

Através desta aplicação, pode-se basicamente movimentar a plataforma até uma posição desejada, editar e executar programas contendo a linguagem G. Para um maior detalhamento da aplicação e das opções, veja o *Anexo D.a.Java Numerical Control Workstation*.

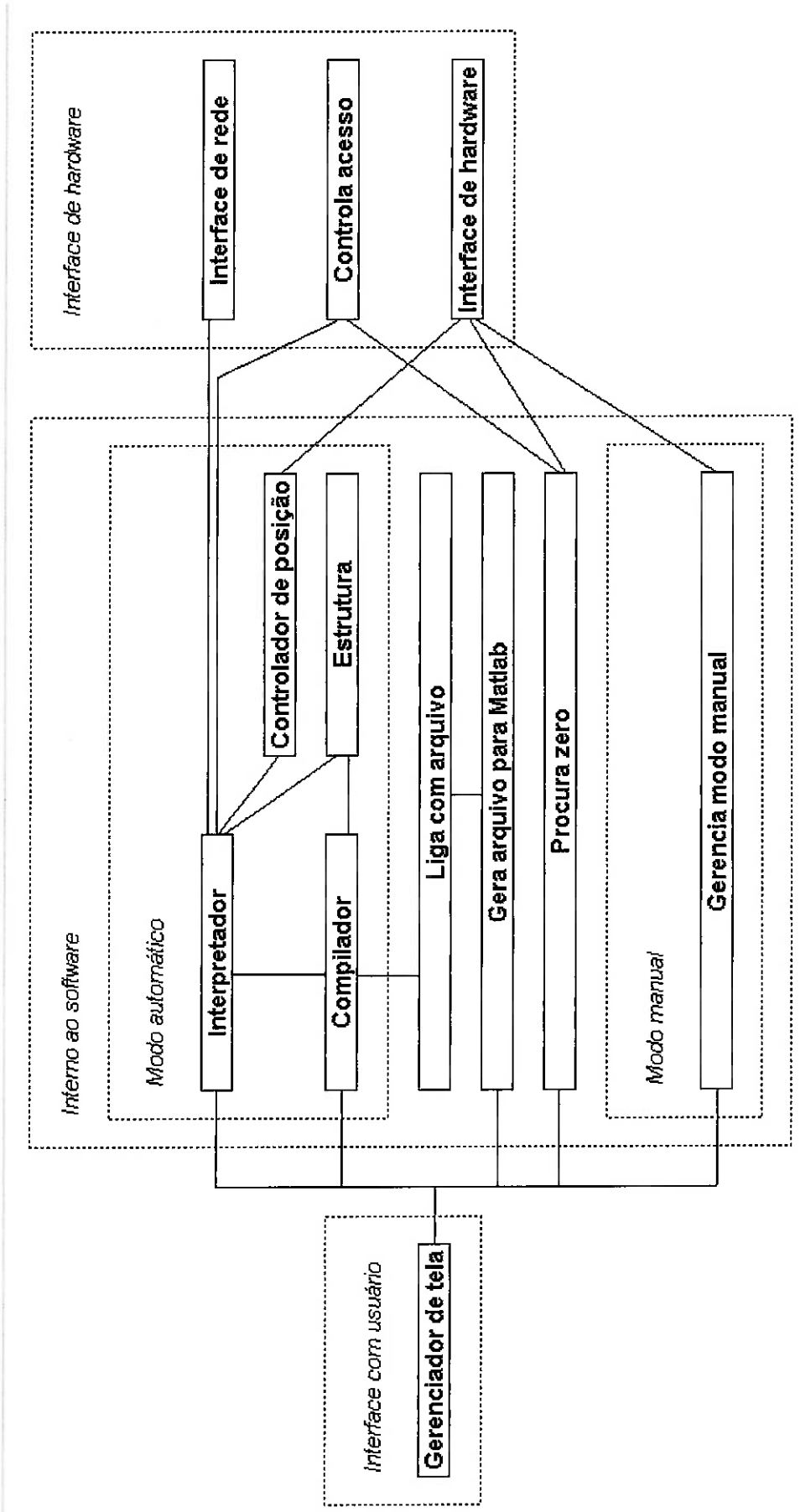


Figura 4 - Estrutura de classes do JNCW

6.2.1.1 Interface com usuário

Este bloco, composto somente pela classe Gerenciador de tela, realiza a interface gráfica com usuário, onde se pode selecionar arquivo, modo de operação e acompanhar a posição da plataforma em tempo real, entre outros.

6.2.1.2 Interno ao software

Este bloco está subdividido em três: um para operação da plataforma em modo manual, outro para modo automático e um terceiro que é geral e não está englobado em nenhum modo específico.

No modo automático, tem-se:

1. Estrutura: fornece a estrutura de dados para armazenagem das funções do programa que contém a linguagem G, depois de compilado;
2. Compilador: responsável por compilar o programa que possui a linguagem G e transformar as funções em símbolos que possam ser compreendidos pelo computador, verificando se a sintaxe de programação está correta;
3. Interpretador: responsável pela execução das funções existentes na estrutura de dados após a compilação do programa;
4. Controlador de posição: responsável pelo controle do posicionamento da plataforma na posição desejada, utilizando para isso um controlador PID.

No modo manual, tem-se somente a classe Gerencia Modo Manual, que realiza toda a supervisão de posicionamento e velocidade da plataforma quando o sistema estiver sendo operado em modo manual. Para isso, recebe e envia informações e dados tanto do bloco de *Interface com usuário* quanto do bloco de *Interface de hardware*.

No terceiro bloco, onde estão as classes que independem do modo de operação, tem-se:

1. Liga com arquivo: esta classe faz a leitura e escrita de dados em arquivos que são armazenados no HD do computador;
2. Gera arquivo para Matlab: aplica um sinal em forma de degrau e lê a correspondente resposta no tempo, gerando um arquivo com esses dados;
3. Procura zero: posiciona a plataforma em uma das três chaves fim de curso existentes.

6.2.1.3 Interface de hardware

Neste bloco tem-se as funções que realizam a interface entre dois equipamentos de hardware, podendo ser dois microcomputadores ou um microcomputador e hardware da célula de manufatura. Tem-se:

1. Interface de rede: utiliza tecnologia RMI para controlar o acesso da aplicação JNCW à aplicação JNCS, quando se deseja ler ou alterar a palavra de controle de 32 bits;
2. Controla acesso: utiliza tecnologia RMI para controlar o acesso da aplicação JNCS à JNCS, quando o servidor deseja executar um programa ou posicionar a plataforma em uma chave fim de curso;
3. Interface de hardware: através de uma biblioteca dinâmica (dll), realiza a interface do microcomputador com o hardware da célula de manufatura, gerando sinais para o PWM, lendo a contagem do encoder e os sinais digitais das chaves fim de curso.

6.2.2 JNCS (Java Numerical Control Server)

Através desta aplicação, pode-se basicamente movimentar a plataforma até uma posição desejada e executar programas contendo a linguagem G. Para um maior detalhamento, veja *Anexo D.b.*Java Numerical Control Server.

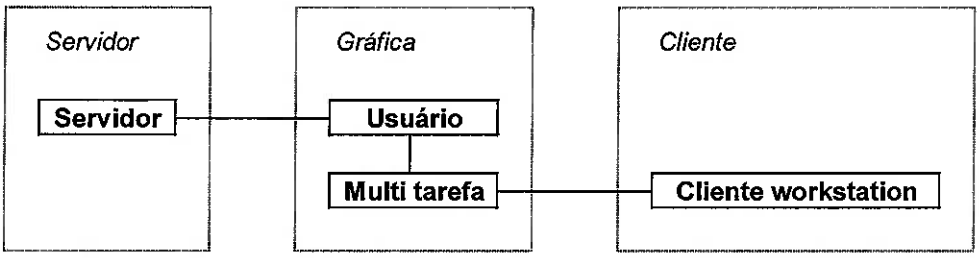


Figura 5 - Estrutura de classes do JNCS

6.2.2.1 Gráfica

A principal função do bloco denominado Gráfica é realizar a interface com o usuário. Tem-se duas classes com as funções:

1. Usuário: responsável pela interface com o usuário propriamente dita, ou seja, é nela que a tela está estruturada e é onde se pode selecionar arquivo, executá-lo ou alterar a palavra de controle;
2. Multi tarefa: realiza o multiprocessamento das opções da classe Usuário. Ou seja, para que seja possível alterar a palavra de controle quando se estiver executando um programa, esta classe cria mais uma "thread" e executa o comando sem interferir no que já estava sendo executado.

6.2.2.2 Servidor

Este bloco, composto somente pela classe Servidor, que utiliza a tecnologia RMI, realiza o controle do acesso da aplicação JNCW à palavra de controle de 32 bits, quando essa quiser ler ou alterar o valor da mesma.

6.2.2.3 Cliente

Este bloco, composto somente pela classe Cliente Workstation, que utiliza a tecnologia RMI, realiza o controle do acesso da aplicação JNCS à aplicação JNCW, quando se deseja executar um programa na wosrkstation.

6.3 Projeto do compilador

O compilador traduz as funções de programação escritas em um arquivo texto para uma lista de comandos, de modo que o software possa interpretá-las e executá-las.

A lista de comandos é composta por uma estrutura que contém todos os campos relativos às funções de programação, podendo ser representada da seguinte maneira:

Comando
.n
.g
.m
.x
.f
.b
.v
.s

Figura 6 – Estrutura de dados para armazenar um comando

A compilação das funções segue uma seqüência lógica, que pode ser resumidamente descrita como:

- Verifica se há um arquivo selecionado e, caso não haja, pede para que isto seja feito.
- Lê linha a linha do arquivo e, para cada uma, verifica a sintaxe. Caso esta esteja correta, a lista de comandos é atualizada; caso contrário, a mensagem de erro correspondente é adicionada à lista de erros, que é mostrada ao operador ao final da compilação.

Seguindo a lógica das funções em linguagem G especificadas, pode-se representar, de forma geral, uma seqüência lógica em forma de grafo a ser seguida pelo compilador para transformar a linha lida em lista de comandos.

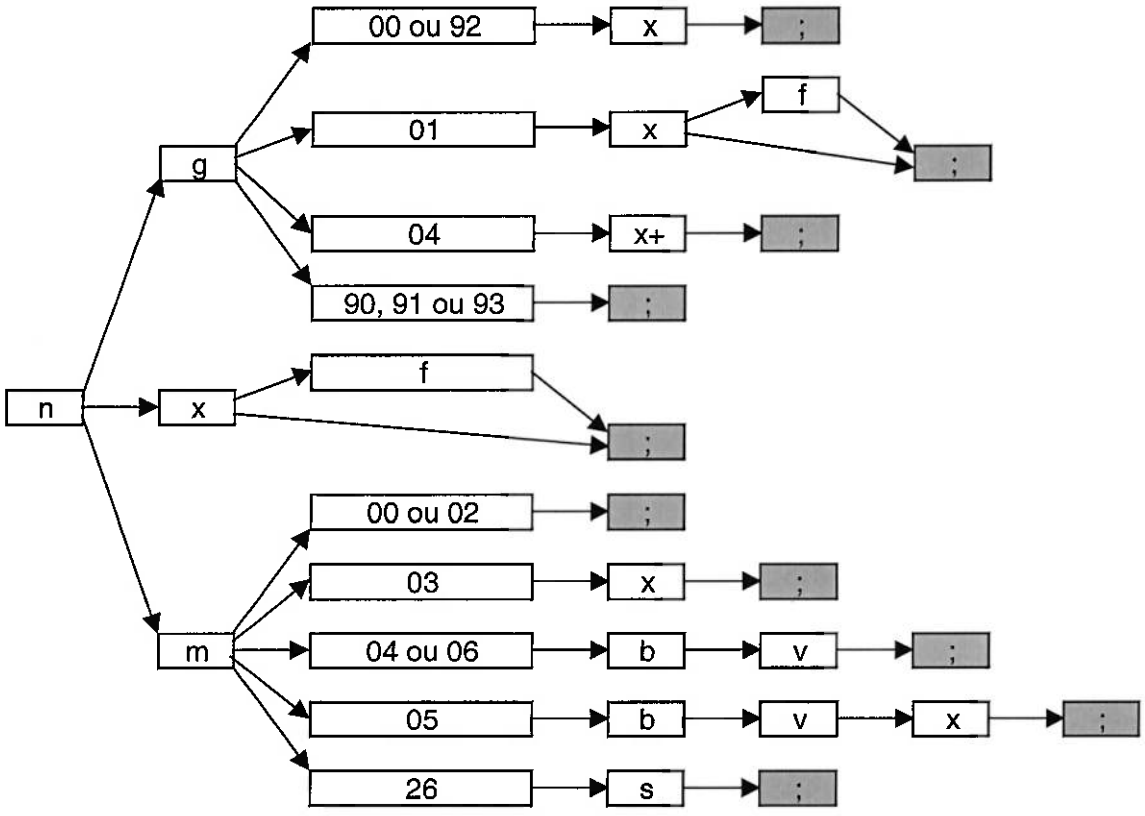


Figura 7 - Grafo representando os possíveis caminhos de compilação

Assim, seguindo a lógica proposta acima, qualquer função que não siga um dos possíveis caminhos será interpretado como *linha com erro*, e o erro correspondente será arquivado em uma lista, para posterior apresentação ao usuário.

Algumas particularidades foram adotadas no projeto, de modo a facilitar a posterior execução dos comandos:

- Como a linguagem G definida possui funções modais, o compilador verifica quais foram os comandos com esta característica programados e os coloca automaticamente na estrutura de comando definida, permitindo ao interpretador olhar somente para o elemento em questão na hora da execução.
- Em relação às coordenadas de posicionamento, mesmo que sejam programadas coordenadas incrementais ou que o zero do trilho seja deslocado, o compilador calcula a posição absoluta em relação ao centro do trilho, sendo este o valor colocado na estrutura.
- Caso, no início do arquivo texto contendo as funções de programação, não estejam especificados o modo de posicionamento, o sistema de coordenadas ou sua origem, o compilador assume:

  Modo de posicionamento na máxima velocidade permitida (g00);

  Sistema de coordenadas absoluta (g90);

  Origem do sistema de coordenadas no centro do trilho (g93).
- Caso o comando "x" seja usado sem outra função na linha, somente com a função número de seqüência "n", que é obrigatória, o compilador assume

que se deseja posicionar a plataforma até a posição especificada em "x", ou seja, que a função g00 ou g01, conforme a que esteja pré-programada, está implícita na linha.

Com base nas definições e no grafo propostos acima, foi elaborado o diagrama de Nassi-Schneiderman para o compilador de código G, que encontra-se no *Anexo B.Diagramas de Nassi-Schneiderman do compilador de linguagem G*

6.4 Projeto do interpretador

O interpretador verifica se o arquivo texto contendo a linguagem G está correto, ou seja, chama a rotina do compilador. Caso esteja, executa os comandos até encontrar o que significa final de arquivo (m02), apresentando um status on line.

Como o compilador gera todas as posições em coordenadas absolutas, o interpretador não precisa se preocupar com as funções g90, g91, g92 e g93, que dizem respeito ao sistema de coordenadas e sua origem, sendo utilizadas apenas para atualizar o status apresentado ao usuário.

Com base nas especificações propostas acima, foi elaborado o diagrama de Nassi-Shneiderman para o interpretador de código G, que encontra-se no *Anexo C.Diagrama de Nassi-Schneiderman do interpretador de linguagem G.*

6.5 Projeto do controlador PID

6.5.1 Lógica

Tem-se a seguinte lógica para a malha de controle do PID:

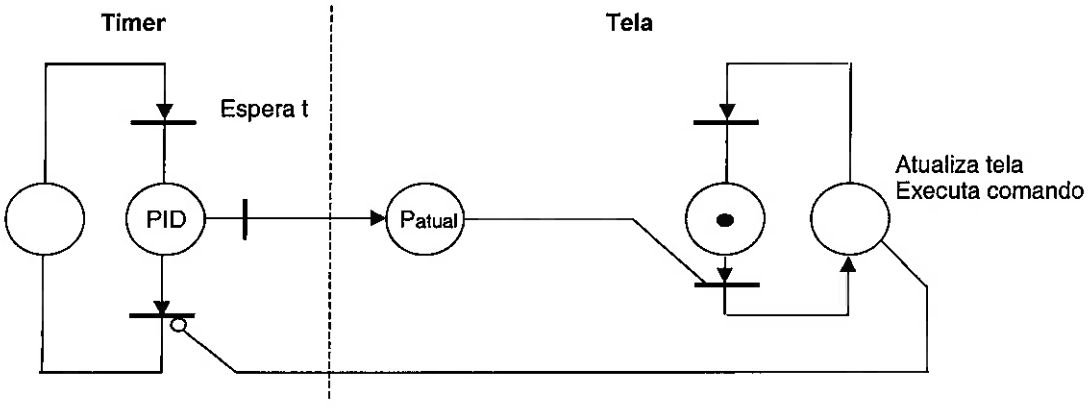
```
public int pid(float Kd, float Ki, float Kp, int T, float porc, long erro, long erro_ant, long soma_erro) {
        //Calcula o valor da velocidade a ser gerada pelo PID
        int Vzero = 0;
        int Vmax = 100;
        int Vmin = -100;

        Double V = new Double( (Vzero + Kp*erro + (Kd/T)*1000*(erro-erro_ant) + Ki*soma_erro) + 0.5);

        int V2 = V.intValue();
        if ( V2 >= (porc*(Vmax - Vzero) + Vzero) )
        {
                Double V3 = new Double(porc * (Vmax - Vzero) + 0.5);
                V2 = V3.intValue() + Vzero;
        }
        if ( V2 <= (Vzero - porc*(Vzero-Vmin)) )
        {
                Double V3 = new Double(porc * (Vzero - Vmin) + 0.5);
                V2 = Vzero - V3.intValue();
        }

        return V2;
}
```

Malha 1- Malha de controle PID

Acima está a implementação da equação de diferenças do controlador PID. Sua entrada é formada pelas constantes derivativa, integral e proporcional, pelo período de amostragem, pela porcentagem da velocidade máxima desejada para o posicionamento e pelos erros atual, anterior e soma dos erros.

De posse desses valores, calcula-se a tensão que deve ser enviada ao acionamento do motor. Caso esta tensão seja suficiente para gerar uma velocidade maior que a programada, o valor da tensão enviado para o motor é limitado ao valor necessário para gerar a velocidade desejada.

A verificação do erro, isto é, se o loop PID deve ser repetido ou não, é feita pela classe que controla o posicionamento, através de um timer, que permanece habilitado enquanto a posição não for satisfatória. Este gerenciamento pode ser representado por uma Rede de Petri:



Malha 2 - Rede de Petri do controlador PID

Nesta rede, o timer é composto por uma transição temporizada, que representa o período de amostragem. Após este período, é feito o cálculo do PID e a próxima transição faz com que novo cálculo seja feito somente se esta estiver habilitada, o que é feito pelo interpretador de código G, caso a função seja de posicionamento ou caso o set-point ainda não tenha sido atingido.

6.5.2 Projeto

Projetou-se o controlador PID utilizando uma técnica apresentada em [3], baseada na observação da resposta da planta a uma entrada degrau.

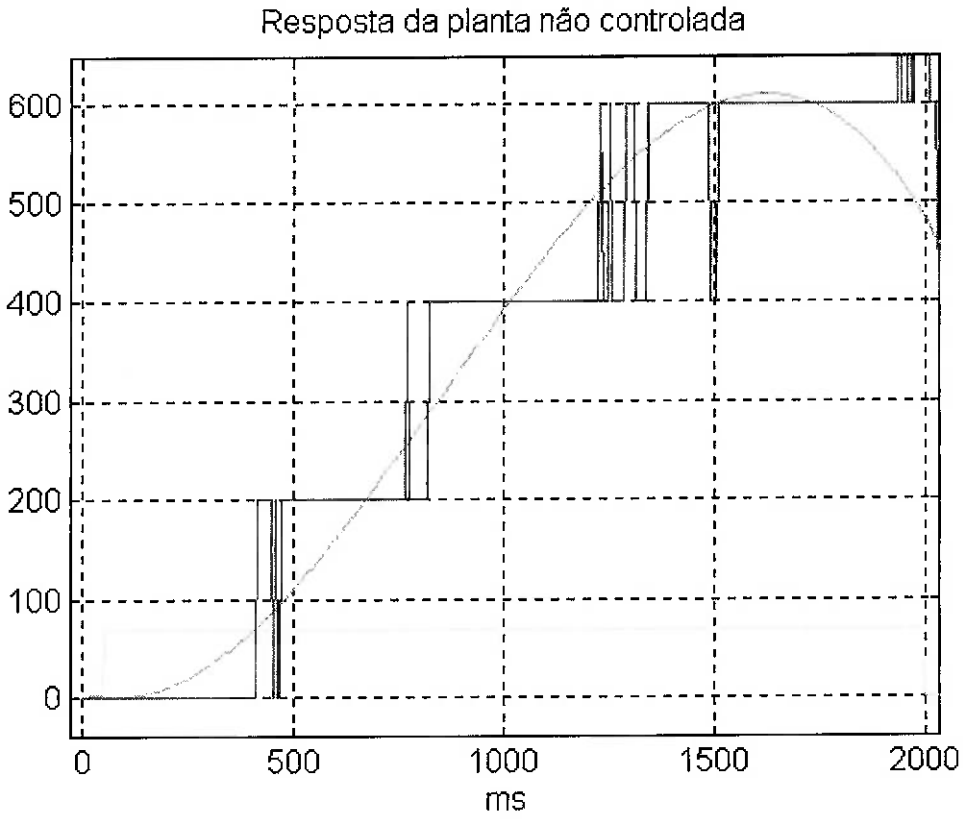


Figura 8 - Resposta da planta a entrada degrau

Com base na resposta acima, pode-se projetar graficamente os ganhos proporcional, integral e derivativo:

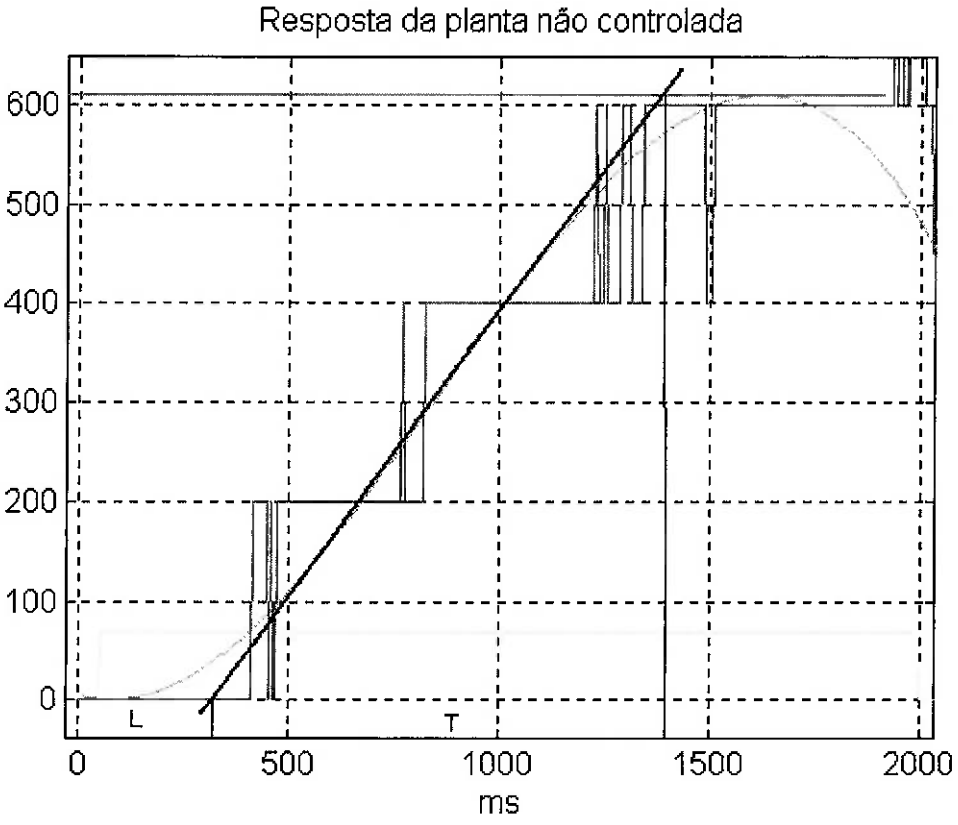


Figura 9 - Projeto do controlador PID

Tem-se:

- L = 0.33 s
- T = 1.04 s

Logo, conforme recomendado:

- Ganho proporcional: Kp = 1.2*L / T = 3.78
- Tempo integral: Ti = 2*L = 2.08
- Tempo derivativo: Td = 0.5*L = 0.52

Assim, pode-se obter os ganhos reais que serão utilizados pelo controlador:

- Proporcional: Kp = 3.78
- Integral: Ki = Kp/Ti = 1.82
- Derivativo = Kd = Kp*Td = 1.97

É importante ressaltar que o projeto do controlador foi feito com a atual configuração da plataforma (com o robô ASEA sem carregar peça). Caso isso mude, será necessário reprojetar o controlador.

7 Cronograma de atividades

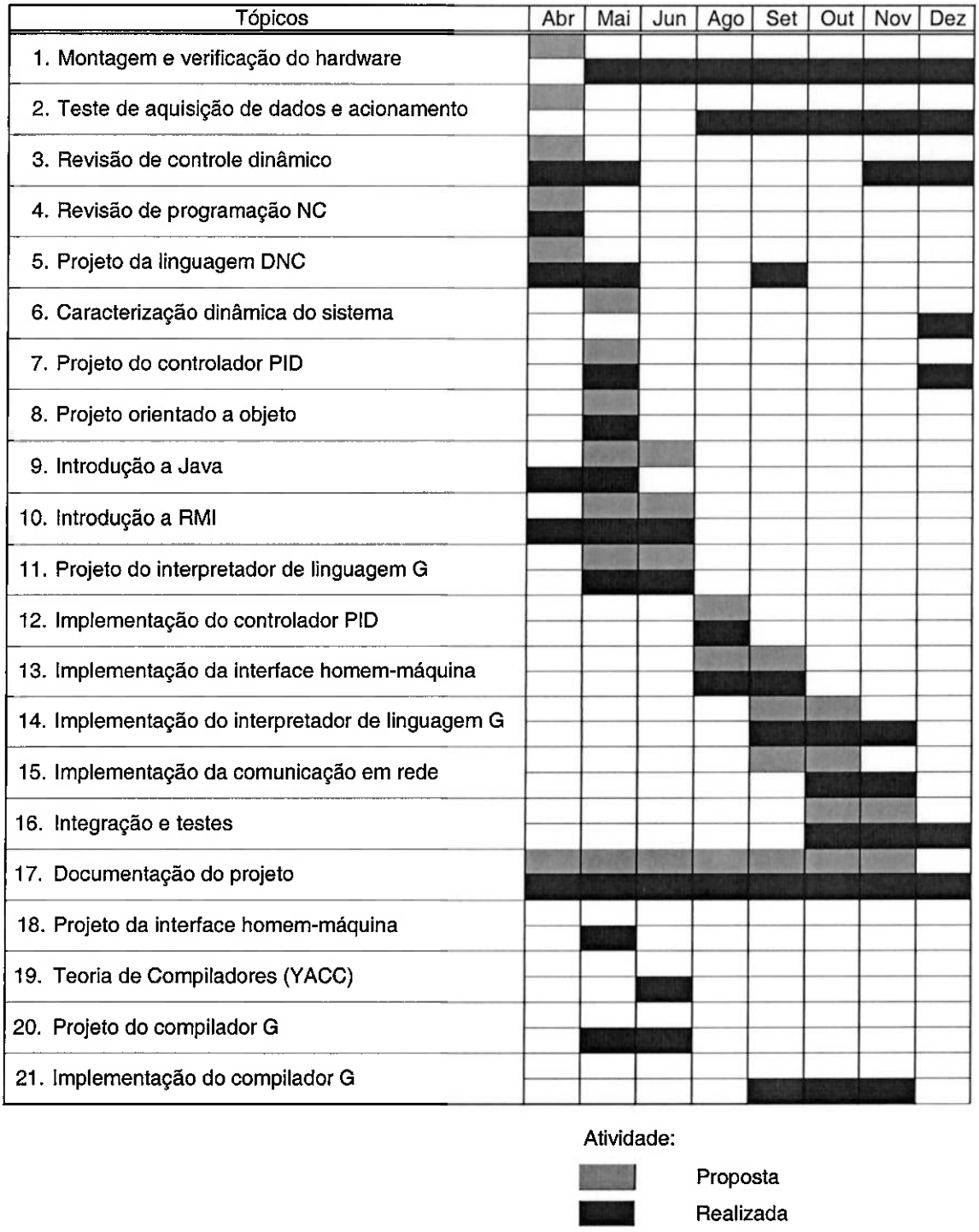


Figura 10 - Atividades propostas e realizadas

8 Conclusões

Pelo cronograma apresentado, verifica-se que houve atraso na parte relativa ao hardware da solução. Isso ocorreu porque o encoder que seria utilizado na solução havia sumido, as chaves de fim-de-curso ópticas estavam quebradas e não havia placa de interface entre o microcomputador e o hardware da célula de manufatura. Perdeu-se tempo tentando encontrar o encoder e a placa, mas por fim, para solucionar esses problemas, optou-se por emprestar um encoder de outra aplicação, projetar e montar as chaves fim-de-curso e comprar a placa de IO do microcomputador no mercado.

Quanto à parte de software, a aplicação em Java que realiza a interface homem-máquina e o controle de posicionamento da plataforma foi desenvolvida dentro do tempo projetado, sendo possível operá-la tanto em modo manual quanto automático, este ultimo tanto localmente quanto pela rede, utilizando um programa contendo linguagem G e a aplicação Java Numerical Control Server (JNCS).

Para realizar a interface entre a aplicação em Java e o hardware da célula de manufatura, optou-se pela elaboração de uma dll em linguagem C++, uma vez que o conceito multiplataforma do Java faz com que essa linguagem não tenha ferramentas para realizá-la diretamente. Com isso, criou-se uma desvantagem por perder o conceito multiplataforma da linguagem, que foi solucionado com a elaboração de duas aplicações:

- JNCW (Java Numerical Control Workstation), a ser executada no microcomputador que contem a placa de IO e a ddl, realizando a interface com a célula. Esse microcomputador deve estar operando com o sistema Windows 95, NT 4.0 ou compatível;
- JNCS (Java Numerical Control Server), a ser executada em qualquer microcomputador que esteja em rede com o primeiro, podendo operar em qualquer sistema operacional, uma vez que trata-se de uma aplicação Java pura.

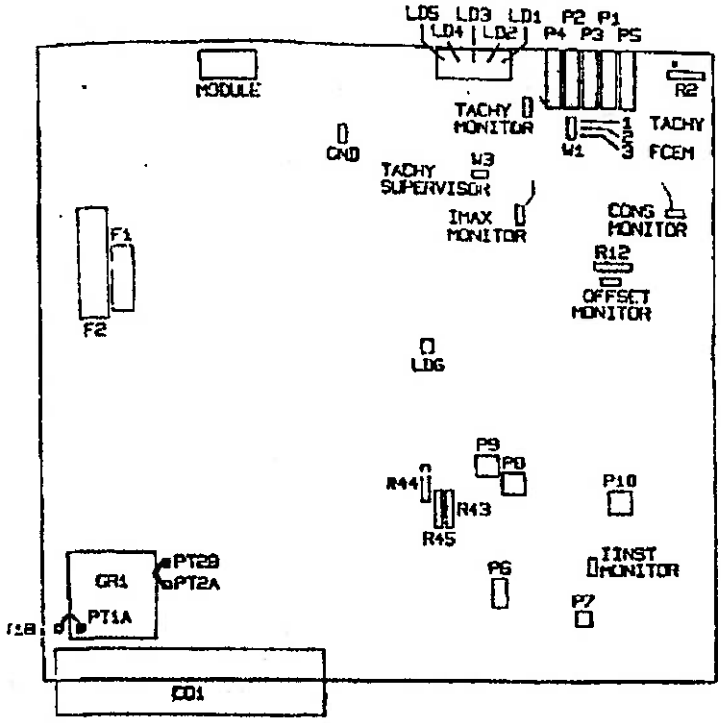
Com o sistema totalmente integrado, foi possível coletar dados sobre o comportamento dinâmico da planta para realizar o projeto do controlador PID, que foi posteriormente ajustado à planta por meio de testes práticos. O posicionamento da plataforma, no entanto, não se dá de forma totalmente precisa, ocorrendo erros da ordem de 50 mm devido a:

- Ruídos descontínuos no sinal do encoder, que prejudicam a leitura. Às vezes a leitura do encoder pela aplicação indicava que a plataforma estava se movimentando, quando na realidade estava parada.
- Atrito entre a guia da plataforma e o trilho, que faz com que o movimento não se dê de forma contínua quando a velocidade é baixa, o que ocorre quando a plataforma está começando a se movimentar ou atingindo a posição desejada quando controlada pelo PID.
- Folga no cabo que liga a plataforma ao eixo do motor, o que faz com que a plataforma não atinga uma posição precisa. Tentou-se esticar o cabo ao máximo, porém não foi possível eliminar totalmente a folga;

- Acoplamento entre o encoder e o eixo do motor, feito através de um pequeno eixo engastado na ponta do eixo do motor e uma borracha presa com arame e fita isolante para unir o encoder ao eixo menor. Como esse acoplamento possui baixa rigidez, existem erros no posicionamento do encoder quando ocorrem módulos de aceleração elevados (na partida e parada total da plataforma).

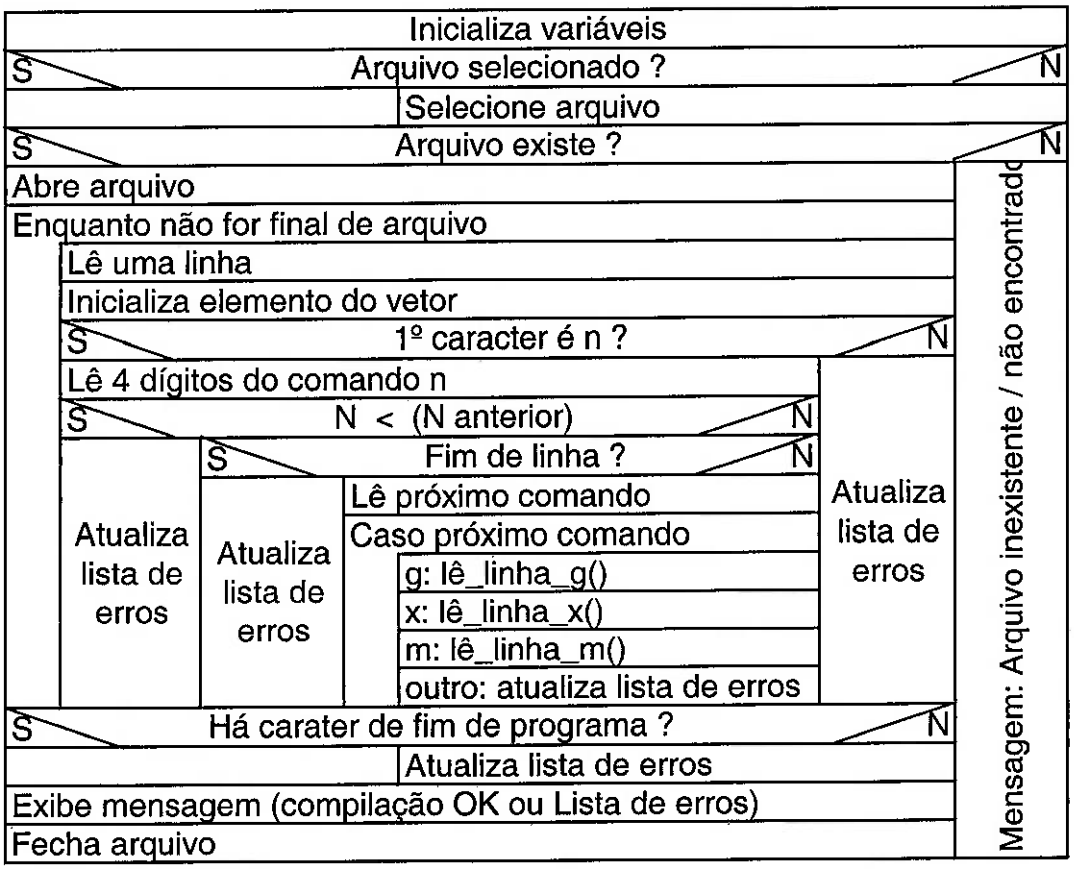
Anexos

A. Esquema da placa de acionamento do servo-motor

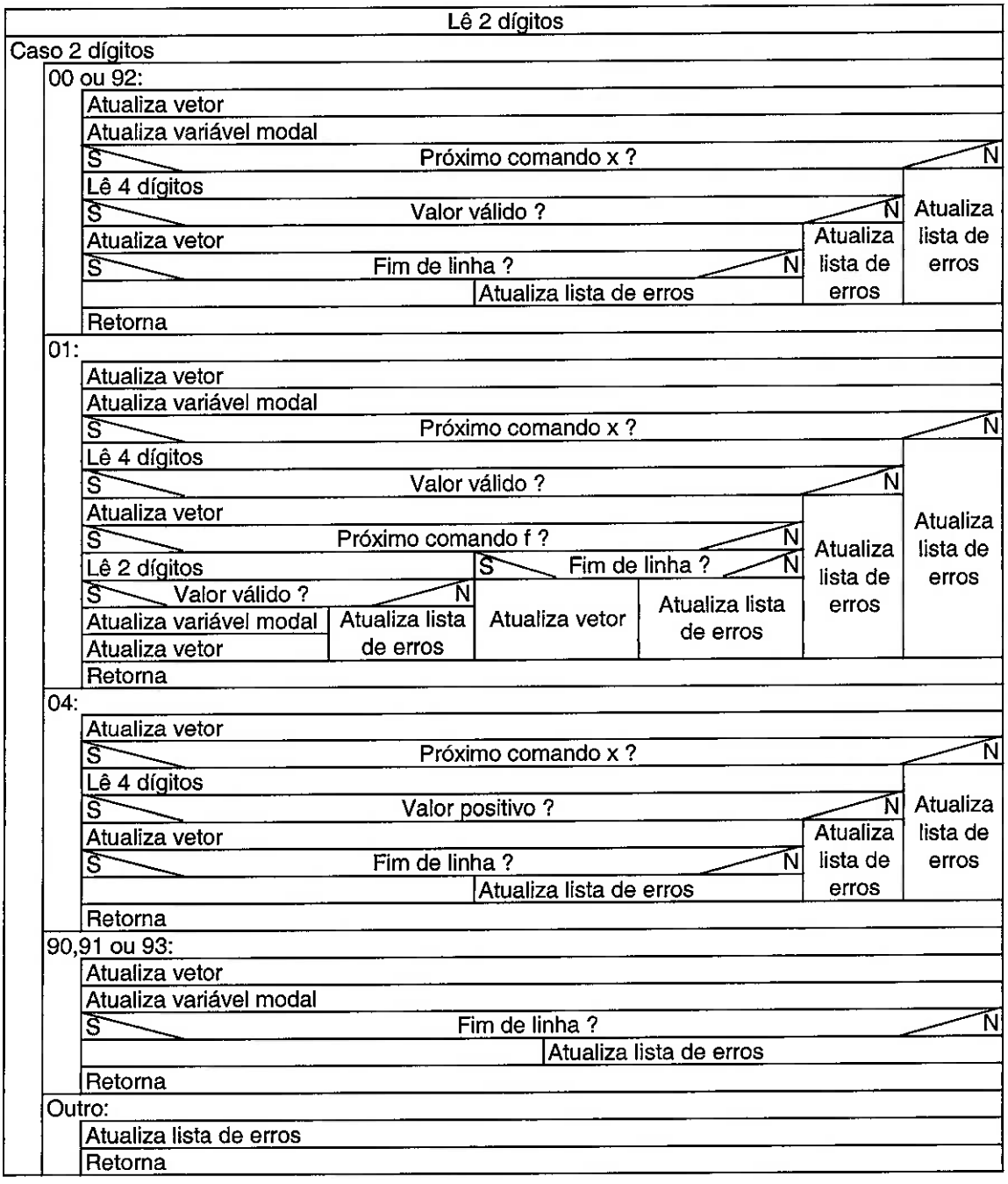


B. Diagramas de Nassi-Schneiderman do compilador de linguagem G

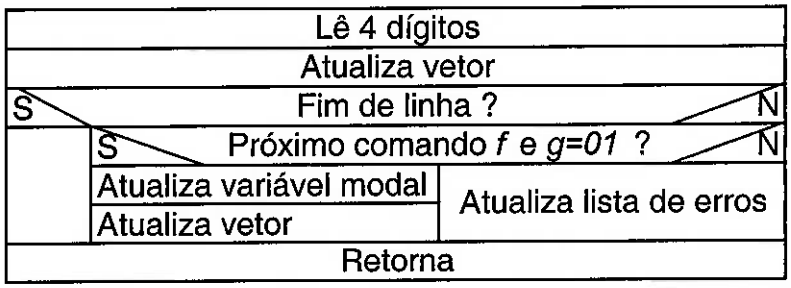
a. Principal



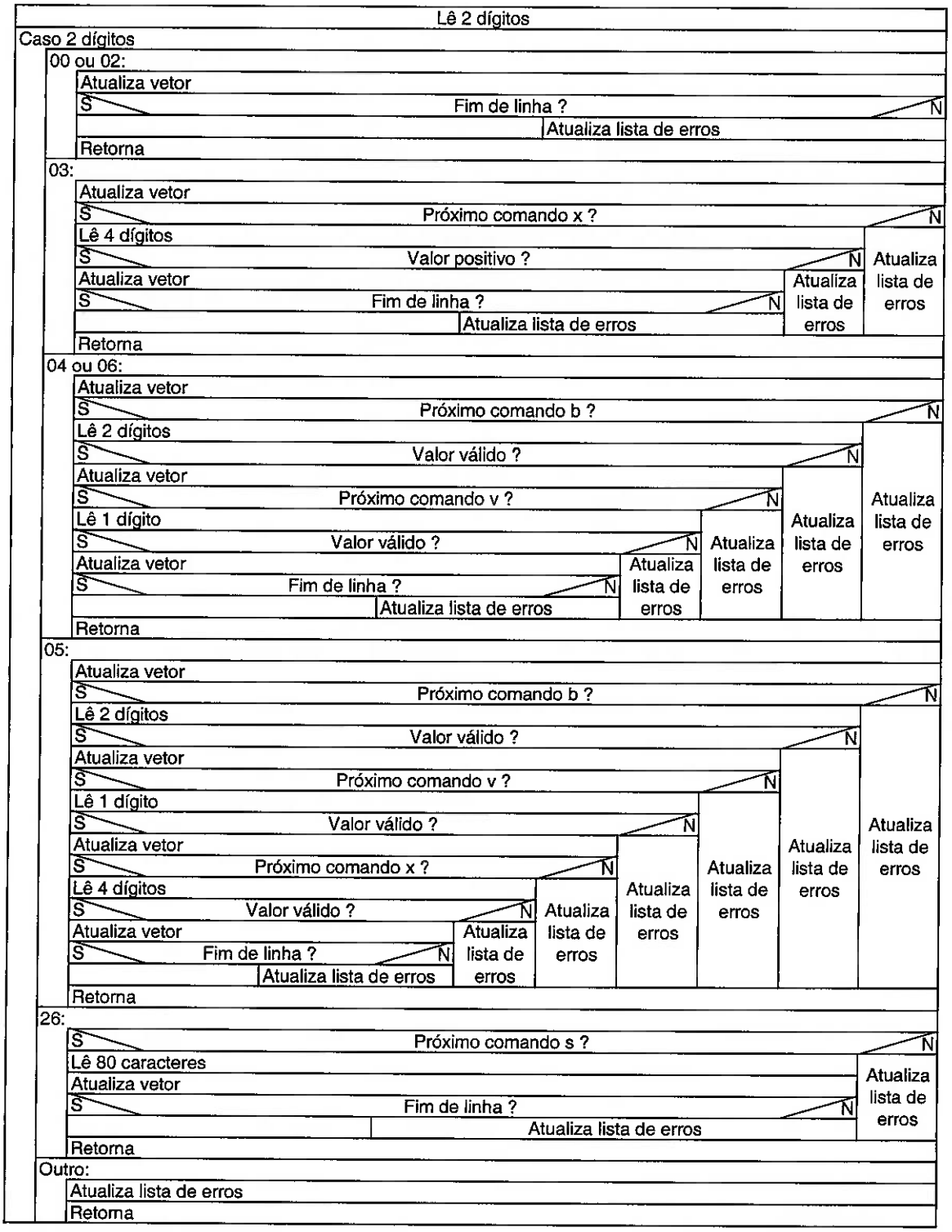
b. Função Lê_linha_g



c. Função Lê_linha_x



d. Função Lê_linha_m

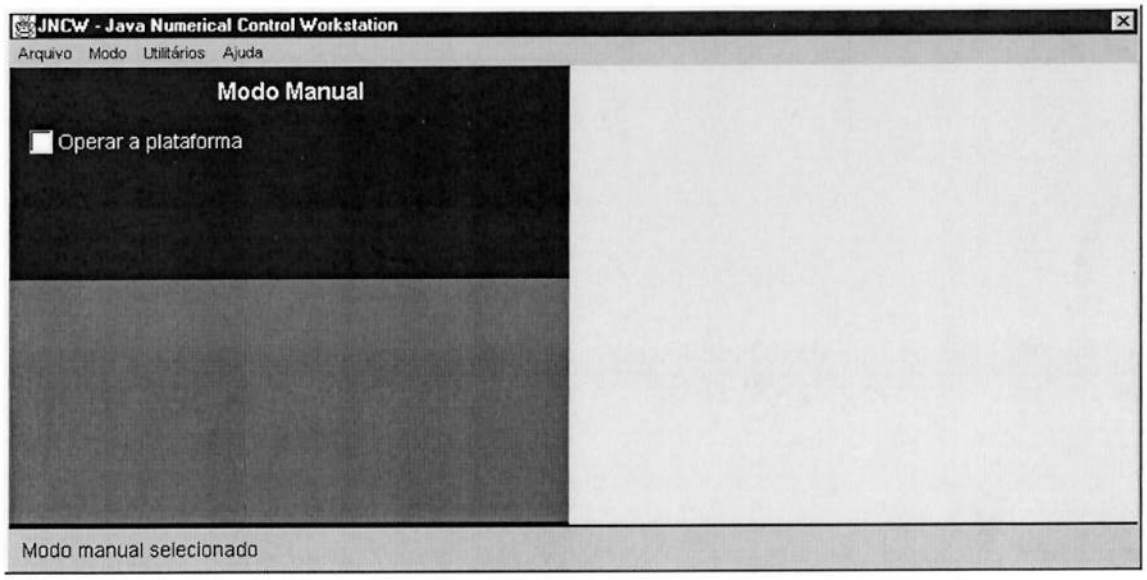


C. Diagrama de Nassi-Schneiderman do interpretador de linguagem G

- Inicializa variáveis
- Compila programa
- Houve erro de programa ? (S / N)
  - S: Indica que houve erro de compilação
  - N:
    - Enquanto não for fim de programa
      - Caso g
        - 00 ou 01:
          - Atualiza status na tela
          - Posiciona até posição x com f% da velocidade máxima
        - 04:
          - Atualiza status na tela
          - Enquanto não deu tempo x
            - Calcula tempo
        - 90, 91, 92 ou 93:
          - Atualiza status na tela
      - Caso m
        - 00:
          - Atualiza status na tela
          - Enquanto não pressiona tecla continua
            - Aguarda tecla continua
        - 03:
          - Não achou linha x e não é fim de programa
            - Procura linha
          - É fim de programa ? (S / N)
            - S: Sai do loop; Atualiza status de erro
            - N: Atualiza status na tela; Desvia programa
        - 04:
          - Faça
            - Leia palavra de interface
          - Enquanto valor(bit b) <> v
        - 05:
          - Leia palavra de interface
          - Valor(bit b) = v ? (S / N)
            - S:
              - Não achou linha x e não é fim de programa
                - Procura linha
              - É fim de programa ? (S / N)
                - S: Sai do loop; Atualiza status de erro
                - N: Atualiza status na tela; Desvia programa
            - N:
        - 06:
          - Leia palavra de interface
          - Faça valor(bit b) = v
          - Envie palavra de interface
        - 26:
          - Mostra mensagem s para usuário
      - Próximo comando
      - Atualiza status final

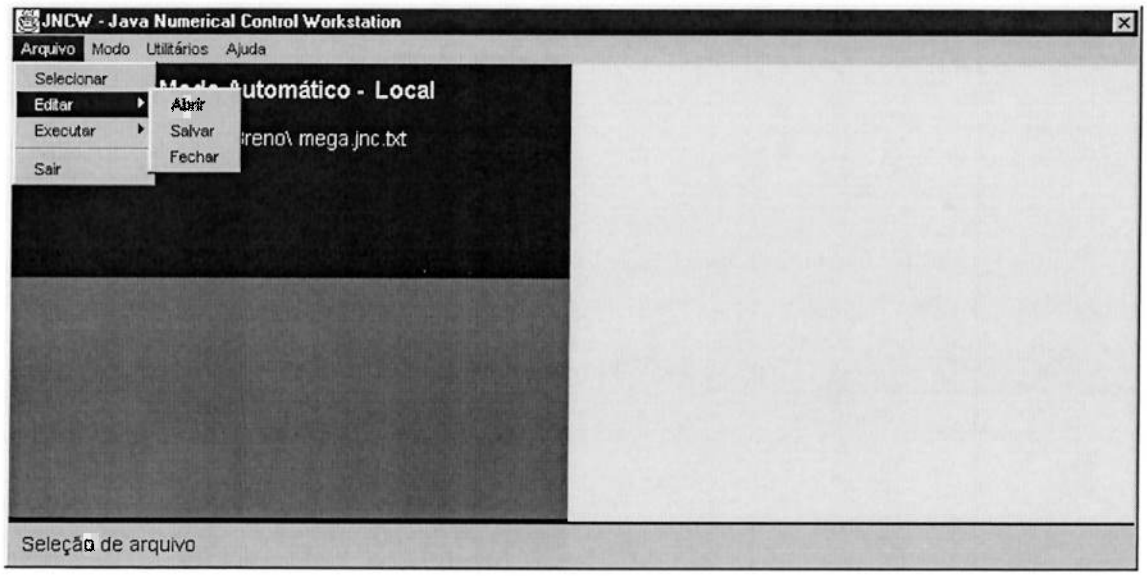
D. Telas do aplicativo desenvolvido

a. Java Numerical Control Workstation

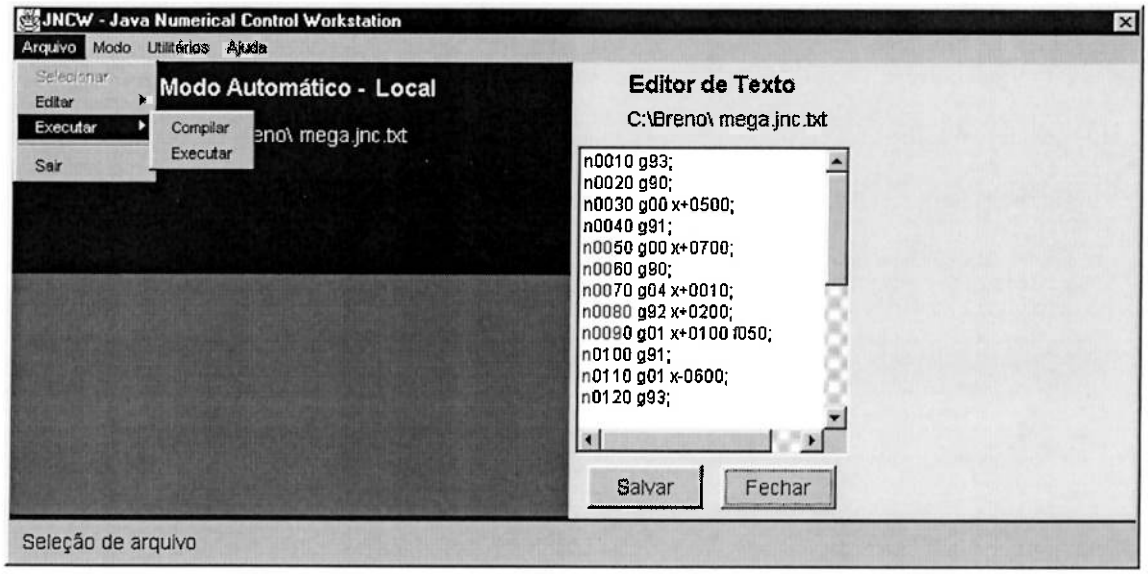


Tela 1 - Principal do JNCW

Ao se executar a aplicação, entra-se na tela principal, onde se tem os menus, uma região na parte superior esquerda, onde de pode ver em qual modo se está operando, uma na inferior esquerda, onde será mostrado a posição, velocidade e aceleração da plataforma quando esta estiver em operação e uma na direita, onde se tem o espaço para um editor de texto e onde se mostra os erros de compilação do programa selecionado.



Tela 2 - Menu arquivo do JNCW, primeira parte

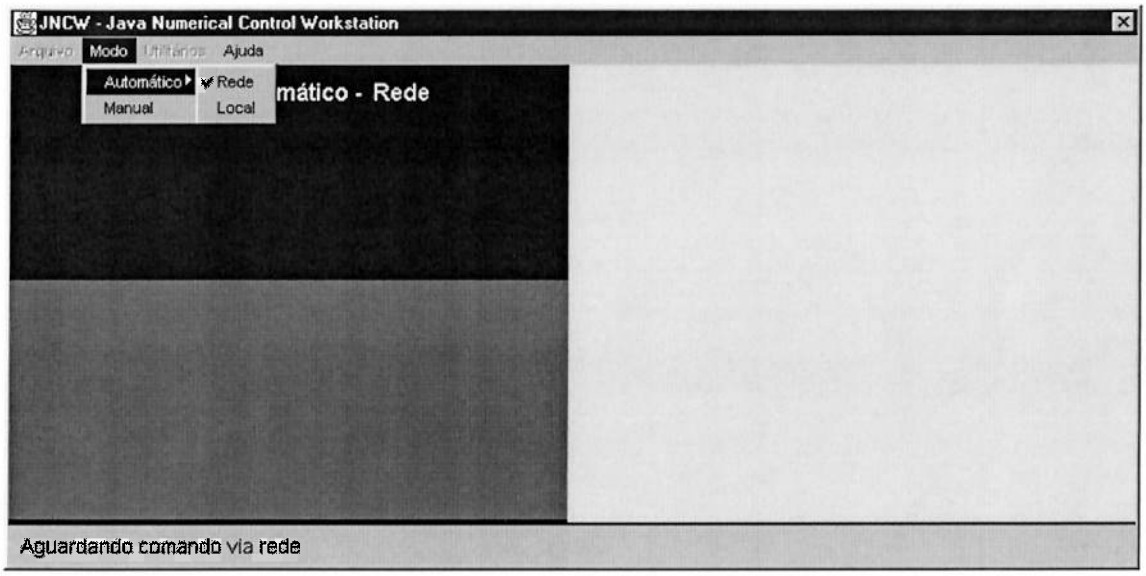


Tela 3 Menu arquivo do JNCW, segunda parte

No menu Arquivo, pode-se selecionar um arquivo contendo linguagem G já existente ou criar um novo.

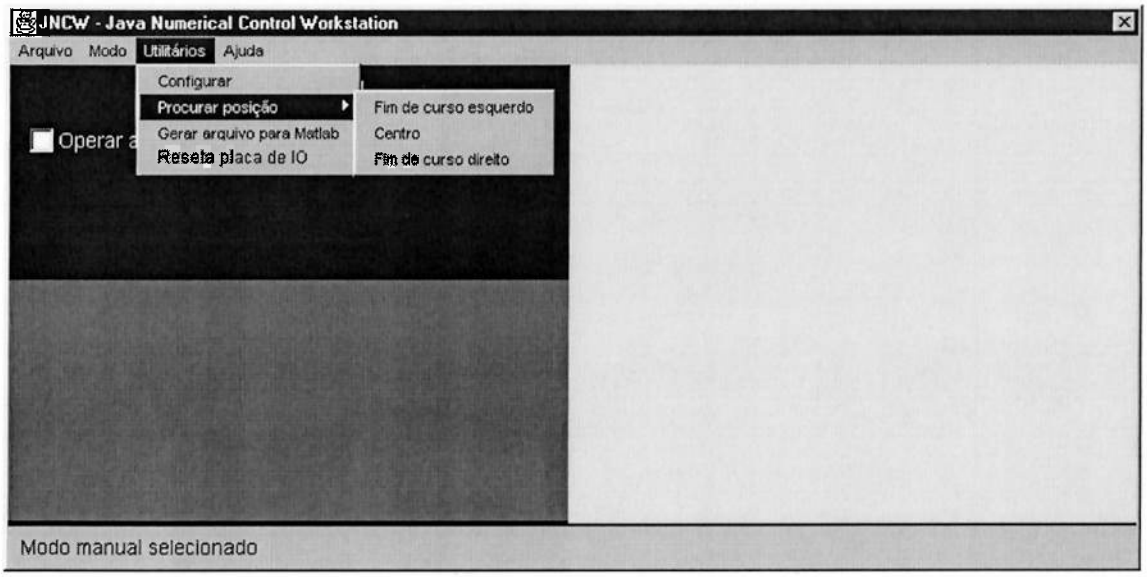
Pelo submenu Editar, pode-se Abrir, Salvar ou Fechar o arquivo selecionado. Ao se abrir o arquivo, o mesmo aparece no editor de texto, conforme pode ser visto na tela 3.

Pelo submenu Executar, pode-se Compilar o arquivo selecionado ou Executá-lo. Esse submenu somente estará disponível quando a opção de Rede Local, dentro do menu Modo, estiver selecionada.



Tela 4 - Menu modo do JNCW

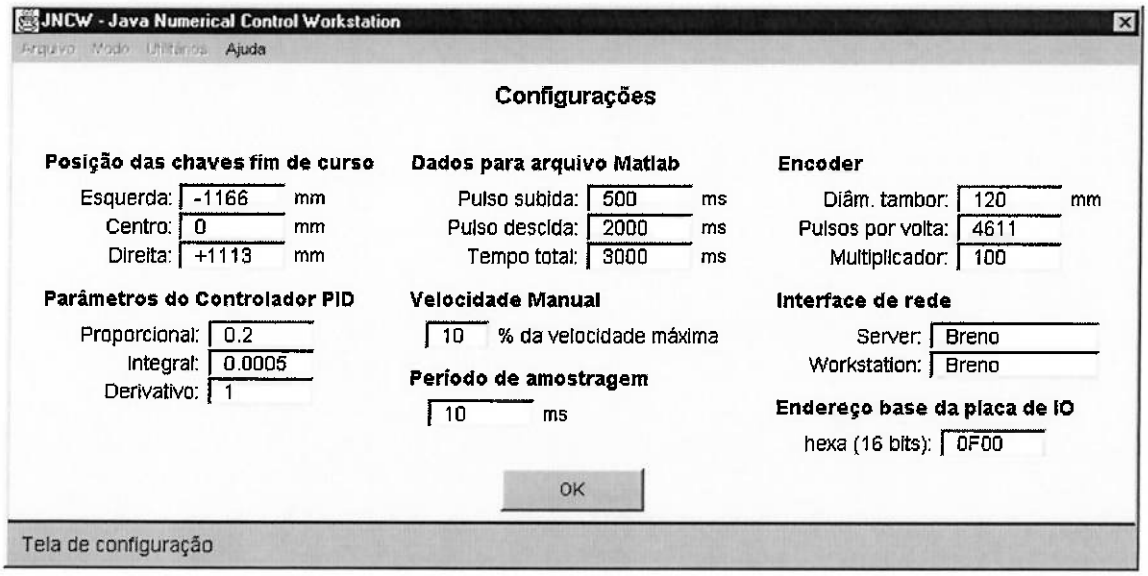
Pelo menu Modo, pode-se selecionar o modo de operação da aplicação: em modo manual, opera-se diretamente a plataforma; em automático local, opera-se através de um arquivo contendo linguagem G; em modo automático de rede, disponibiliza-se a aplicação para ser operada via rede pela aplicação JNCS.



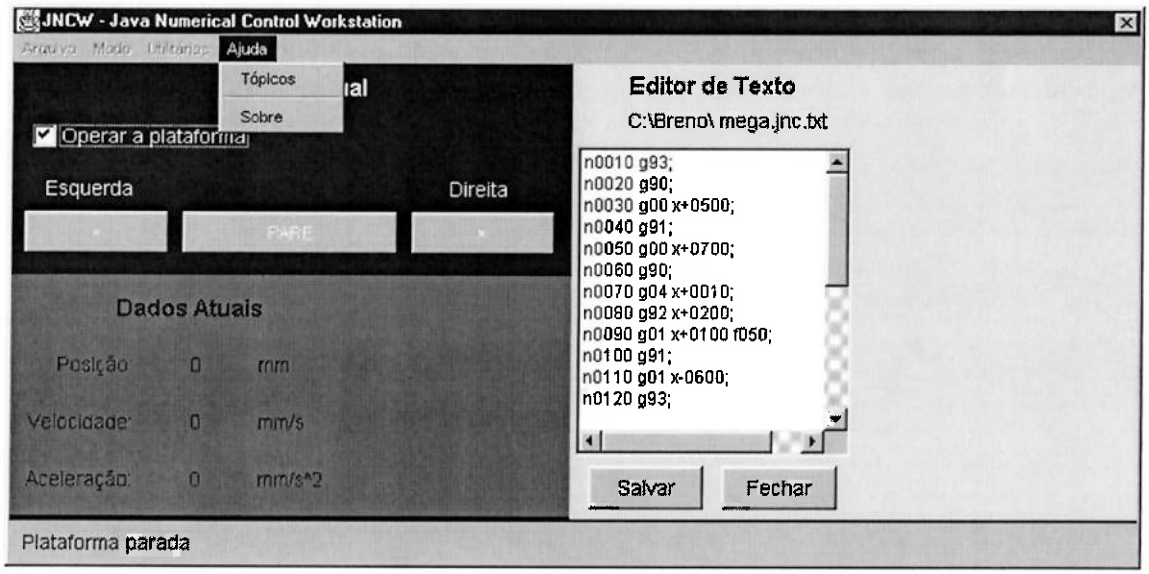
Tela 5 - Menu utilitários do JNCW

Pelo menu Utilitários, pode-se Configurar os parâmetros de operação da plataforma, posicionar a plataforma em uma das três chaves fim de curso, gerar um arquivo para o Matlab contendo o sinal em forma de degrau e a resposta da planta, para se projetar o controlador PID, e resetar a placa de IO do microcomputador com o hardware.

Ao se entrar no submenu Configurar, obtém-se a tela abaixo, onde pode-se parametrizar diversas variáveis da aplicação:



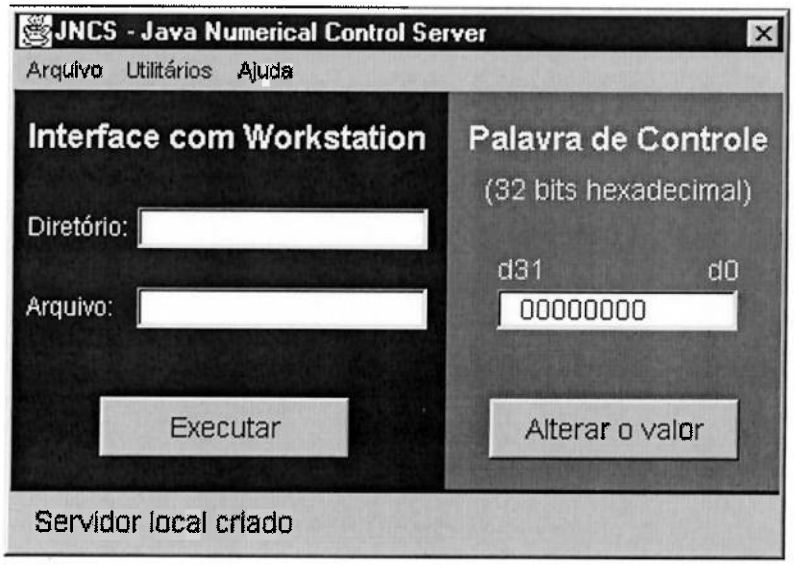
Tela 6 - Configurações do JNCW



Tela 7 - Menu ajuda do JNCW

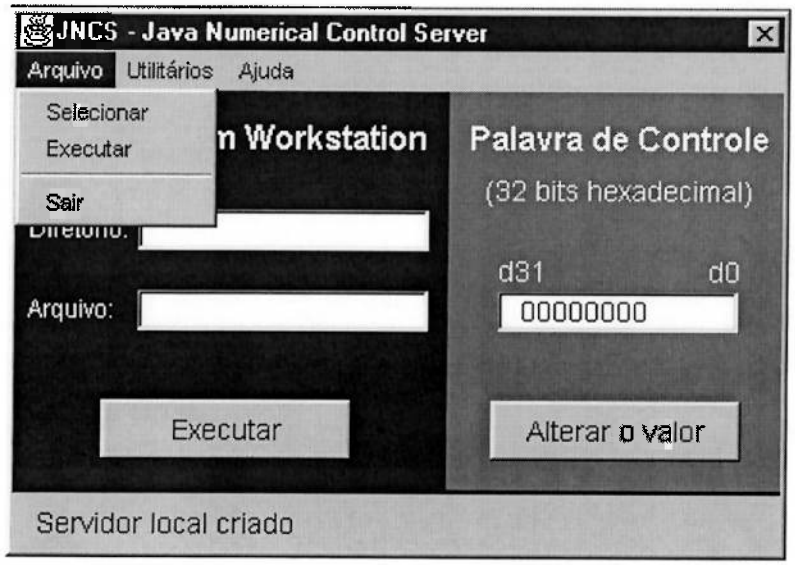
Pelo menu Ajuda, pode-se acessar um arquivo de ajuda contendo os tópicos relativos a como lidar com a aplicação e uma tela de Sobre.

b. Java Numerical Control Server



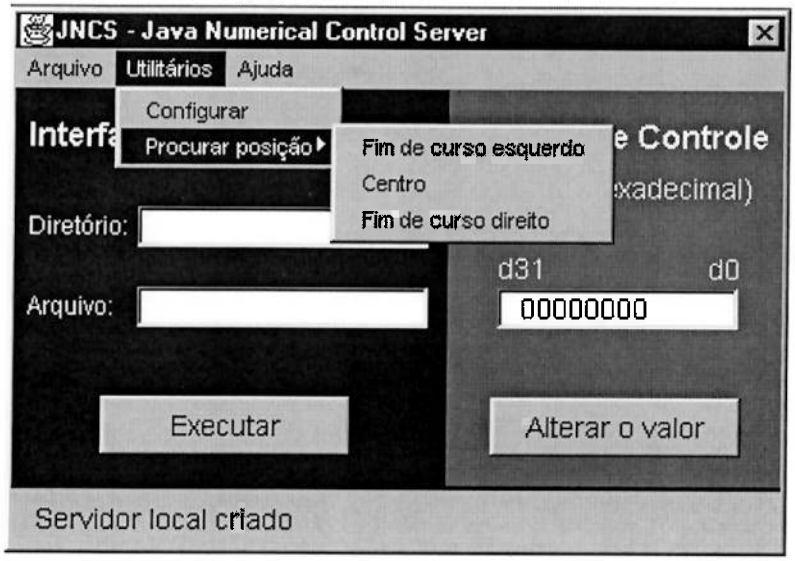
Tela 8 - Principal do JNCS

Ao se executar a aplicação, obtém-se esta tela, onde, na parte esquerda, há indicação do diretório e arquivo que será executado na aplicação JNCW, e, na direita, o valor da palavra de controle de 32 bits e um botão para se alterar seu valor.

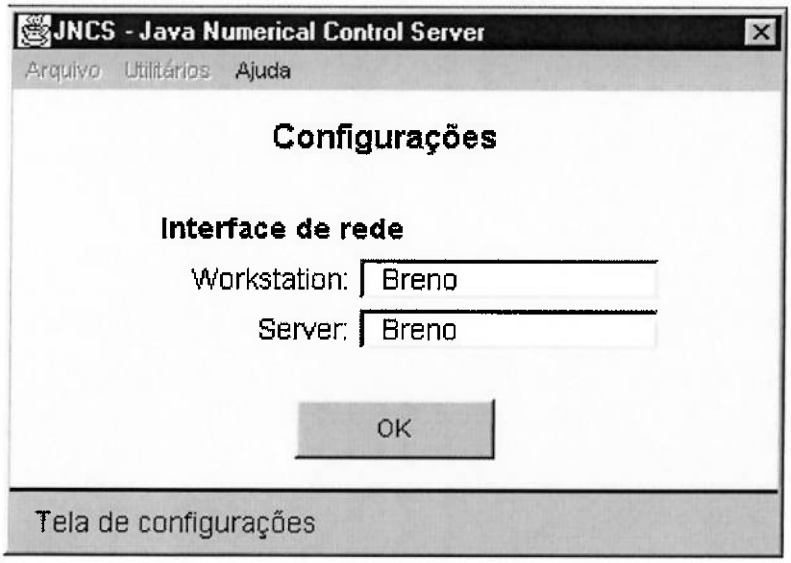


Tela 9 - Menu arquivo do JNCS

Pelo menu Arquivo, pode-se selecionar um arquivo já pronto contendo a linguagem G e mandar executá-lo no JNCW. Não é possível editar programas nesta aplicação.

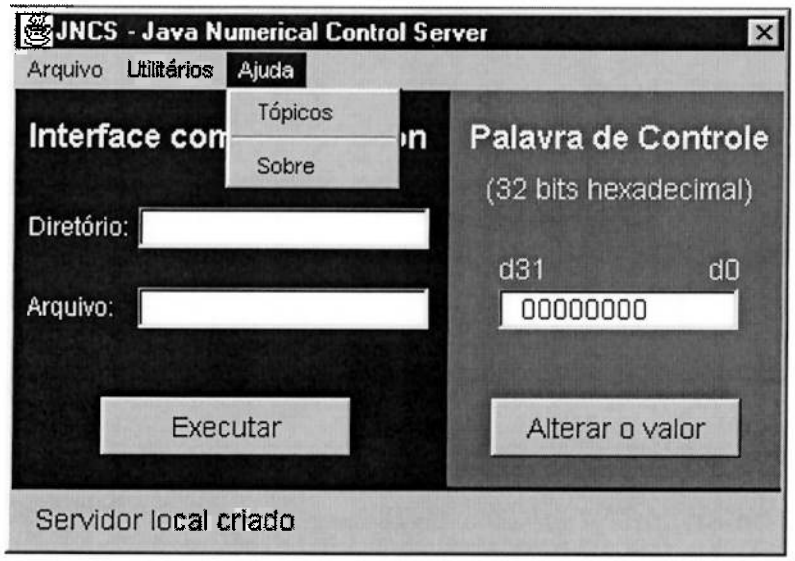


Tela 10 - Menu utilitários do JNCS



Tela 11 - Tela de configurações do JNCS

Pelo menu utilitários, pode-se Configurar os parâmetros da aplicação, obtendo a tela apresentada acima, ou posicionar a plataforma, através da aplicação JNCW, em uma das três chaves fim de curso.



Tela 12 - Menu ajuda do JNCS

Pelo menu Ajuda, pode-se acessar um arquivo de ajuda contendo os tópicos relativos a como lidar com a aplicação e uma tela de Sobre.

E. Disquete de instalação

O disquete de instalação dos aplicativos JNCW e JNCS possui os arquivos:

- JNCW.exe: execute para instalar o JNCW (Java Numerical Control Workstation) e siga as instruções que serão apresentadas pelo arquivo leiame.txt;
- JNCS.exe: execute para instalar o JNCS (Java Numerical Control Server) e siga as instruções que serão apresentadas pelo arquivo leiame.txt;
- JNC.exe: execute para obter o arquivo texto em Microsoft Word 97 deste documento a apresentação feita à banca avaliadora.

Referências bibliográficas

[1]. SITE da *Rational Software Corporation* em www.rational.com

[2]. SITE *Java Technology Home Page* em http://java.sun.com/

[3]. OGATA, K. **Engenharia de controle moderno.** Rio de Janeiro, Prentice-Hall do Brasil, 1982.

[4]. OGATA, K. **Discrete-time control systems.** Second edition New Jersey, Prentice-Hall, 1995.

[5]. MACHADO, A. **Comando numérico aplicado às máquinas-ferramenta.** 4.ed. São Paulo, Editora Ícone, 1990.

[6]. CABRAL, E.L.; MARUYAMA, N.; HORIKAWA, O.; COZMAN, F.G. **Apostila de PMC-503.** s.n.t.

[7]. MANUAL de instalação Transaxe série 700 Servo-amplificadores 705 e 710. s.l., s.ed., 1992.

[8]. MANUAL placa Sensoray Model 421, s.l., s.ed., 1997.

[9]. APOSTILA sobre linguagem de programção java, Impacta s.n.t.

[10]. GROOVER, M.P.; ZIMMERS, E.W., Jr. **CAD/CAM: computer-aided design and manufacturing** New Jersey, Prentice-Hall, 1984.

[11]. GROOVER, M.P. **Automation, production systems, and computer-aided manufacturing.** New Jersey, Prentice-Hall, 1980.

[12]. GROOVER, M.P.; WEISS, M.; NAGEL, R.N.; ODREY, N.G. **Industrial robotics: technology, programming and aplications.** Mexico, McGraw-Hill, 1986.

[13]. BISCUOLA, J.R.; TOLEDO, V.A. **PMC580/581 Comando Numérico para eixo de posicionamento de robô.** São Paulo, s.ed., 1996.

[14]. KUAE, L.K.N.; BONESIO, M.C.M.; VILLELA, M.C.O. **Manual para apresentação de dissertações e teses.** São Paulo, s.ed., 1991.