MARCELO SHIMADA

Desenvolvimento de um sistema de medição de Circularidade de baixo custo e alta precisão

Trabalho de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduado em Engenharia.

Área de concentração:
Engenharia Mecatrônica

Orientador:
Oswaldo Horikawa

São Paulo
1999

SUMÁRIO

Lista de Figuras

Resumo

O trabalho consiste na construção de um sistema de medição de circularidade e com o uso deste, estudar a aplicação de métodos avançados de medição de circularidade. O sistema incluí: uma mesa giratória que emprega mancais de rolamento, na qual o corpo de prova é fixado e rotacionado, sensores não contactantes que medem os erros de circularidade do objeto, sensores ópticos para medir a posição angular do objeto e um microcomputador que, com o auxílio de um placa de conversão Analógico/Digital, coleta e processa os sinais gerados pelos sensores não contactantes. O programa para aquisição de dados possui as rotinas de tratamento e apresentação dos dados em forma de gráficos lineares e polares, rotinas para eliminação das componentes da excentricidade e da média, filtro digital, manipulação na forma de arquivos e as implementações do método da reversão e do método aprimorado da reversão. O programa foi desenvolvido em linguagem Delphi. Para o uso do método da reversão é utilizado um sensor não contactante e é necessário alta Repetibilidade na aquisição de dados. O outro método, denominado método aprimorado da Reversão, permite uma medição de circularidade com precisão superior à do movimento rotativo da mesa empregada na medição utilizando-se de dois sensores não contactantes e conseguindo eliminar os erros acidentais. Um dos principais objetivos desta pesquisa é demonstrar que é possível a obtenção do erro de circularidade de peças com precisão superior a dos componentes utilizados no sistema de medição, com o uso de recursos mecatrônicos.

Introdução

A situação atual do país acarreta ao término das indústrias que não possuem condições de fornecer produtos com alta qualidade com mínimo custo e baixo preço. No segmento da indústria mecânica, aqueles que não aumentarem sua produtividade e desempenho são superados pelas grandes corporações que possuem condições e crédito suficiente para a compra de equipamentos de alta tecnologia.

Este trabalho tem interesse em uma parte importante do processo produtivo: o setor de qualidade, especificamente a da medição e tratamento das peças fora do padrão (não conformidade). Mesmo hoje em que a inspeção dos produtos é realizado durante todo o processo, o número de peças com falhas ao final do processo pode ser considerada como relevante num mercado competitivo.

Um dos principais componentes mecânicos é o que possui perfil cilíndrico. Este tipo de perfil pode ser obtido via torno mecânico, trefilação, etc. Os produtos vão desde eixos com erro de forma com exigência de precisão de fabricação da ordem de centésimos de milímetro até rolamentos com erro de forma com exigência de precisão de fabricação da ordem de microns.

O que este trabalho apresenta é um sistema de medição de circularidade de componentes mecânicos de perfil cilíndrico que pode chegar a ter a precisão exigida pelas indústrias de rolamentos e, portanto, acima das exigências das demais indústrias mecânicas, com custo de fabricação e operação muito abaixo dos modelos encontrados no mercado. Isto pode ser alcançado não através de melhoras construtivas ou de fabricação que aumentam a confiabilidade, a precisão, e o custo dos componentes físicos do sistema e sim através da aplicação do Método Aprimorado da Reversão (MAR) que será apresentado mais adiante no texto.

2. Objetivos

Este trabalho está continuando com uma pesquisa anterior. Esta possuía a mesma bancada e sensores de contato (apalpadores mecânicos). Porém, a pesquisa anterior não conseguiu realizar a interface entre a medição gerada pelos sensores com o microcomputador. A implementação dos métodos avançados de medição, também não foi realizada. Com base na pesquisa anterior, definiu-se novos objetivos para este trabalho que são mencionados abaixo:

- Melhorar o medidor de circularidade já existente, tanto na precisão quanto na acurácia de medição;
- diminuir a influência das fontes de erros no sistema de medição;
- substituição dos sensores do tipo contactante por modelos não contactantes, diminuindo o efeito de deformação por pressão de agulha (que possibilitam alterações da superfície do corpo de prova que);
- Construção de encoder rotativo de modo a permitir a sincronização da aquisição de dados do sensor com a rotação da mesa e demais acessórios que se tornem necessários no decorrer do trabalho (exemplo: circuitos eletrônicos, amplificadores e circuitos lógicos);
- Instalação, configuração e operação da placa conversora A/D para aquisição de dados do sensor para o microcomputador, realizando, assim a interface que faltava no trabalho anterior;
- Desenvolvimento de programa de uso no microcomputador com procedimentos para aquisição e armazenagem de dados, processamento e análise dos dados (métodos avançados de medição);

3. Metodologia Adotada

Para um sistema convencional de medição possuir alta precisão ele precisa ser constituído fisicamente de componentes de alta precisão de fabricação, o que eleva muito o custo. E que o resultado da medição é o valor direto dos sensores de medição. Isto acarreta num valor em que a precisão é de uma ordem acima dos componentes utilizados no sistema. Ou seja, para que o resultado tenha precisão de 1μm, os componentes do sistema tem que ter precisão de 0.1 μm.

Já o sistema em desenvolvimento não possui na sua constituição física componentes caros de alta precisão. Em geral, são da ordem de centésimos de milímetros, ou seja, 10 μm, que são de fácil obtenção em qualquer tipo de usinagem. Possuem custo muito mais baixo do que o do convencional. Só que o resultado da medição fica com precisão de uma ordem superior, ou seja, 100μm ou 0.1 mm. Como fazer para obter melhores resultados ? Isso é obtido não considerando como resultado da medição a medição direta da peça. Aplica-se compensação de erros com métodos algébricos e, através de um programa de computador, aumento a precisão para a mesma ordem do sistema convencional de 1μm. Este método algébrico a ser aplicado pode ser o método da reversão ou o método aprimorado da reversão que são chamados de métodos indiretos de medição, por não serem a simples medição direta do corpo de prova.

O custo de desenvolvimento de um software, neste caso não muito complexo, é muito menor do que a obtenção de componentes mecânicos de alta precisão.

4. Comparação do Sistema em desenvolvimento com Sistemas Convencionais

Pode-se fazer um breve comparação entre o sistema de medição em desenvolvimento e os sistemas convencionais na figura 4.1.

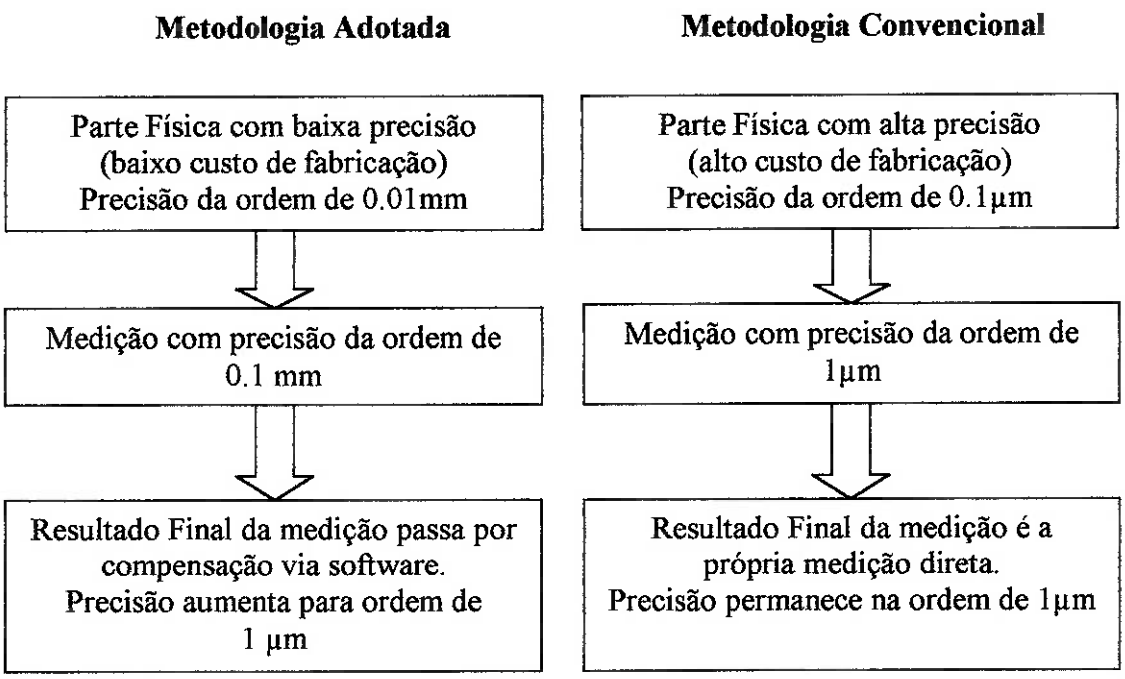


Fig. 4.1. Comparação com sistemas convencionais.

5. Descrição do Sistema em Desenvolvimento

A figura 5.1. oferece um visão geral do sistema de medição:

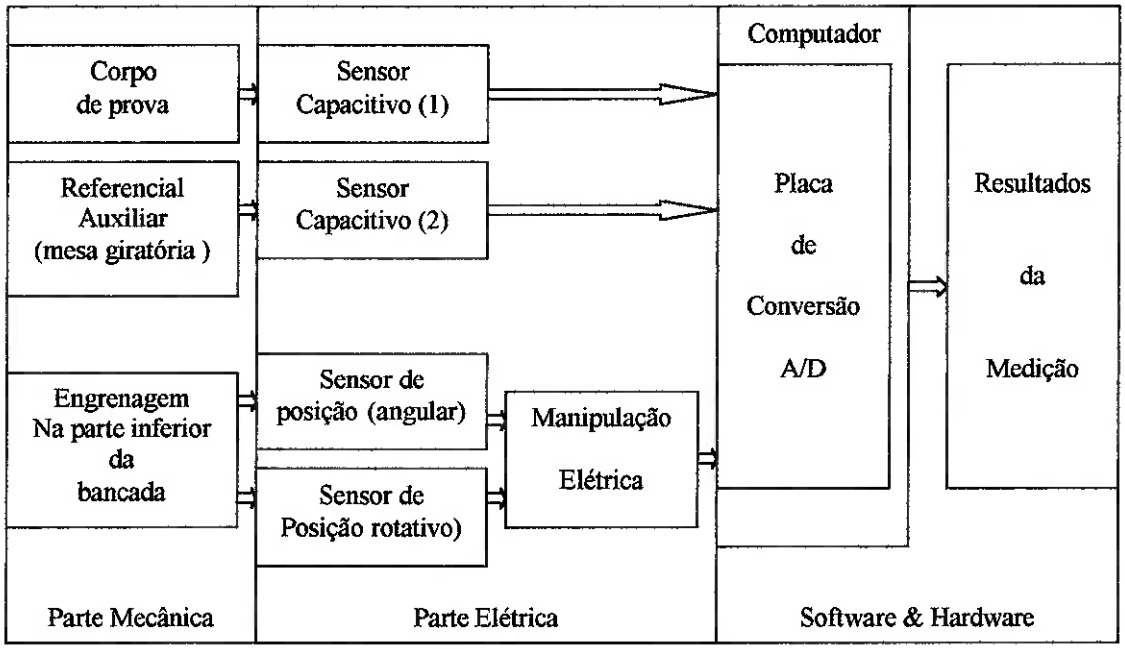


Fig. 5.1. Visão Geral do Sistema de Medição.

Pode-se dividir todo o sistema em três partes principais:

- A parte Mecânica, em que se coloca o corpo de prova, um referencial auxiliar de medição que é usado para no método indireto de medição e um sistema de referencias angular.
- A parte elétrica que é composta por 4 sensores, sendo 2 sensores de medição do corpo de prova e mais 2 sensores para o sistema de referencia angular. Para que seja possível a interface entre os sensores de referencia angular e o microcomputador é necessário uma manipulação elétrica do sinal que provém dos sensores.
- E a parte de hardware e software: hardware seria o próprio microcomputador e a placa de aquisição Analógica/Digital. O software seria o programa utilizado.

6. Parte Mecânica

O trabalho de pesquisa anterior construiu praticamente toda a parte mecânica. Porém, para melhorar o sistema, foi necessário algumas mudanças que implicaram na fabricação de novos componentes mecânicos. A seguir estão uma descrição dos principais componentes da parte mecânica.

6.1. Bancada

A bancada onde será realizada as medições é formada basicamente por uma mesa giratória na parte superior, uma engrenagem na parte inferior, um eixo interligando a engrenagem e a mesa giratória e o corpo do medidor. O eixo se encontra preso a mancais de rolamento de esferas.

O movimento rotativo é fornecido por um pequeno motor elétrico que está ligado por correias ao disco inferior, que possui um formato de polia acima da parte onde se encontra os dentes da engrenagem.

A figura 6.1.1 ilustra melhor as partes acima descritas. Já a figura 6.1.2. mostra as principais dimensões da bancada. Grande parte dos erros de movimento será gerado pelo erro de rotação dos mancais.

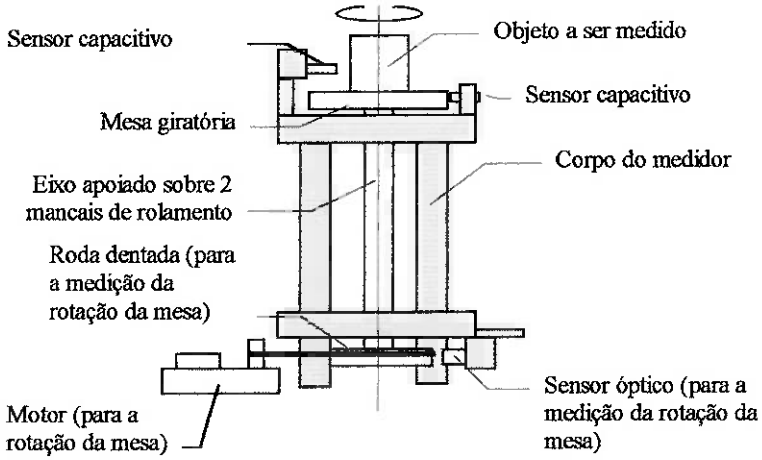


Fig. 6.1.1. Esquema da bancada de aquisição de dados.

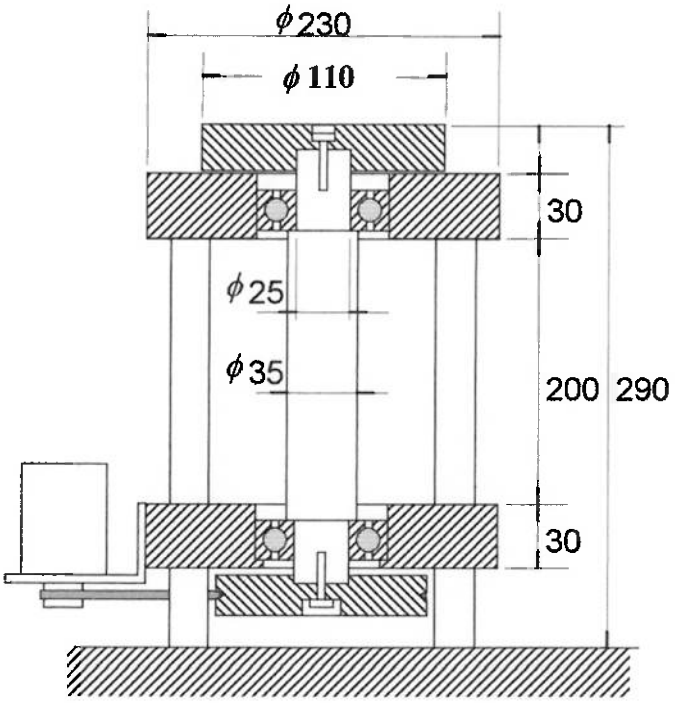


Fig. 6.1.2. Principais dimensões da bancada de medição.

Conforme visto na Figura 5.1, a roda dentada na parte inferior da bancada é utilizada para medir a variação angular do corpo de prova.

Esta bancada está sendo reaproveitada de um trabalho anterior, que não foi concluído. Já possuía acessórios para fixação de apalpadores eletrônicos, porém, não é possível o reaproveitamento desta parte do sistema, pois os testes iniciais mostravam que a força que os apalpadores eletrônicos aplicavam sobre o corpo de prova era suficiente para gerar uma faixa visível. Indicava, portanto, que ocorria um desgaste do corpo de prova e, possivelmente, da ponta do apalpador eletrônico.

O motor indicado na figura 6.1.1. não era, no início do trabalho, capaz de girar o eixo a baixa velocidade. Ele foi substituído por outro modelo que, em conjunto com uma redução, permitiu um torque maior e velocidades mais baixas. O motivo para se desejar velocidades mais baixas, é que assim é possível reduzir os níveis de ruído durante a medição com a aplicação de filtros digitais.

6.2. Fixação dos sensores não contactantes

A pesquisa anterior tinha previsto o uso de apalpadores eletrônicos como sensores de medição do corpo de prova e do referencial auxiliar. Porém, os testes iniciais mostraram que a força que estes tipos de sensores aplicavam sobre o corpo de prova era suficiente para gerar uma faixa visível. Indicava, portanto, que ocorria um desgaste do corpo de prova e, possivelmente, da ponta do apalpador eletrônico.

Outro motivo para a substituição do tipo de sensor utilizado foi o fato de que os testes iniciais também demonstravam altos níveis de ruídos, que diminuíam a precisão dos resultados. A variação lida pela placa A/D era da ordem de 0.4 μm com o sensor lendo uma superfície parada.

A necessidade da montagem dessa nova fixação é que a do trabalho anterior não permite colocar o sensor capacitivo na direção do raio do corpo de prova devido a diferenças geométricas da forma dos dois tipos de sensores (contactante e não contactante).

Um esboço de como ficou a instalação da nova fixação pode ser vista na figura 6.2.1. Os desenhos de fabricação se encontram no anexo.

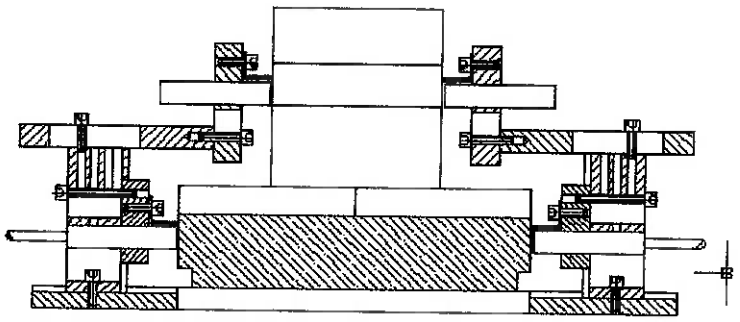


Fig. 6.2.1. Montagem da nova Fixação.

7. Parte Elétrica

Conforme explicado, o movimento angular do corpo de prova e da mesa é determinado utilizando-se a engrenagem na parte inferior da bancada como referencial. Com o auxílio de dois sensores ópticos que se localizam na parte inferior da bancada de medição, gera-se pulsos elétricos de sincronização. Um deles tem como função indicar o início e final de uma volta e o outro indica ângulo de rotação.

A seguir ser estão descritos os principais componentes da parte elétrica.

7.1. Circuitos para gerar sinal de sincronismo

Para que a medição seja feita a intervalos regulares e independente da velocidade de giro do sistema, é necessário o uso de um sinal de sincronismo externo que venha da bancada de medição. Este sinal é obtido inicialmente a partir dos encoder's que lêem a engrenagem na parte inferior da bancada de medição. Para que este sinal seja conveniente, ele passa por um pequeno circuito de manipulação. Um esquema elétrico deste circuito se encontra em anexo.

O que o circuito faz pode ser visto na figura 7.1.1.

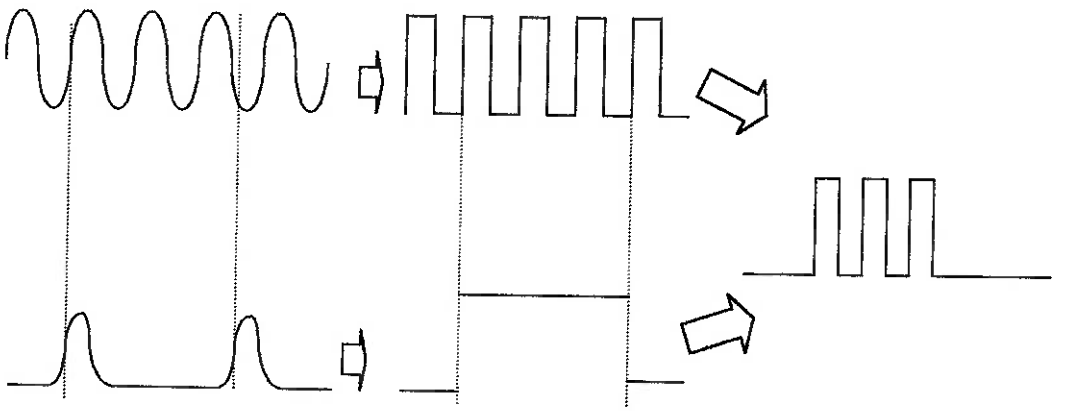


Fig. 7.1.1. Manipulação do sinal de sincronismo externo.

Inicialmente os sinais são enquadrados, ou seja, transformados em sinais TTL. Em seguida eles são unidos através de uma operação lógica e em finalmente enviados para a placa de conversão A/D. Este sinal de controle externo, conforme o manual da placa conversora, tinha que ser TTL e com amplitude de 0 e 5 V. Mais tarde, utilizou-se outro método em que não era necessário tanto rigor.

7.2. Placa Conversora A/D

A placa conversora a ser utilizada é da marca LYNX, possui 10 bits de resolução e alcance de ± 5V.

7.3. Sensores Capacitivos

Os sensores capacitivos a serem utilizados na medição direta do corpo de prova são modelos ADE 3800 com alcance máximo de ±10V para ±25μm. Para este trabalho, configurou-se o ganho para se adequar a placa A/D: ± 5V para ± 25 μm.

Estas resoluções e alcances da placa conversora e dos sensores capacitivos foram consideradas satisfatórias para o tipo de trabalho a ser realizado, pois indicam que a menor unidade possível de ser detectada é 12.5μm/1024=0.0122μm.

8. Software

Foi escolhido para o desenvolvimento do software a linguagem Pascal por ter sido indicado no manual da placa de aquisição (existência de vários exemplos nesta linguagem) e por ser de fácil entendimento.

O software básico, primeiramente, foi desenvolvido para ambiente D.O.S. Após atingir um certo nível de complexidade do software, o programa foi convertido para ambiente Windows 9x. Esta mudança tinha como principal vantagem a interface gráfica de fácil uso para a maioria dos usuários. A principal vantagem foi a da possibilidade de se utilizar nomes longos no armazenamento de arquivos e da programação visual ser mais de fácil utilização. Todas as versões possuem os recursos básicos.

O software a ser utilizado nesta pesquisa pode ser dividido em 4 partes fundamentais:

- Coleta de dados;
- Processamento dos dados coletados;
- Apresentação dos resultados obtidos com os dados coletados;
- Armazenamento dos resultados e dados coletados para futuros testes;

A seguir, menciona-se cada versão do programa (para D.O.S. e Windows 9.x) e cada parte acima mencionada.

8.1. Coleta de Dados

O manual da placa conversora A/D possui o procedimento básico para fazer a aquisição de dados (pág. 4-5). Com esse procedimento, resta apenas desenvolver o método para a aquisição sincronizada de dados.

Isto é feito do seguinte modo: o sinal de controle enviado pelo encoder's à entrada da placa conversora A/D torna possível sincronizar a tomada de dados de modo que a cada volta, a engrenagem na base inferior, que possui 128 dentes polidos (superfícies refletoras), envie 256 pulsos acarretando em 256 aquisições de sinal do apalpador eletrônico por volta. Observe a figura 8.1.1.

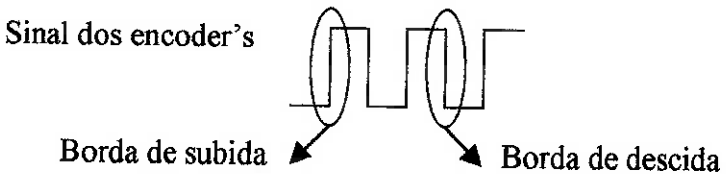


Fig. 8.1.1. Instantes para coleta de dados.

Para ocorrer esse sincronismo é necessário programar a placa corretamente e criar um algoritmo para que a placa espere o início da coleta de dados. Isso é feito controlando-se os contadores da placa e o modo de operação.

Utilizando-se o modo 0 da placa, quando o sinal da porta de entrada da placa conversora estiver com 0V (nível lógico baixo), o contador permanece com o último valor (por exemplo escreveu-se no contador o valor de 10000). Logo que o sinal mudar (borda de subida), o valor do contador também mudará e com isso pode-se acionar a aquisição de dados. Pode-se aplicar este mesmo princípio para quando desejamos a aquisição de dados quando for borda de descida do sinal de entrada, o que nos proporciona o dobro de situações em que podemos fazer aquisição de sinal. Maiores informações podem ser obtidas no manual da placa conversora A/D.

Um outro modo utilizado, mais simples, é o uso de um dos canais de leitura disponíveis (no total são 16) sendo que quando o sinal mudar de amplitude bruscamente (exemplo: de 0.5 V para 1V e vice-versa) ocorre a aquisição de sinal enviado pelos sensores capacitivos. A vantagem deste método, é a possibilidade de o operador indicar os níveis em que ocorre a aquisição, ou seja, não é necessário que o sinal tenha amplitude de 5V (nível lógico alto) e 0V (nível lógico baixo).

Para parar o processo de coleta de dados, programou-se um contador de tempo. Deste modo, quando o sinal dos encoder's permanecer por muito tempo com baixa amplitude (indicando um volta completa realizada) o microcomputador termina o procedimento de aquisição.

A aquisição de dados de forma sincronizada com o movimento rotativo dos dentes polidos da engrenagem da parte inferior da bancada de medição se encontra e operação e os testes iniciais indicam funcionamento adequado. A leitura inicia-se na primeira borda de

subida do sinal de controle e termina após o sinal de controle ir para nível lógico baixo por muito tempo, realizado a aquisição de 256 pontos.

Atualmente, é utilizado 3 canais de leitura, 2 para os sensores de medição e o outro para medir os encoder's, utilizando o método acima mencionado.

8.2. Processamento dos Dados

Após ter coletado os dados, deve-se fazer manipulações algébricas para diminuir certos erros. Isto é, compensação por software.

Os principais tipos de compensação utilizados são:

- Eliminação do valor médio do sinal: Após o sinal ser armazenado na memória, retira-se o valor médio do conjunto de pontos. Isto é necessário pois se deseja apenas a variação relativa da superfície. Este cálculo é simples: soma-se todos os 256 valores e divide-se esta soma pela quantidade de pontos (256). Em seguida subtrai-se de todos os pontos o valor encontrado;
- Eliminação do erro de centralização do corpo de prova – excentricidade: Até o momento, não se comentou nada a respeito da centralização do objeto no medidor, ou seja, o procedimento no qual a centróide do objeto é justaposta com o centro do eixo do medidor. A não centralização pode implicar no surgimento na leitura do medidor, de uma componente periódica de período 1 ciclo por volta do eixo do medidor e de amplitude igual à excentricidade entre o centro de rotação e a centróide do perfil do objeto. Esta componente é indesejável na avaliação da circularidade pois aumenta o alcance necessário para se fazer a medição.

A figura 8.2.1 representa o erro de centralização e suas conseqüências na medição de circularidade:

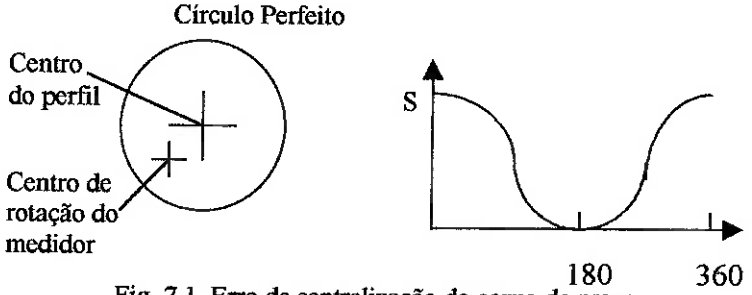


Fig. 7.1. Erro de centralização do corpo de prova.

A eliminação desta componente, 1 ciclo por volta, pode ser feita atuando-se na centralização física do objeto no medidor, mas também pode ser feita matematicamente, o que é muito mais cômodo em muitos casos. O processo matemático consiste primeiramente em expandir S do medidor numa série de Fourier, usando para isso, um algoritmo computacional conhecido como FFT (Fast Fourier Transform). Uma vez feita a transformação, a componente de ordem 1 é igualado a zero, e em seguida procede-se à transformação inversa de Fourier, usando neste caso o algoritmo IFFT (Inverse Fast Fourier Transform).

Outro método, seria calcular o centro do perfil em relação ao centro de rotação do medidor e, através de relações geométricas, retirar a componente S da medição fazendo-se as correções necessárias. A posição do centro do perfil em relação ao centro de rotação do medidor é de fácil obtenção. Inicialmente define-se um sistemas de coordenadas x-y com origem no centro de rotação do medidor. Sendo x_i e y_i as coordenadas de cada um dos n pontos medidos, a coordenada do centro do perfil é dado pelo ponto (a,b) sendo:

$$a = 2\frac{\sum_{i=0}^{n} x_i}{n} \qquad (7.1)$$

$$b = 2\frac{\sum_{i=0}^{n} y_i}{n} \qquad (7.2)$$

A demonstração das equações 7.1 e 7.2 pode ser vista no livro Metrology & Precision Engineering, A.J.T.Scarr, Cap. 8 – p. 125.

Escolheu-se o uso do método com fft.

- Filtros digitais: Sua implementação é muito mais simples que as dos filtros analógicos, além de terem um desempenho muito superior.

No momento está previsto o uso de filtros digitais em duas circunstâncias:

a) Durante a coleta de dados – fazendo-se várias medições no momento adequado (borda de subida ou descida do encoder) e em seguida pegar-se o valor médio;
b) Após a tomada dos 256 pontos – utilizando-se filtragem por fft, em que se anulam componentes de alta ordem. Neste trabalho, as componentes acima da ordem 10 foram igualadas a zero;

- Método da Reversão: consiste na implementação do método da reversão;
- Método Aprimorado da Reversão: consiste na implementação do método aprimorado da reversão;

8.3. Apresentação dos Resultados

Assim que se tem os dados coletados e processados, eles devem ser apresentados ao usuário de forma coerente e de fácil compreensão. Para isto, foram escolhidos 3 formas de apresentação:

- Impressão direta na tela dos valores coletados: simplesmente mostra os valores obtidos (escala em V ou em μm);
- Gráfico Linear: Torna fácil visualização do sinal e comparações com outros sinais e a verificação do nível de ruído através da verificação de oscilação contínua;
- Gráfico Polar: Este tipo de representação é muito encontrado em livros.

8.4. Armazenamento dos Dados/Resultados

Os dados, assim que são coletados, processados e apresentados, devem ser guardados para futuras comparações e análises (no próprio software ou em outros).

Para facilitar a compatibilidade dos dados com outro software, decidiu-se usar o formato de arquivo tipo texto (.txt). Os valores são gravados na escala V, e na última linha fica o valor que corresponde a escala para conversão para μm.

8.5. Software para D.O.S.

Um programa escrito em linguagem Pascal para ambiente D.O.S. foi desenvolvido para a manipulação dos dados coletados. A versão atual possui os seguintes recursos: aquisição de dados, manipulação de arquivos, extração do valor médio da curva, extração da excentricidade, rotinas gráficas(linear e polar), testes simples de repetibilidade (obtenção

de média e desvio padrão de 10 medições) e as implementações do método da reversão e do método .

A seleção das funções é feita através de "menus" de comando (exemplo na figura 8.5.1). O programa já possui rotinas gráficas para visualização do sinal através de gráficos lineares e polares (exemplo nas figura 8.5.2 e figura 8.5.3). Os dados usados para verificar o funcionamento dos gráficos foram obtidos com o uso dos apalpadores eletrônicos.

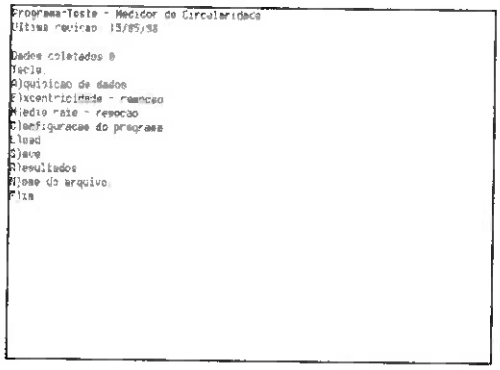


Fig. 8.5.1. Menu do Software (DOS).

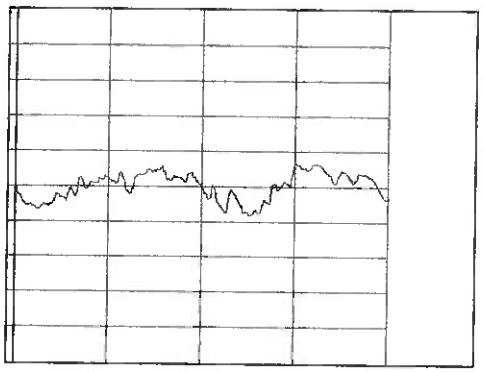


Fig. 8.5.2. Exemplo de Gráfico Linear (DOS).

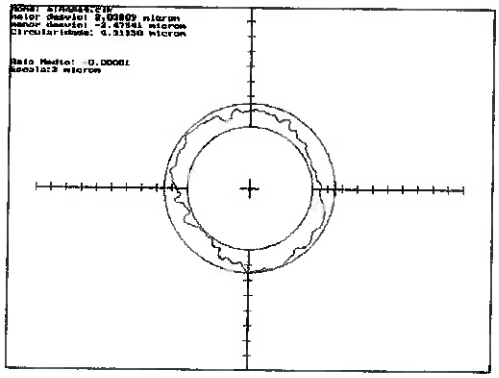


Fig. 8.5.3. Exemplo de Gráfico Polar (DOS).

8.6. Software para Windows 9x

Esta nova versão do software foi feita para que se pudesse ter acesso a nomes longos e devido a programação visual. Possui praticamente os mesmo recursos que as versões anteriores. Porém, somente esta versão receberá os recursos de filtro digital após coleta de dados. Note porém, que não foi encontrado meios para fazer a aquisição de dados no Delphi, sendo que foi decidido o uso de 3 programas ao invés de um: um para a

interface (feito em Delphi) e dois para a coleta de dados (para o método da reversão e o método aprimorado da reversão).

A figura 8.6.1. ilustra como é a interface.

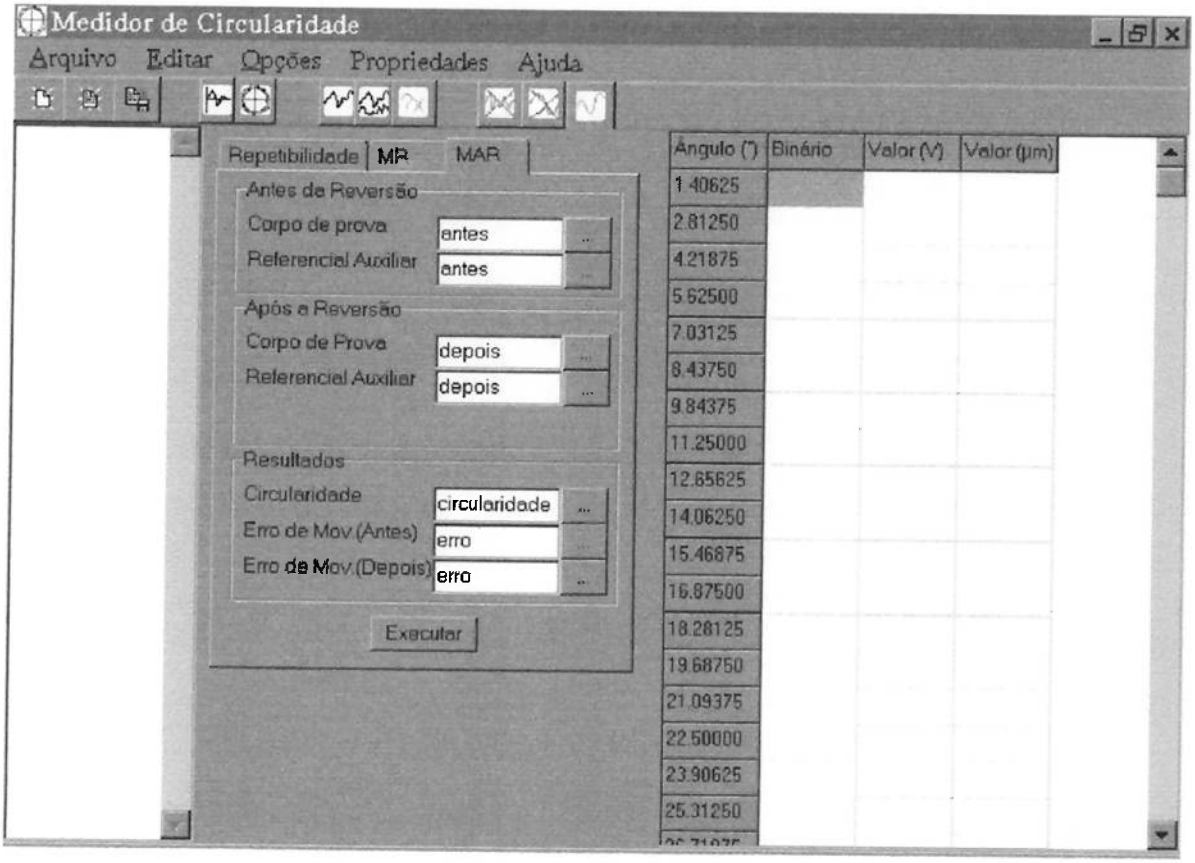


Fig. 8.6.1. Interface do programa para Windows 9x.

9. Descrição das Atividades Realizadas

Conforme já comentado anteriormente, este trabalho continuou um outro trabalho. Uma parte do trabalho anterior foi possível de ser reaproveitada. As principais realizações feitas, podem ser vistas na tabela 1:

Tabela 1.

Parte Mecânica	- Substituição dos sensores de Medição para sensores capacitivos não contactantes;
	- instalação de acessórios para Fixação para os novos sensores (figura 9.1);
	- substituição do motor de acionamento por um de maior torque a baixa rotações (figura 9.2);

Parte Elétrica	- montagem de cabos/conexões;
	- instalação e configuração da placa de conversão A/D;
	- Montagem de circuito eletrônico do sinal de controle de sincronismo para aquisição;
	- Montagem dos Encoders;

Software	- Programa para ambiente Windows 9x com interface simples de operação e configuração;
	- Implementação do M.R. e do M.A.R.
	- implementação para visualização de gráficos lineares e polares;
	- implementação de rotinas para filtragem (ex.: valor médio, excentricidade, fft);

Testes do sistema de Medição	- Repetibilidade;
	- Uso do M.R. e do M.A.R.;
	- comparação dos resultados (MR e MAR) com sistema de medição de coordenadas;
	- Verificação do método mais robusto;

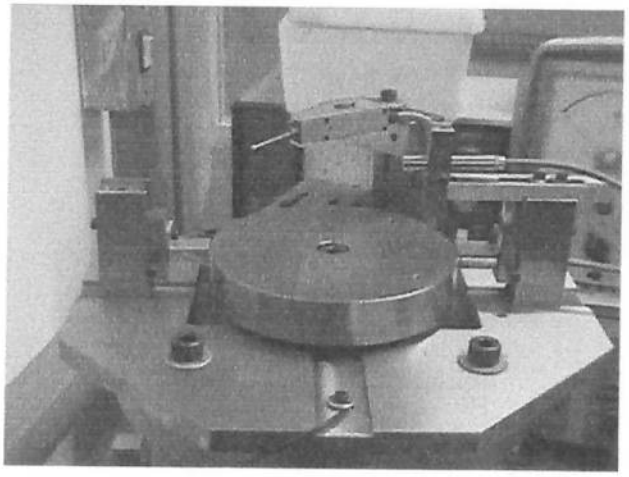


Fig. 9.1. Foto da nova fixação dos sensores de medição.

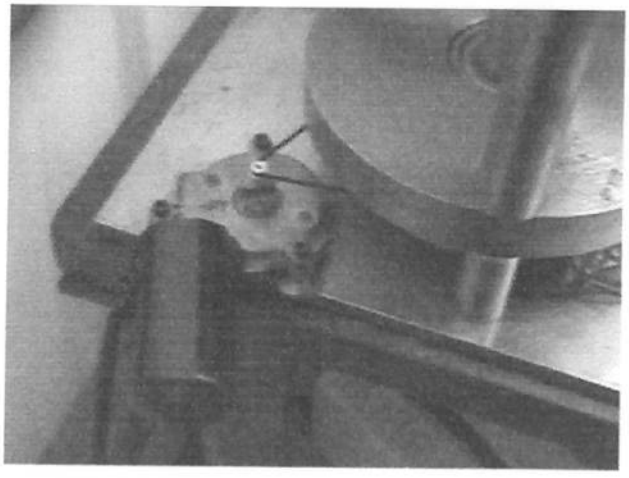


Fig. 9.2. Foto do novo motor de acionamento.

10. Resultados

10.1. Repetibilidade

- Objetivo: verificar se a mesa de aquisição de dados possui alta Repetibilidade (medições com pequenas diferenças entre elas e em relação a média - baixo desvio padrão). Isto aumenta a confiabilidade da bancada de medição e aumenta a confiabilidade dos resultados obtidos pelo método da reversão.
- Método Utilizado: Realizou-se a aquisição de 10 coletas de dados seguidamente. (A escolha do valor dez foi feita arbitrariamente, sem nenhuma técnica estatística que indicasse esse valor). As amostragens foram colocados sob forma de gráfico linear onde poderiam ser melhor visualizadas. Poderia ser também realizado uma manipulação matemática (extração de excentricidade e extração do raio médio) nos dados coletados, porém, neste tipo de experiência ocorre mais ênfase na medição direta da peça. Criou-se um gráfico linear contendo todas as medições (figura 10.1.1) e outro gráfico contendo a curva média com o seu respectivo desvio padrão (figura 10.1.2). Sendo que o valor máximo do desvio padrão foi de 0.87 V (4.35 μm).
- Fatos observados: Foi percebido que as curvas tinham uma "aparência" próxima uma das outras e que a variação entre as curvas estava entre +1V/-1V (figura 10.1.1).
- Conclusões: A estimativa do erro acidental da mesa de coleta de dados é de +1V/-1V, ou seja, da ordem de 10μm. Estas condições desfavorecem o uso do método da reversão.

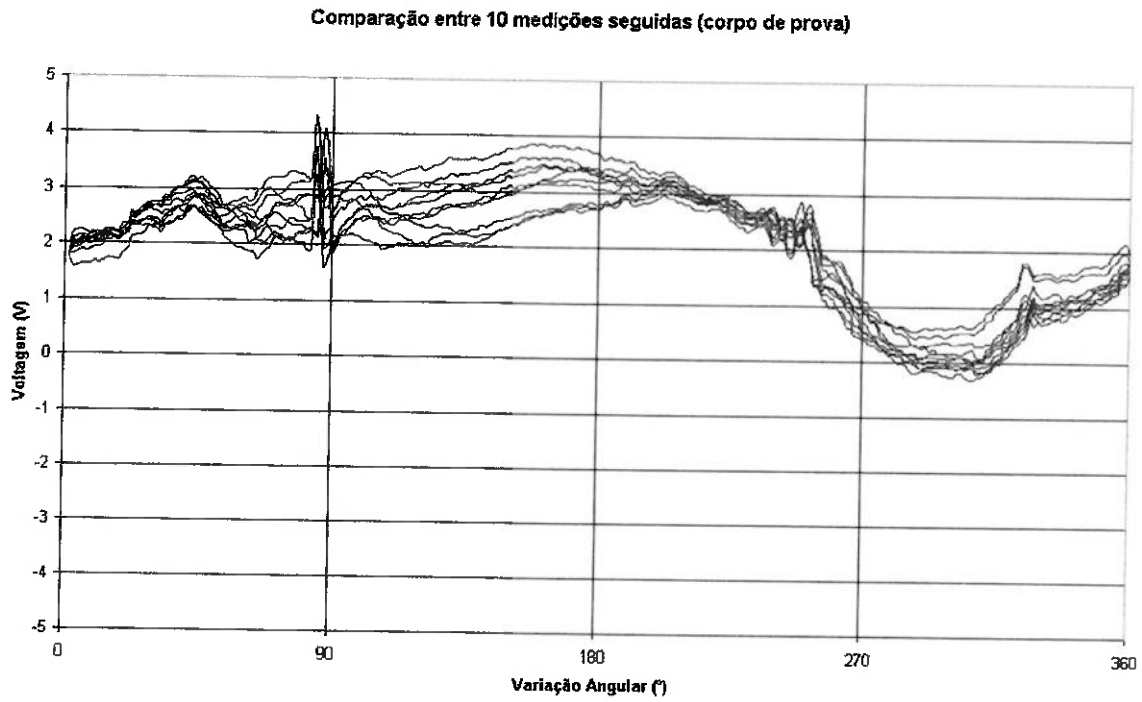


Fig. 10.1.1. Este gráfico permite observar como as medições foram obtidas com pouca dispersão (aproximadamente 2V – correspondente a 10 μm).

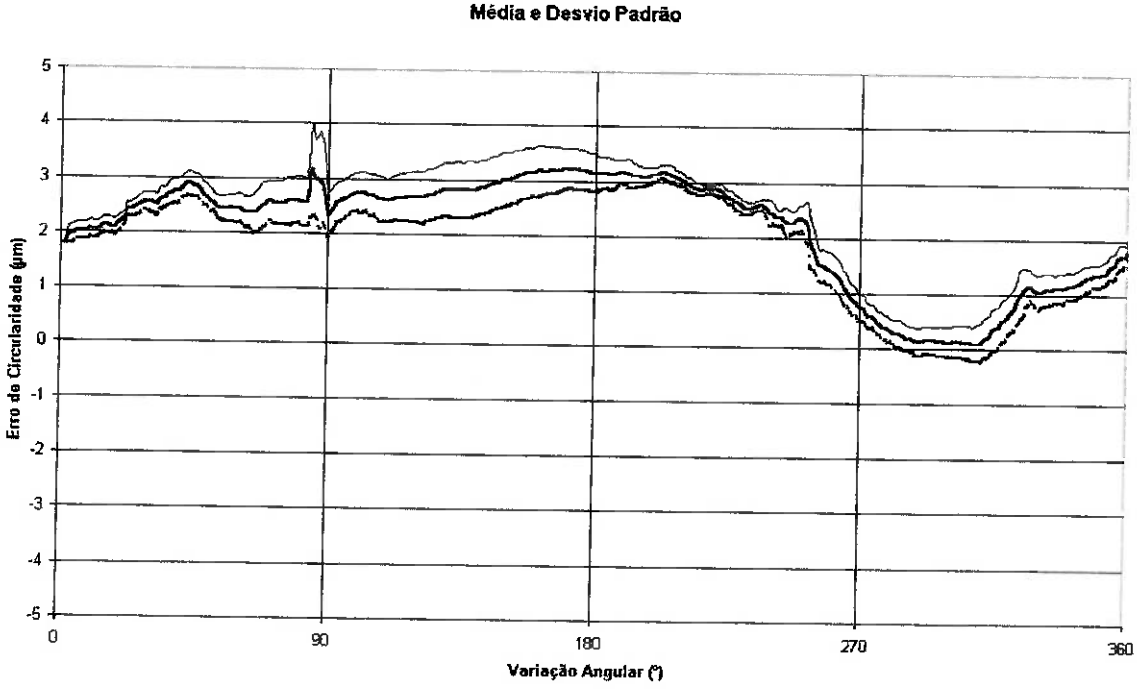


Fig. 10.1.2. Verificação da dispersão durante a medição e trechos onde ocorrem maiores diferenças entre as medições.

10.2. Método da Reversão

- Objetivo: conseguir isolar o erro de movimento rotativo do erro de circularidade (erro de forma) do corpo de prova.
- Método Utilizado: Através do princípio do método da Reversão, realizou-se a coleta de duas amostragens trocando a posição do apalpador eletrônico e do corpo de prova (objeto cilíndrico com altura de 70 mm e diâmetro de 58 mm), ou seja revertendo-os em 180°. Esses mesmos valores serão a seguir utilizados para o MAR, juntamente com a medição da base. Os valores das medições podem ser vistos nas figura 10.2.1. e 10.2.2. Estas amostragens devem sofrer manipulação matemática (extração de excentricidade e raio médio através de Fast Fourrier Transform - fft). Para se estimar o erro de uma medição para outra, foram feitas duas medições do corpo de prova.
- Fatos observados: As medições tornaram possível criar vários gráficos que foram reunidos em dois gráficos (um para circularidade – figura 10.2.3. e outro para o erro de movimento – figura 10.2.4.) onde pode-se notar uma curva com um alto desvio. A comparação entre os erros de movimento indica uma variação de no máximo 5μm.
- Conclusões: O método da reversão permitiu obter resultados com precisão apesar das limitações estruturais da mesa de aquisição de dados (uso de mancais de rolamento). Como era esperado, as curvas dos erros de movimento estão muito semelhantes, levando-se em consideração as limitação de precisão de 10 μm (valor de repetibilidade).

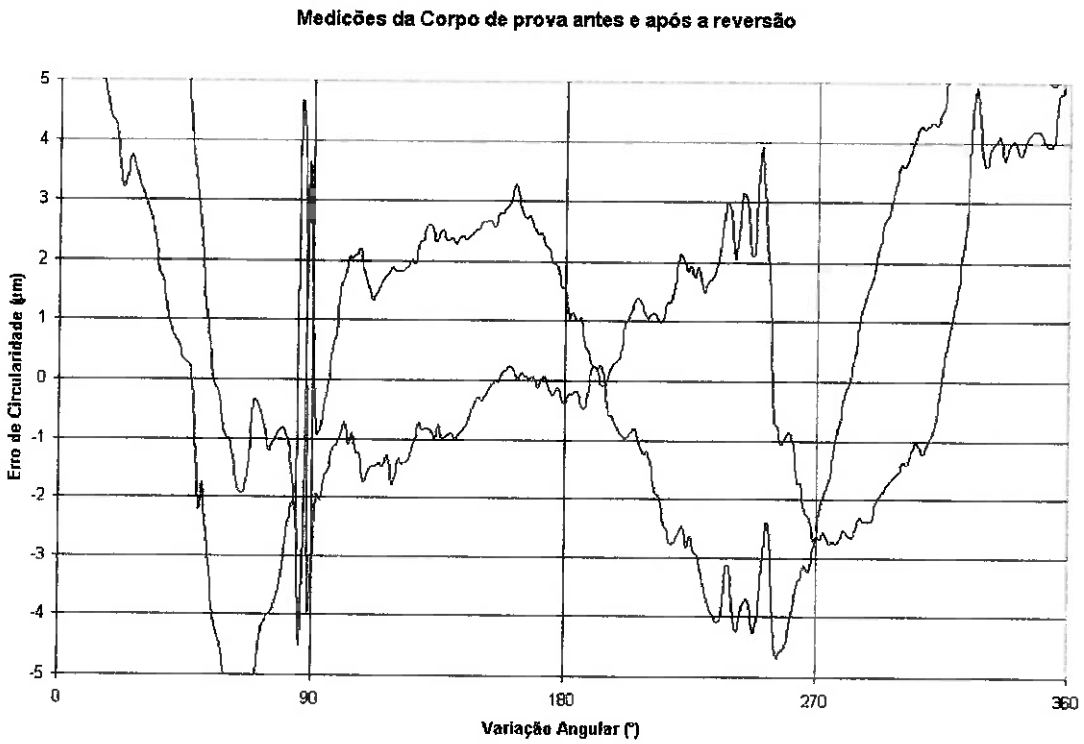


Fig. 10.2.1. Medições do corpo de prova cilíndrico de diâmetro 58 mm e altura 70 mm (1° Par de Medição).

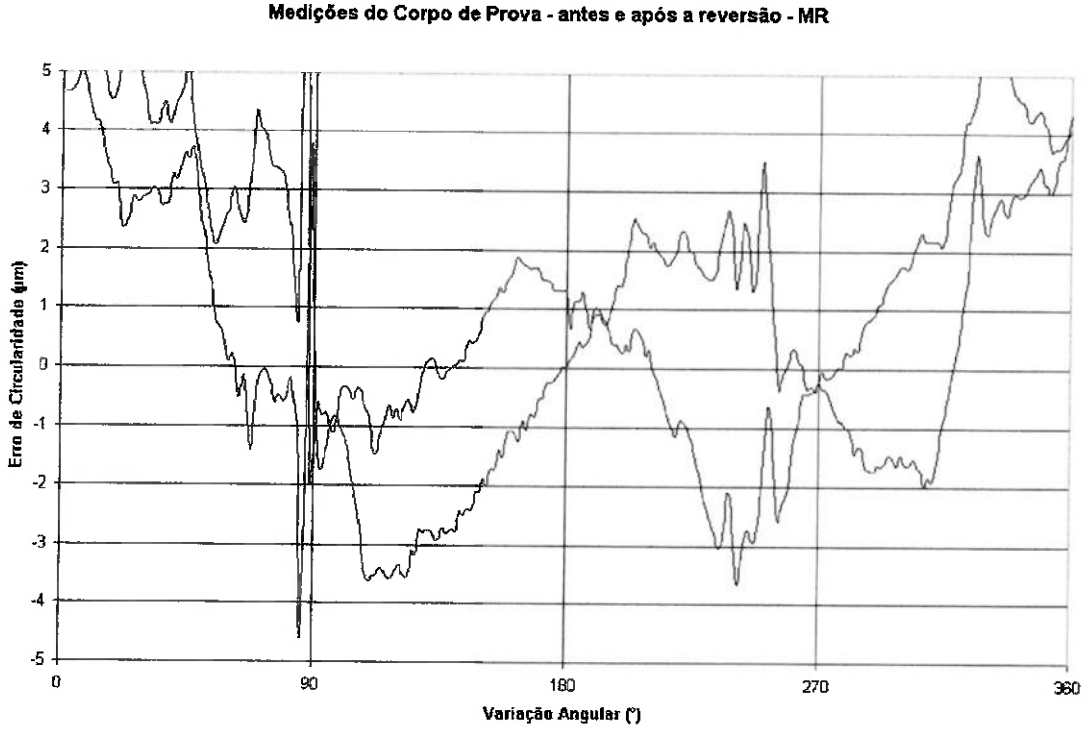


Fig. 10.2.2. Medição do corpo de prova (Segundo Par de Medição)

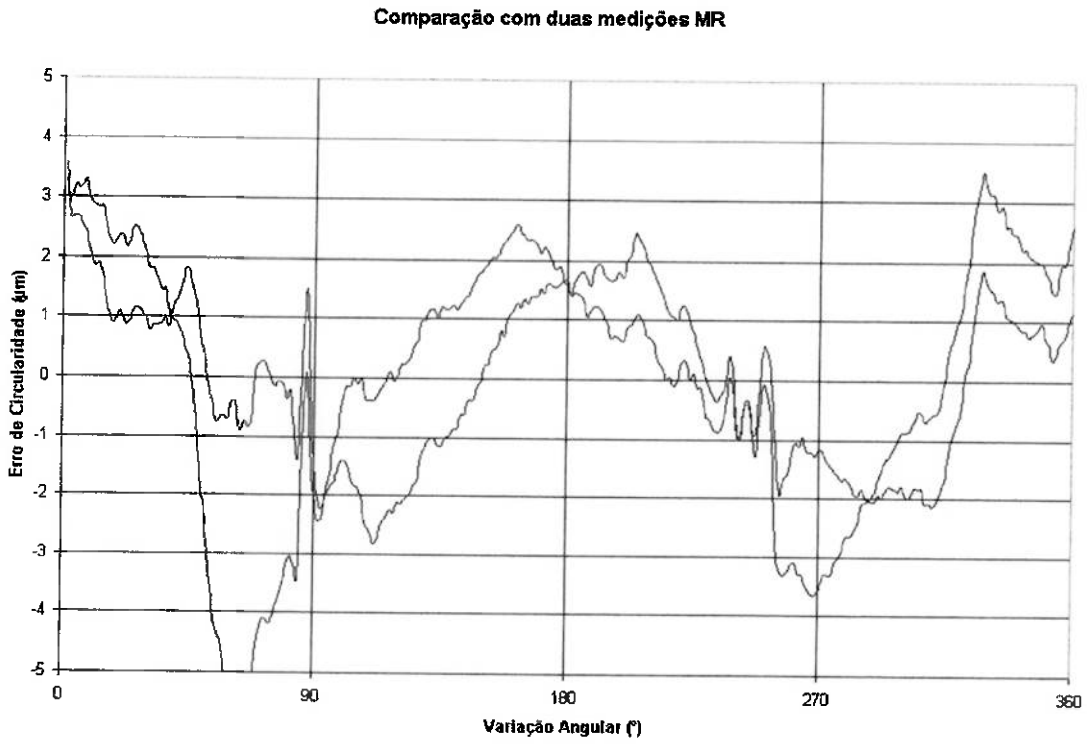


Fig. 10.2.3. Comparação entre os erros de forma obtidos pelas curvas das figuras 10.2.1 e 10.2.2.

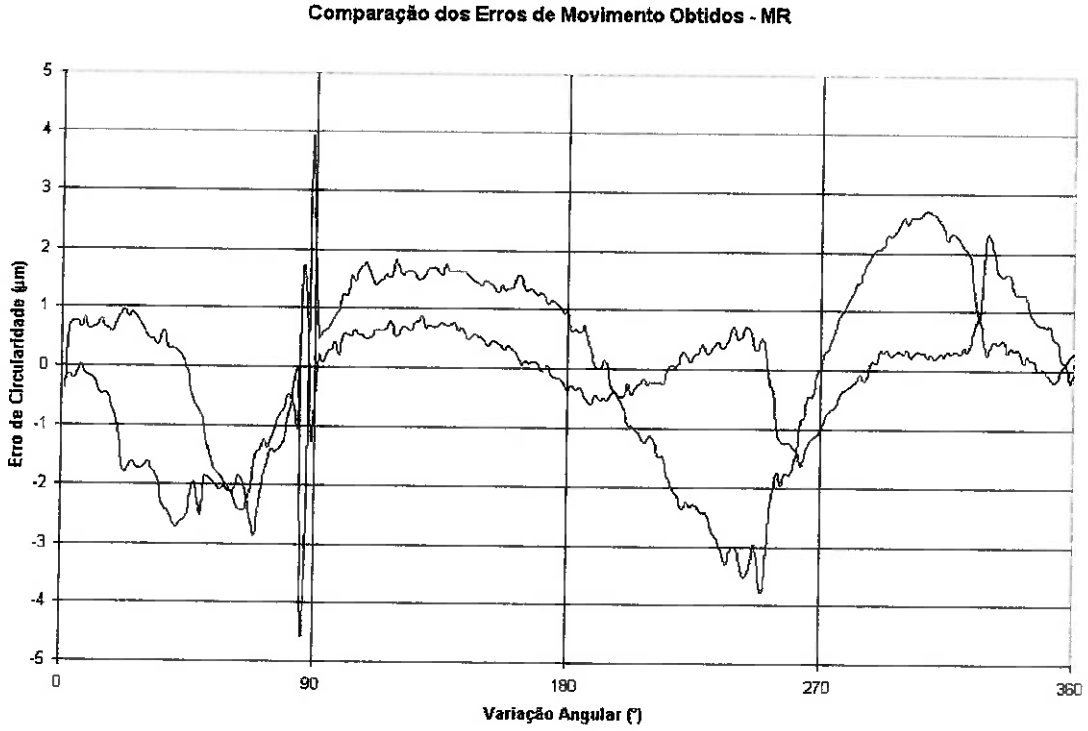


Fig. 10.2.4. Comparação entre os erros de movimento obtidos pelas curvas das figuras 10.2.1 e 10.2.2.

10.3. Método Aprimorado da Reversão

- Objetivos: conseguir isolar o erro da mesa de aquisição (erro de movimento) do erro de circularidade (erro de forma) do corpo de prova, entretanto com menor influência do erro acidental.
- Método Utilizado: Consiste em se utilizar o método aprimorado da reversão, em que ocorre a medição simultânea de duas peças, sendo uma o corpo de prova (figura 10.2.1. e 10.2.2.) e outra servindo de referência (figuras 10.3.1. e 10.3.2). Ao se realizar apenas a reversão do corpo de prova e do primeiro sensor, permanecendo o segundo sensor na mesma posição e referencia, pode-se, através de métodos algébricos, obter o erro de forma do corpo de prova (figura 10.3.3) e o erro de movimento da bancada de medição antes e após a reversão (figura 10.3.4 e 10.3.5). A diminuição da influência do erro sistemático é devido ao fato de que a medição secundária (referencial) torna possível extrair da medição primária (corpo de prova) o erro acidental. Estas amostragens devem sofrer manipulação matemática (extração de excentricidade e raio médio).
- Fatos observados: após serem feitas várias amostragens, verificou-se que as variações nos resultados obtidos pelo M.A.R. eram menores que as obtidas pelo M.R. Diferença máxima de 3 μm. Os erros de movimento obtidos possuem maior variação que os do MR. Isso indica que o movimento não foi tão repetitivo quanto que se podia concluir através do MR.
- Conclusões: O M.A.R. obteve melhores resultados, logo significa que é um método mais robusto que o M.R. Possui menor influência a erros acidentais durante a medição, fornecendo valores de erro de forma do corpo de prova mais confiáveis. As figuras 10.3.4 e 10.3.5 (erro de movimento) possuem maior variação entre elas, indicando que a componente do erro acidental foi transferido do gráfico do erro de forma (figura 10.3.3) para eles (figuras 10.3.4 e 10.3.5) em comparação aos do MR (figura 10.2.3).

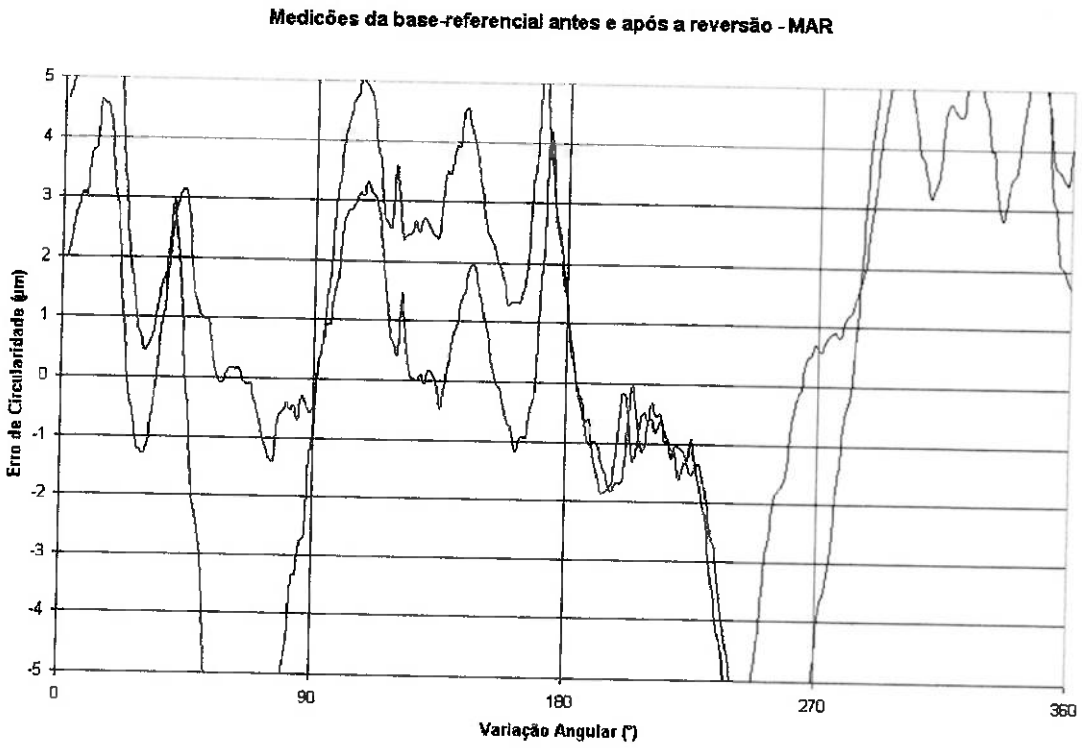


Fig. 10.3.1. Medições do Referencial (Primeiro Par de Medições)

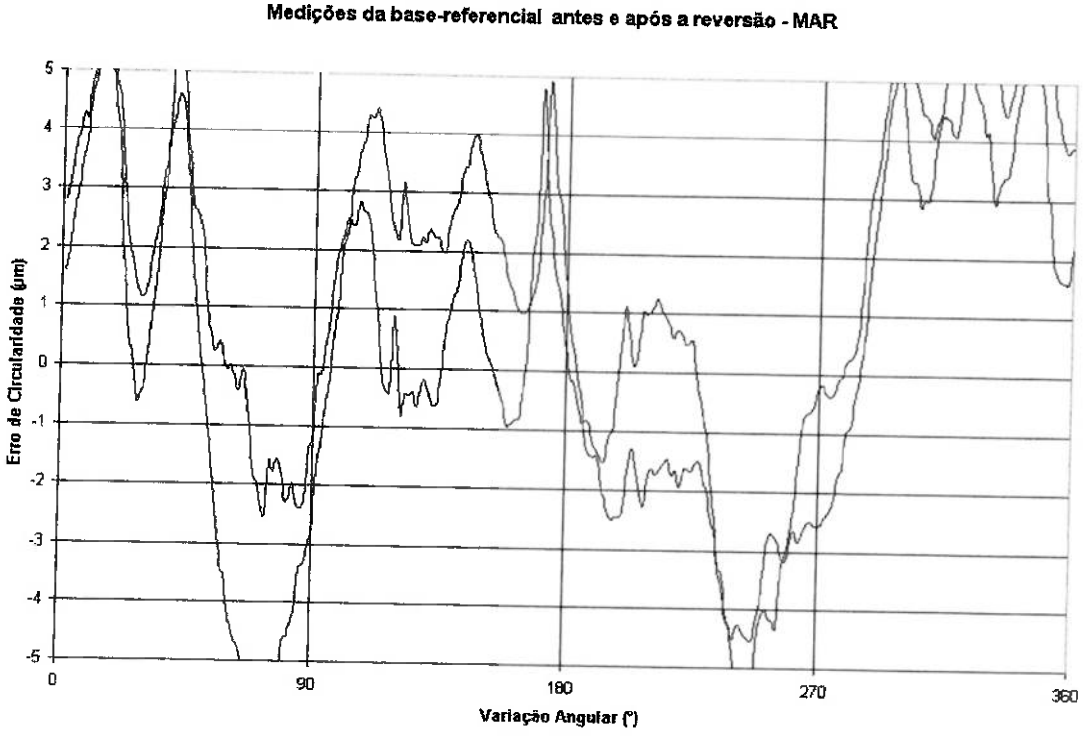


Fig. 10.3.2. Medições do referencial (Segundo Par de Medições).

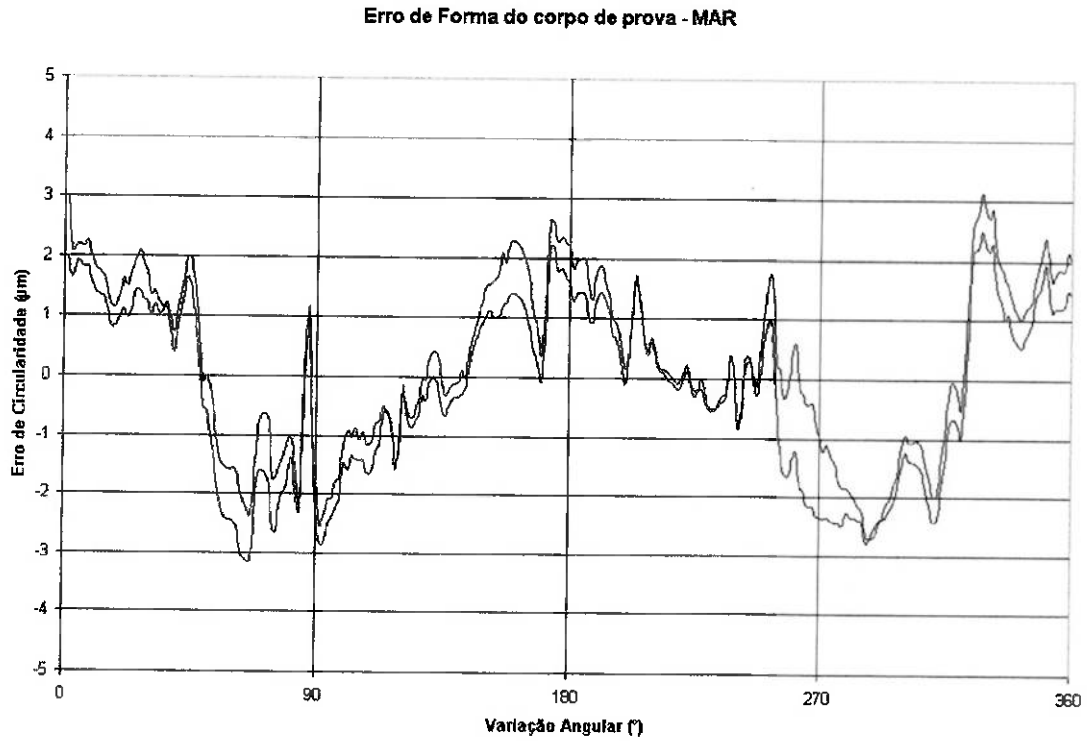


Fig. 10.3.3. Erro de forma obtido das curvas das figuras 10.2.1, 10.2.2, 10.3.1 e 10.3.2.

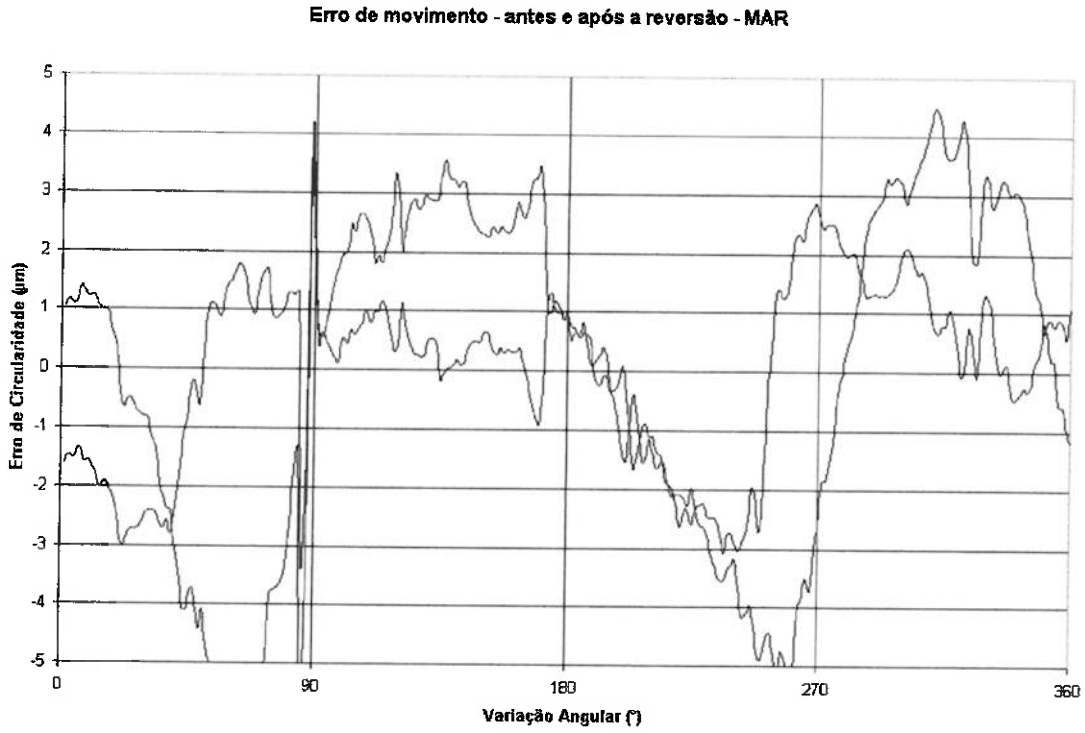


Fig. 10.3.4. Erro de Movimento obtido das curvas das figuras 10.2.1e 10.3.1.

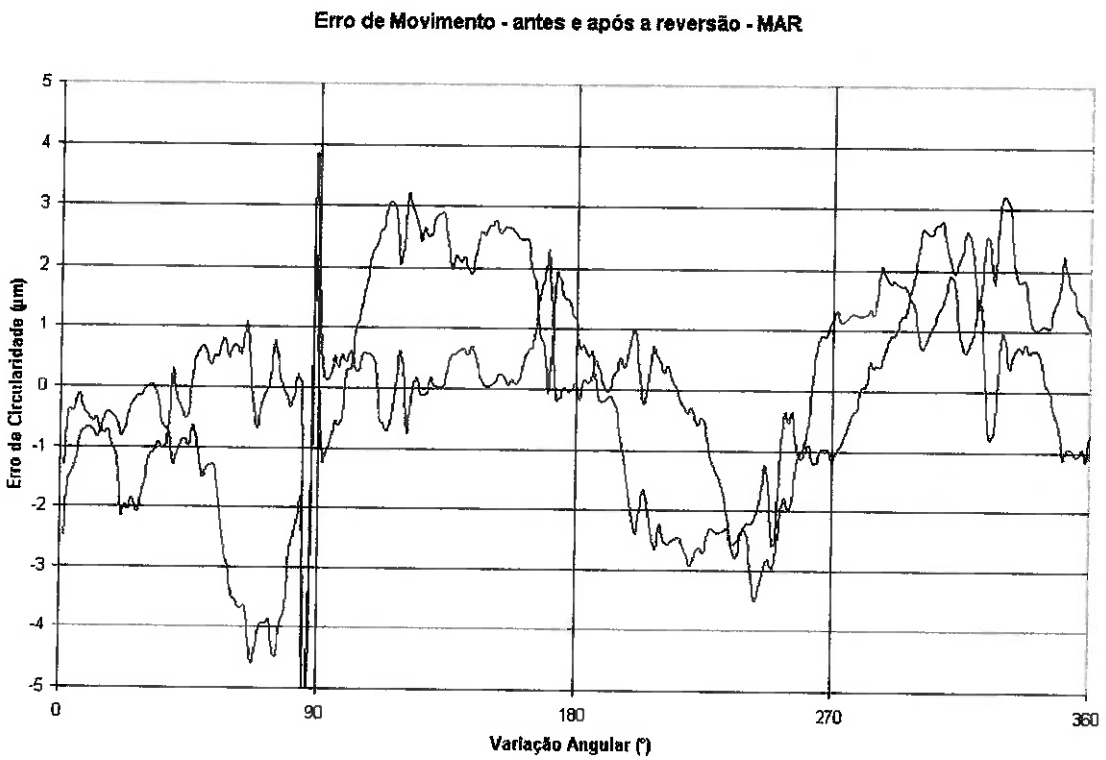


Fig. 10.3.5. Erro de movimento obtido das curvas das Figuras 10.2.2 e 10.3.2.

10.4. Rotação Angular do corpo de prova de um ângulo conhecido

- Objetivos: verificar se há coerência entre os valores obtidos nesta experiência com os da experiência anterior.

- Método utilizado: Novamente o MR e o MAR serão utilizados nos cálculos do erro de forma e do erro de movimento. Consiste em mudar (girar) o corpo de prova de um ângulo conhecido (sugerido: 45°). Com esta mudança, o erro de forma irá transladar enquanto que o erro de movimento deverá permanecer o mesmo. Na figura 10.4.1 pode-se ver as medições do corpo de prova antes e após a reversão e na figura 10.4.2 as medições da base - referencial.

- Fatos observados: O gráfico obtido para a circularidade moveu-se na quantidade do mesmo ângulo utilizado (45°) . Veja a figura 10.4.3 para o erro de forma (MAR) antes de transladar 45° via matemática e a figura 10.4.4 para comparar com um dos erros de forma obtido antes de mudar a posição em 45° usando o MAR e a figura 10.4.5 usando o MR.

- Conclusões: Os procedimentos adotados estão coerentes e que o erro de forma do corpo de prova está próximo dos valor real. A obtenção do erro de forma usando o MAR possui pouca influência da posição do corpo de prova na bancada de medição.

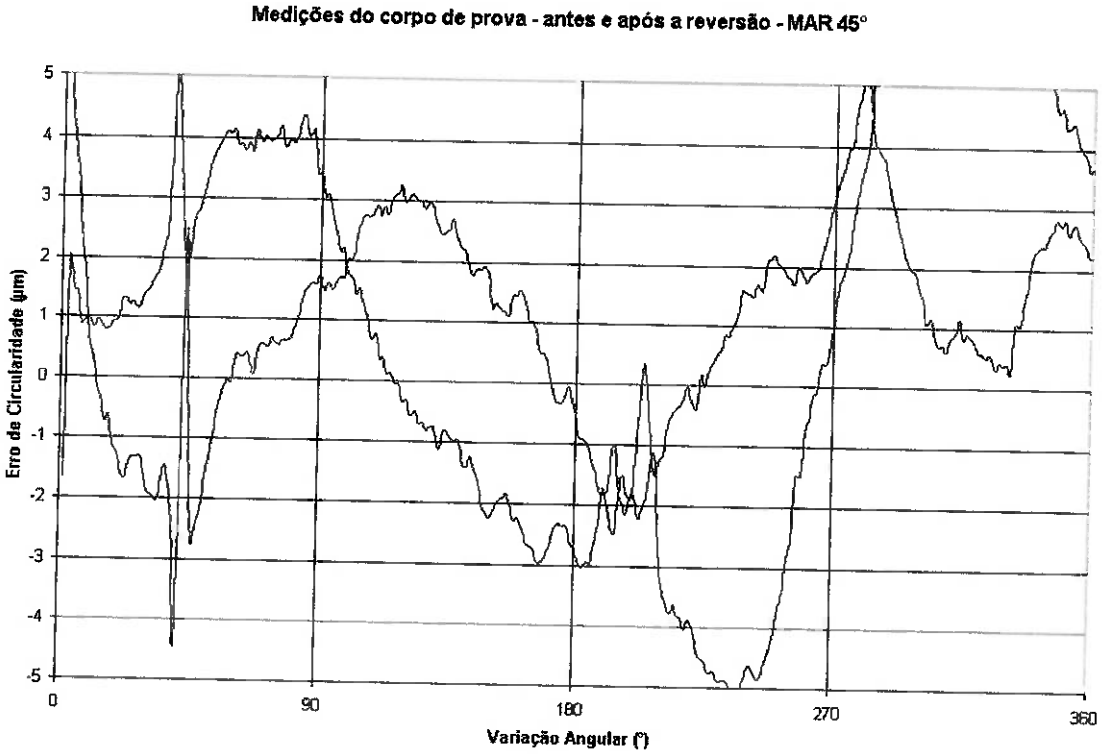


Fig. 10.4.1. Medições do corpo de prova antes e após a reversão com o corpo de prova já girado de 45°.

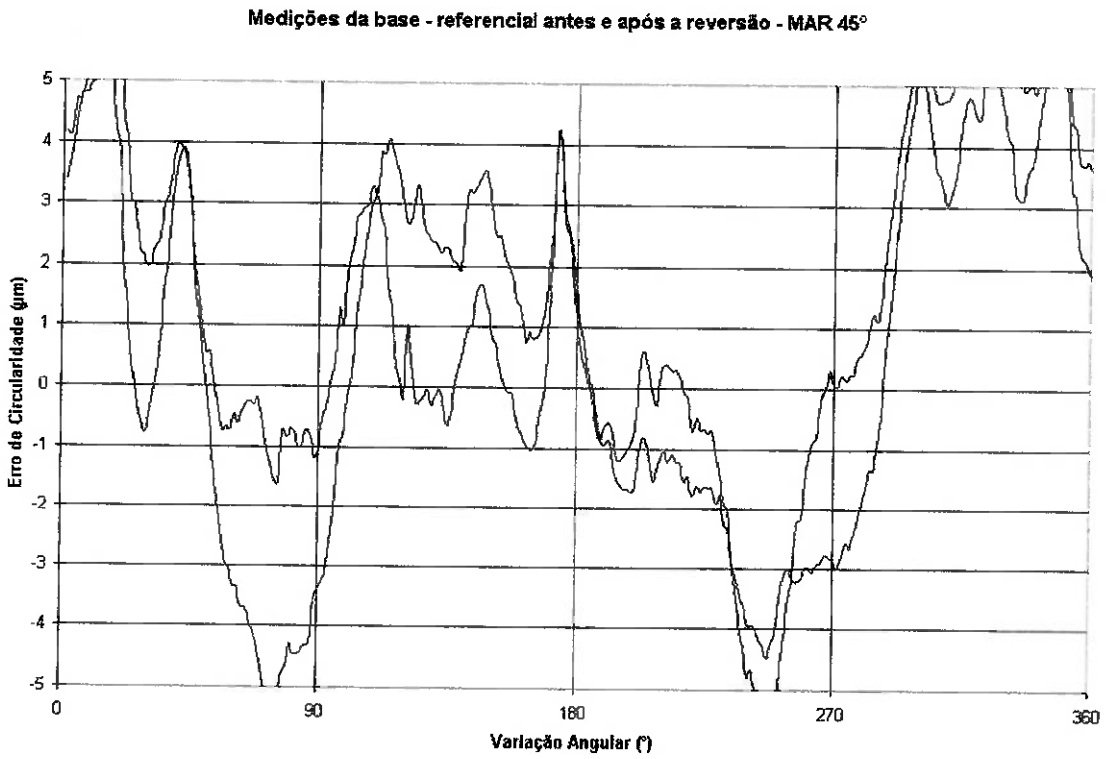


Fig. 10.4.2. Medições do referencial auxiliar antes e após a reversão já girado de 45° o corpo de prova.

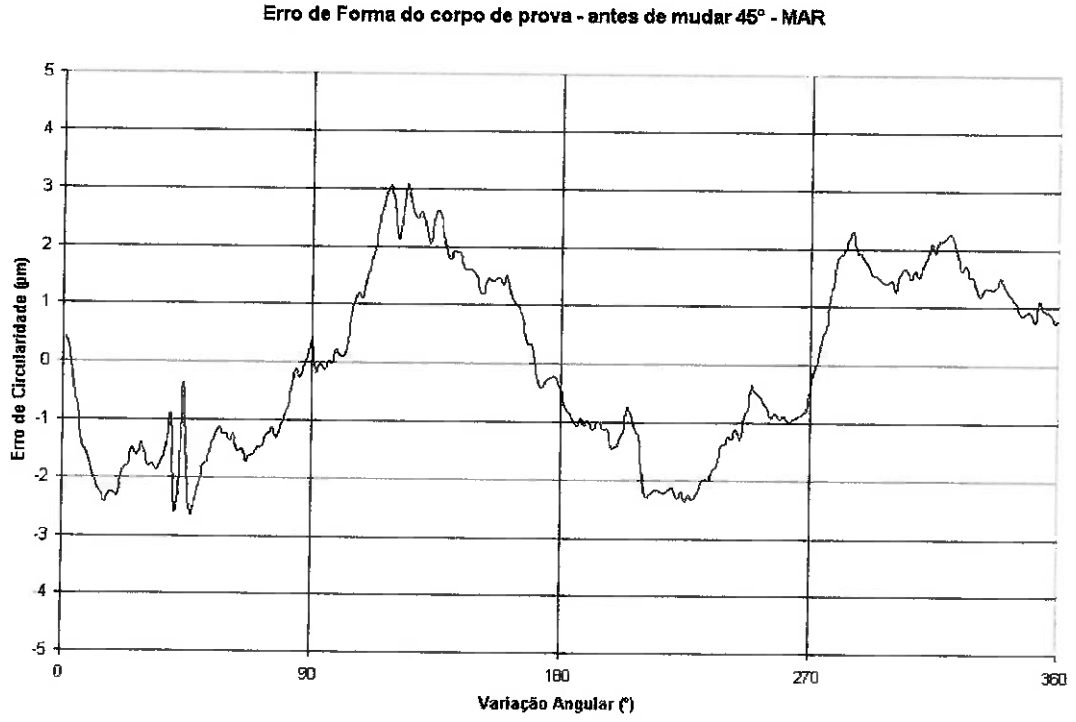


Fig. 10.4.3. Erro de Forma obtido pelo MAR com as curvas das Figuras 10.4.1 e 10.4.2.

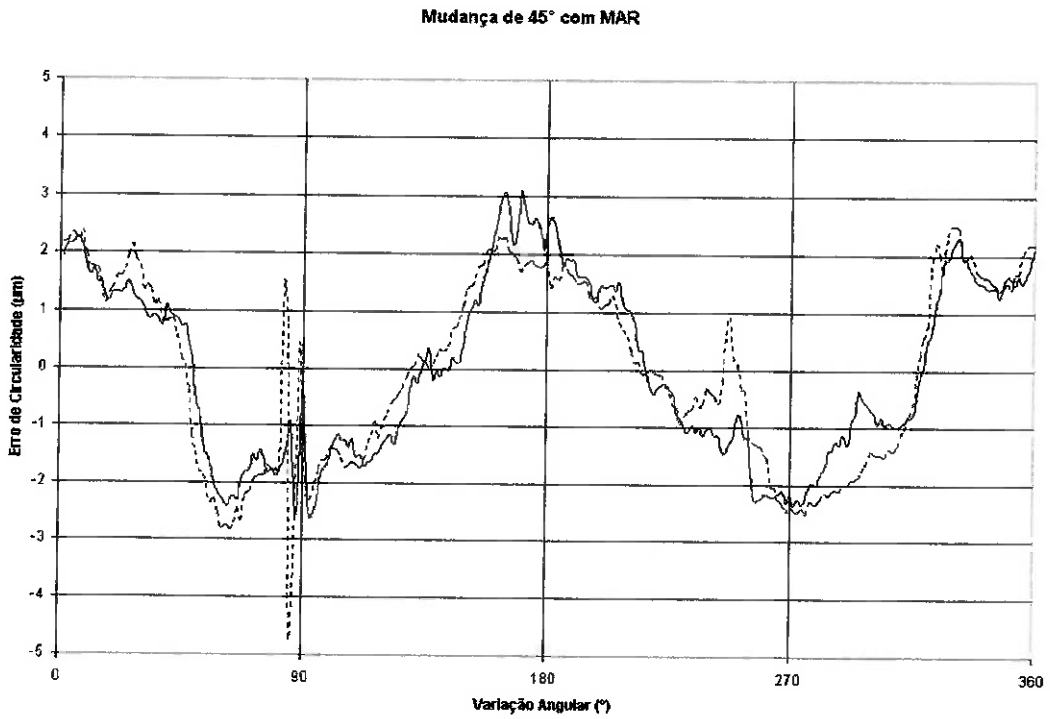


Fig. 10.4.4. Transladando a curva da Figura 10.4.3 em 45°, e comparando-a com uma curva obtida pelo MAR (figura 10.2.3)

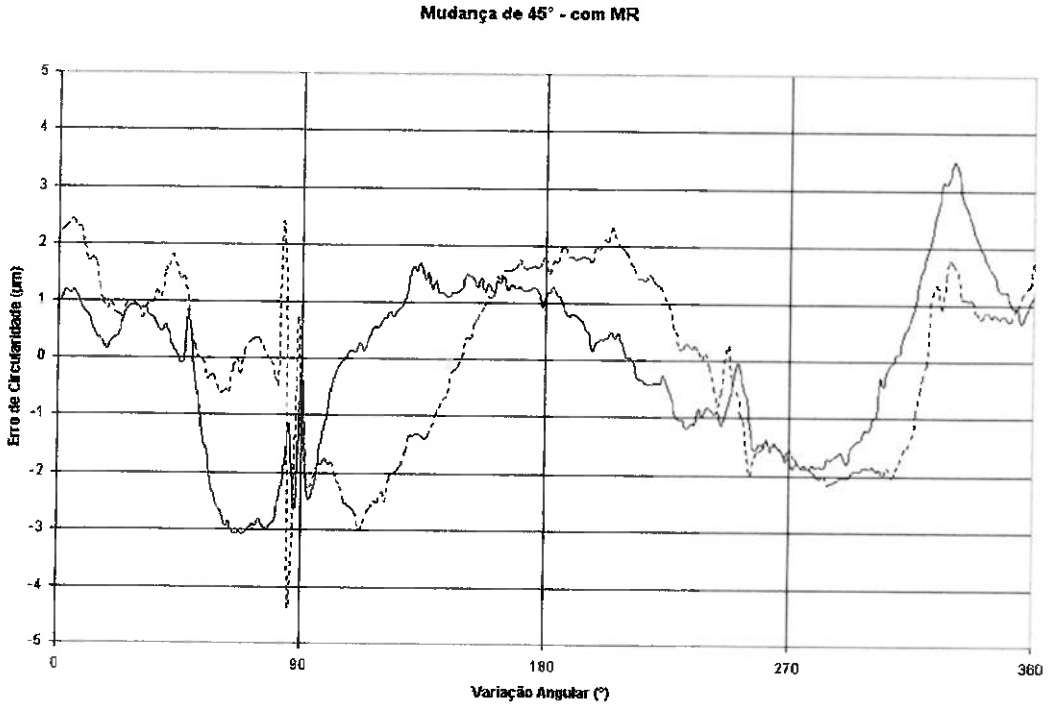


Fig. 10.4.5. Comparação dos erros de forma obtidos por MR

10.5. Indução de Erros de Movimento I

- Objetivos: Verificar qual método (MR ou MAR) é menos influenciado por falhas externas que são aplicadas através de pequenas "batidas" no eixo de rotação da bancada de medição. Deste modo, determina-se qual é o mais robusto contra leves "batidas" (pequenas variações).

- Método Utilizado: Aplicaram-se pequenas "batidas"/deslocamentos no eixo de rotação e consequentemente o valor da medição do sensor que mede o corpo de prova e o referencial auxiliar mudaram (figura 10.5.1). Os dados lidos podem ser vistos nas figuras 10.5.2 e 10.5.3 e os resultados obtidos estão nas figuras 10.5.4 (MR) e 10.5.5 (MAR), onde estão sendo feitos comparações com os dados obtidos anteriormente através de medição comum, sem "batidas".

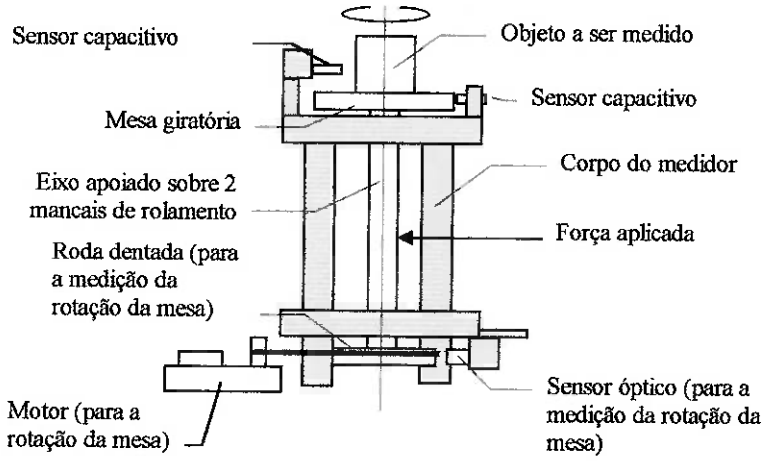


Fig. 10.5.1. Esquema de onde e como é aplicado cargas externas.

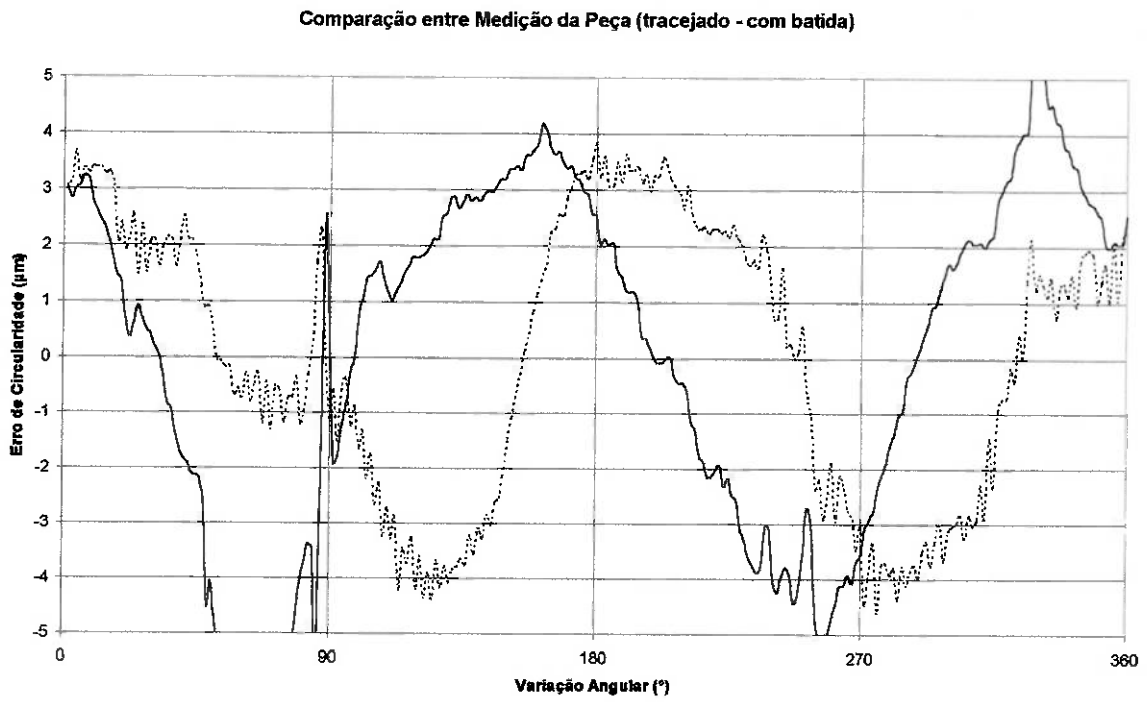


Fig. 10.5.2. Comparação da leitura dos sensores com e sem carga externa.

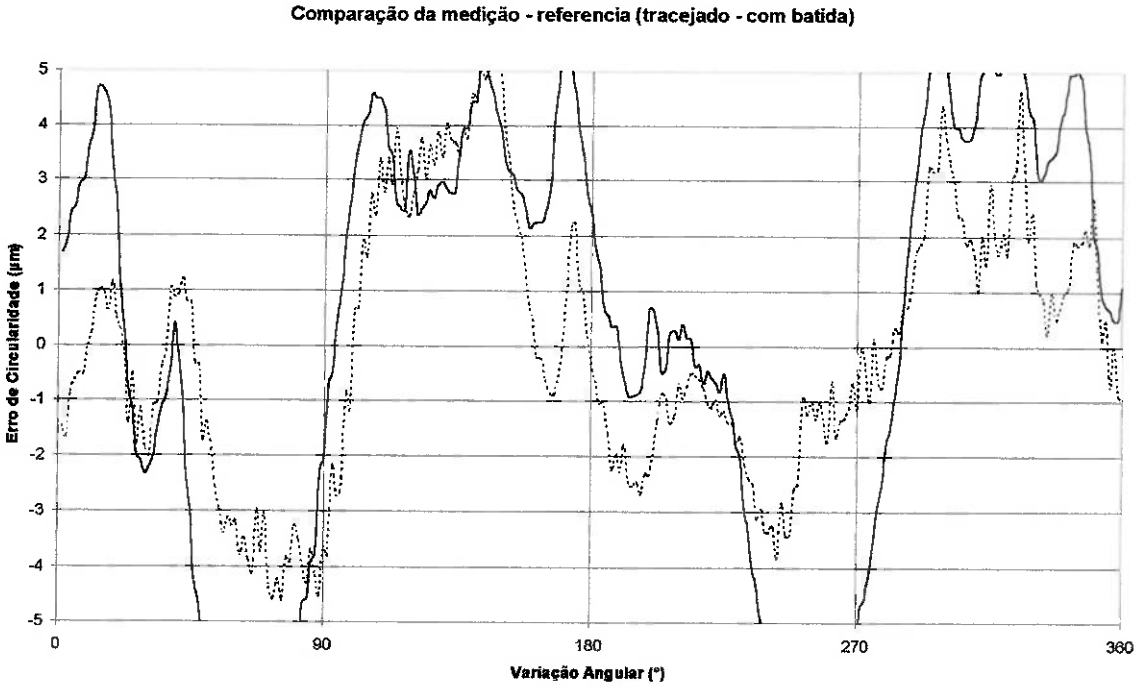


Fig. 10.5.3. Comparação entre as leituras do sensor com e sem carga externa.

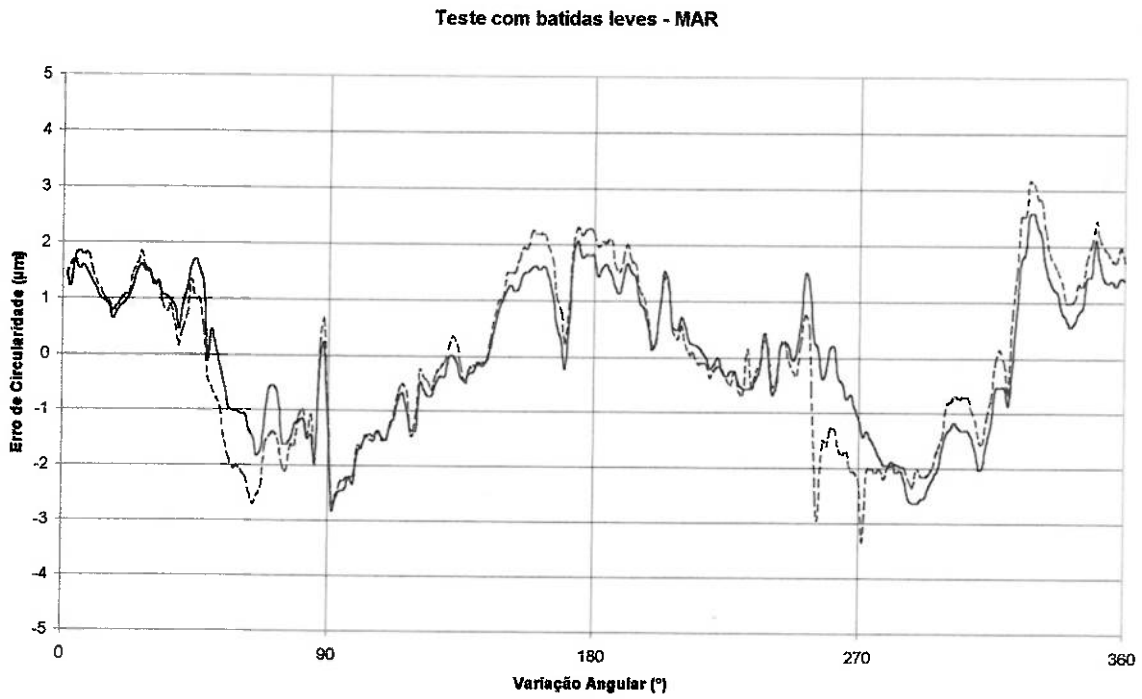


Fig. 10.5.4. Comparação entre erro de forma obtidos pelo MAR.

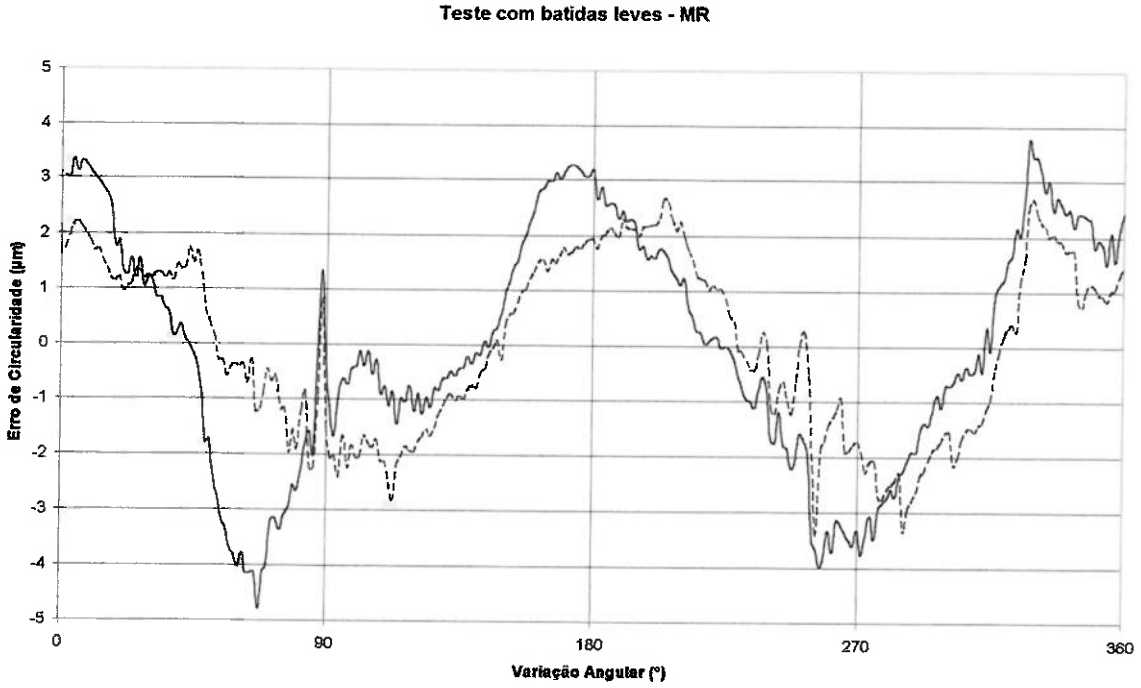


Fig. 10.5.5. Comparação dos erros de forma obtidos pelo MR.

- Fatos Observados: O gráfico que mostra o resultado do MAR(com "batidas") possui menor diferença comparado com outro gráfico obtido pelo MAR(sem "batidas") que a comparação do MR.
- Conclusões: O MAR mostrou-se mais robusto contra pequenos deslocamentos que podem ocorrer durante a medição, tornando os resultados de circularidade do corpo de prova mais confiáveis que os do MR.

10.6. Indução de Erros de Movimento II

- Objetivos: Verificar qual método (MR ou MAR) é menos influenciado por falhas externas que são aplicadas através de pesadas "batidas" no eixo de rotação da bancada de medição. Deste modo, determina-se qual é o mais robusto contra pesadas "batidas" (grandes variações).
- Método Utilizado: Aplicou-se grandes "batidas"/deslocamentos no eixo de rotação e consequentemente o valor da medição do sensor que mede o corpo de prova e o referencial auxiliar mudaram (figura 10.5.1). Os dados lidos podem ser vistos nas figuras 10.6.1, 10.6.2 e 10.6.3; os resultados obtidos estão nas figuras 10.6.4 (MR) e 10.6.5 (MAR), onde estão sendo feitos comparações com os dados obtidos anteriormente através de medição comum, sem "batidas". Note que neste ensaio, é possível notar claramente onde ocorreram as "batidas", ao contrário do ensaio anterior, em que não era muito nítido os locais de variação.

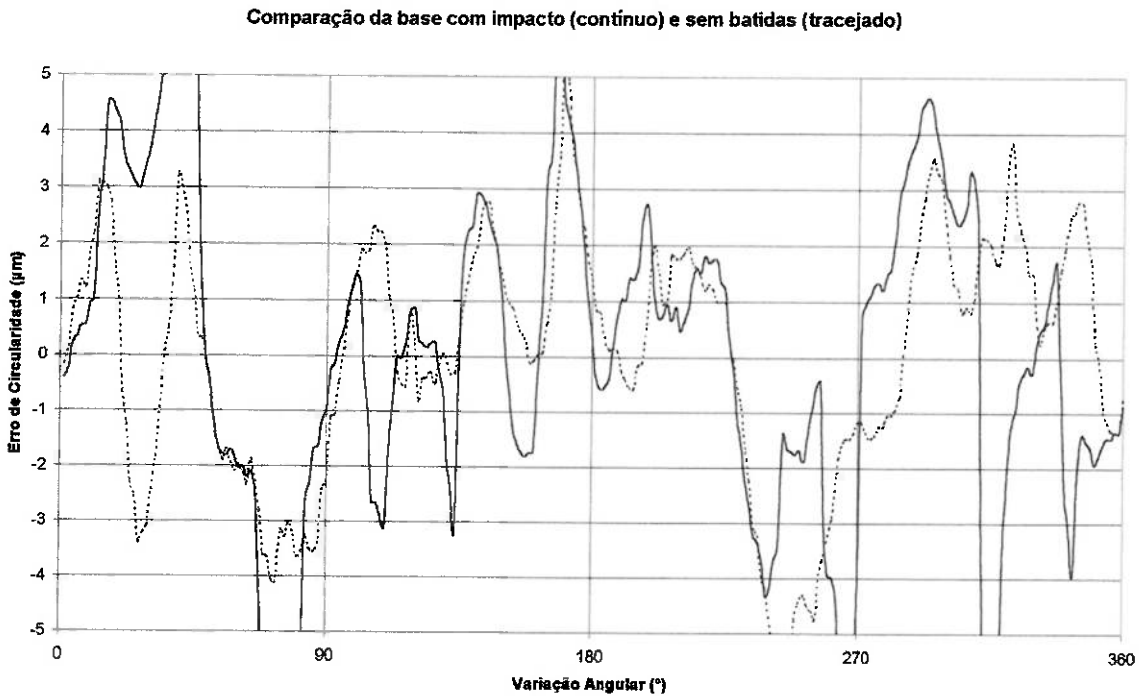


Fig. 10.6.1. Medição do referencial auxiliar com "batidas" pesadas aplicadas no eixo.

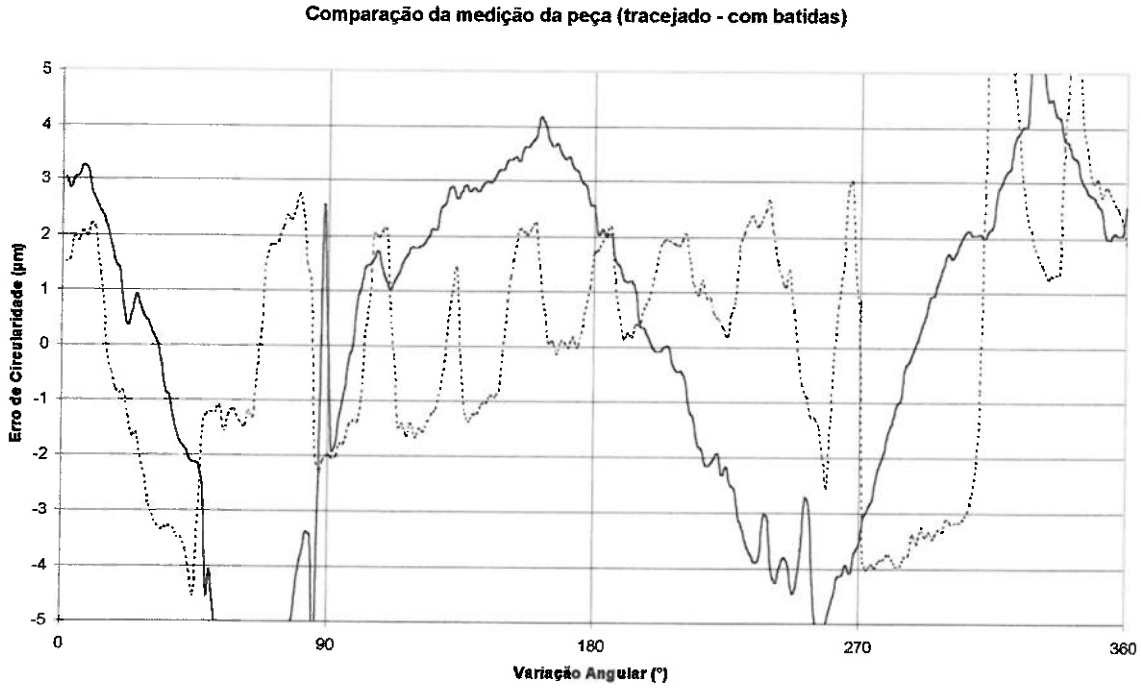


Fig. 10.6.2. Medição do corpo de prova com "batidas" pesadas aplicadas no eixo.

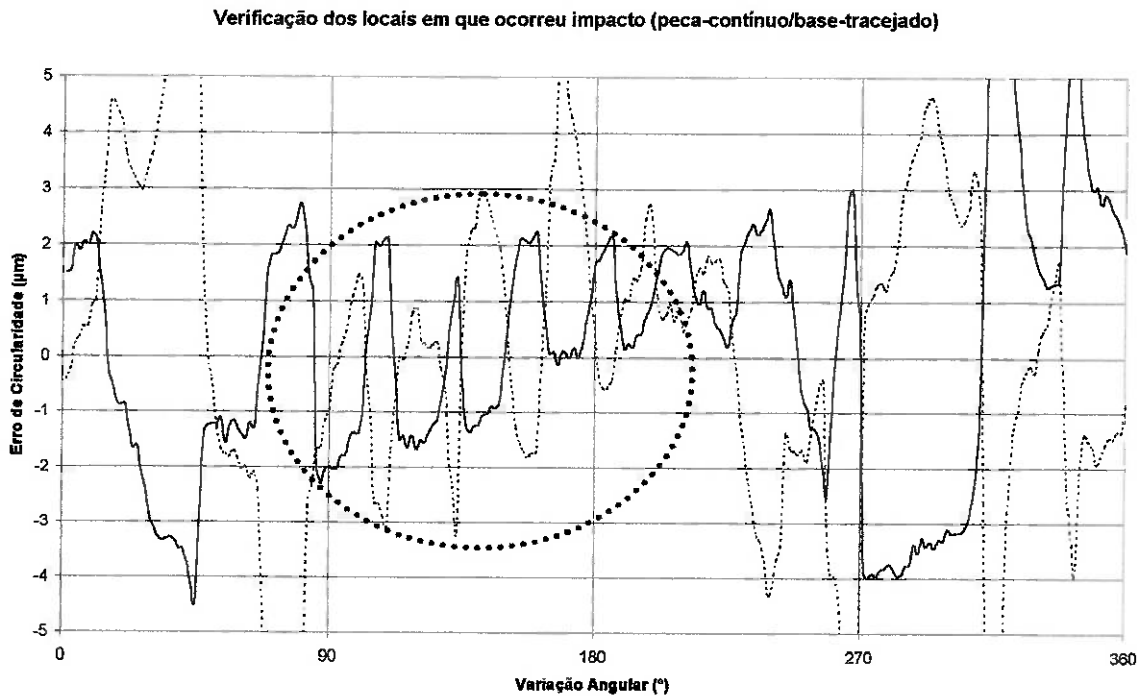


Fig. 10.6.3. Determinação dos instantes em que foram aplicadas as cargas, através da comparação da medição do corpo de prova e do referencial auxiliar.

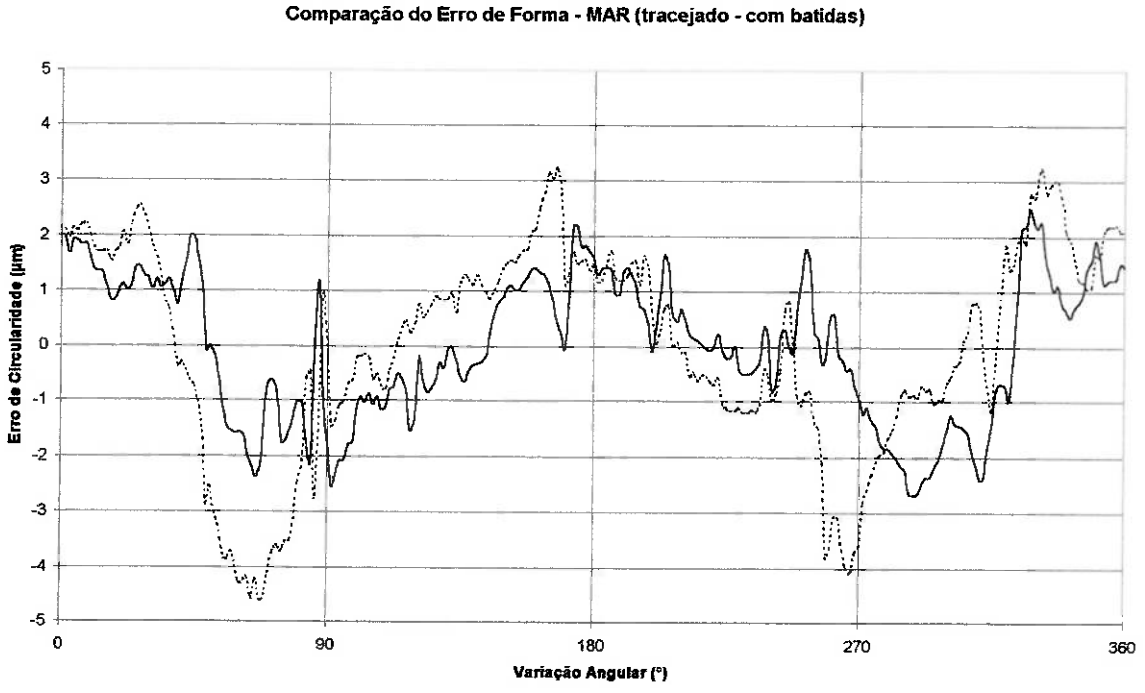


Fig. 10.6.4. Comparação do Erro de Forma obtido das curvas das fig. 6.3. com e sem "batidas".

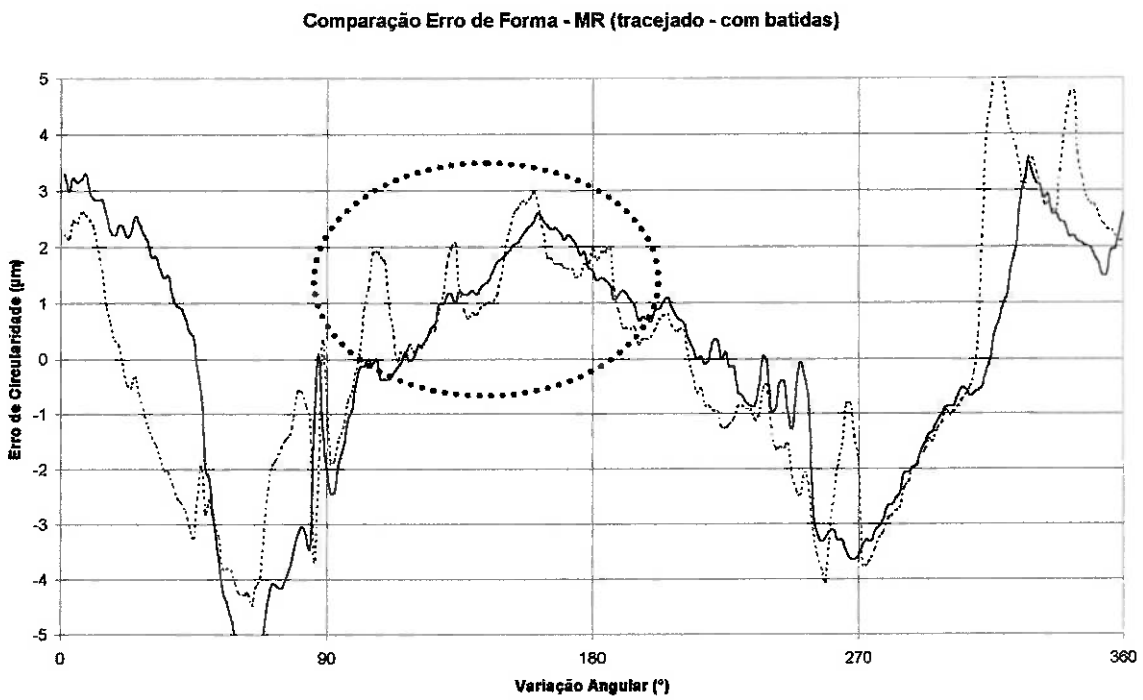


Fig. 10.6.5. Comparação do erro de forma obtido pelo MR das curvas da fig. 6.3. (com e sem "batidas").

- Fatos Observados: O erro de forma obtido pelo MR possui picos locais gerados pelas "batidas" aplicadas no eixo. O mesmo não ocorre com o MAR, que conseguiu reduzir a componente deste tipo de erro.
- Conclusões: O MAR mostrou-se mais robusto quanto a pesadas "batidas", porém, conforme o nível de deslocamento forçado, o resultado perde muito sua confiabilidade. Mas ainda é melhor que se fosse usado apenas o MR.

10.7. Medição Comparativa

- Objetivos: Verificar a validade dos resultados obtidos pelo MAR, fazendo-se a comparação do erro de forma com os dados lidos por sistema de medição encontrado comercialmente. Foi escolhido o medidor de coordenadas (MC) por estar disponível no departamento de Engenharia Mecatrônica e possuir alta precisão.

- Método Utilizado: Fez-se a medição no MC numa faixa próxima a faixa medida no MAR. Note que o sensor do MC é pontual, enquanto que foram utilizados sensores capacitivos para o MAR, que medem por região de superfície. O resultado da medição pelo MC

encontra-se na figura 10.7.1 e a comparação (após deslocamento angular, extração da excentricidade e valor médio) na figura 10.7.2.

- Fatos Observados: A comparação inicial (figura 10.7.2) difere muito do resultado do MAR. Filtrou-se os dois sinais, usando-se o Fast Fourrier Transform (FFT) para 10ª ordem. Houve considerável melhora na comparação, porém, após retirar-se a componente de 2ª ordem do erro de forma do MAR, a comparação ficou a mais próxima possível. Pode-se atribuir a retirada da componente de 2ª ordem devido a um possível desnível na mesa de medição do MAR comparado com a mesa de medição do MC (figura 10.7.3). A nova comparação entre MC, MAR e MR pode ser vista na figura 10.7.4.

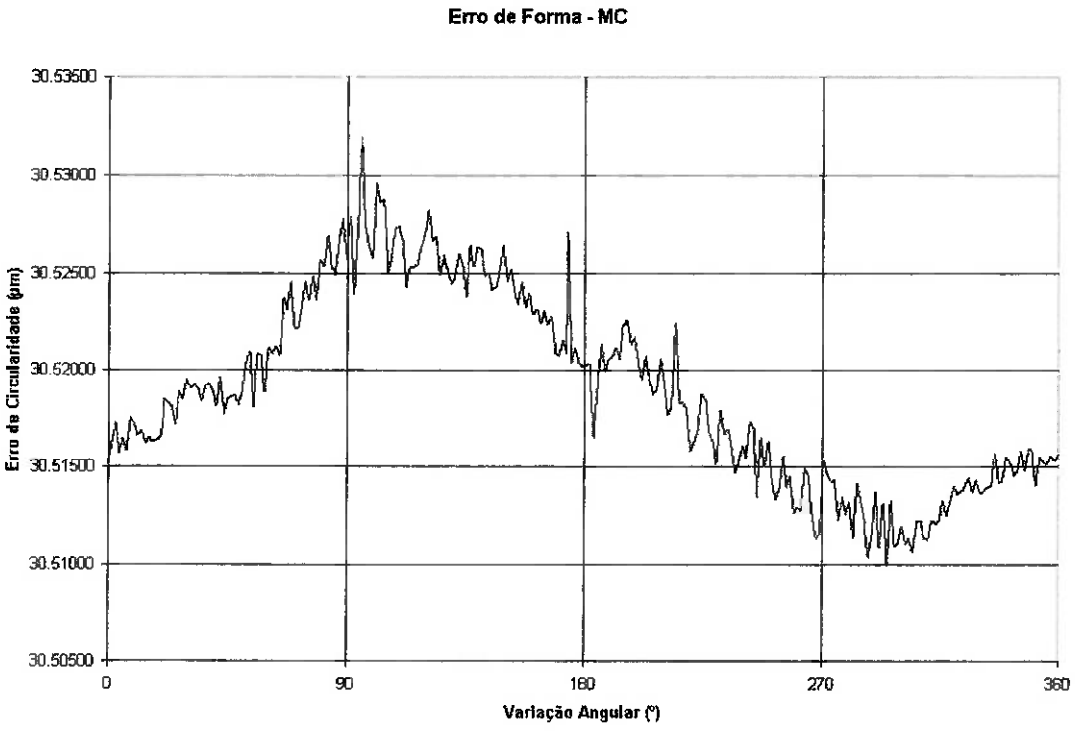


Fig. 10.7.1. Leitura obtida pelo MC para o corpo de prova utilizado.

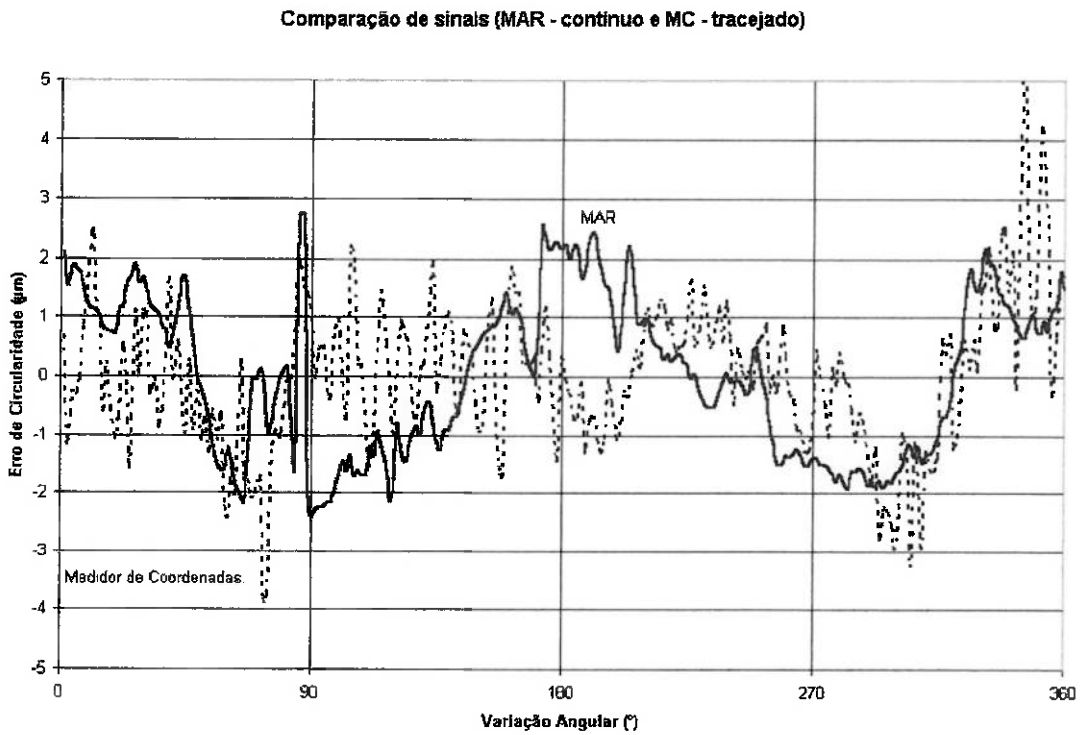


Fig. 10.7.2. Comparação dos erros de forma obtidos pelo MAR e MC (após extração do valor médio e da excentricidade).

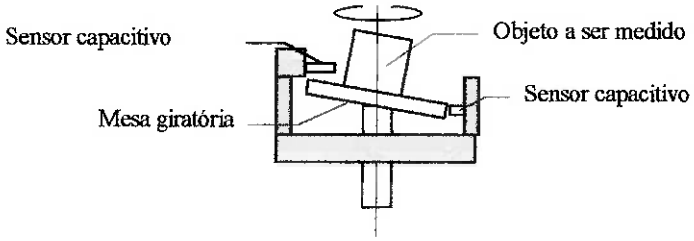


Fig. 10.7.3. O erro de perpendicularismo entre a mesa giratória e o eixo pode aumentar a componente de 2ª ordem da leitura do sensor.

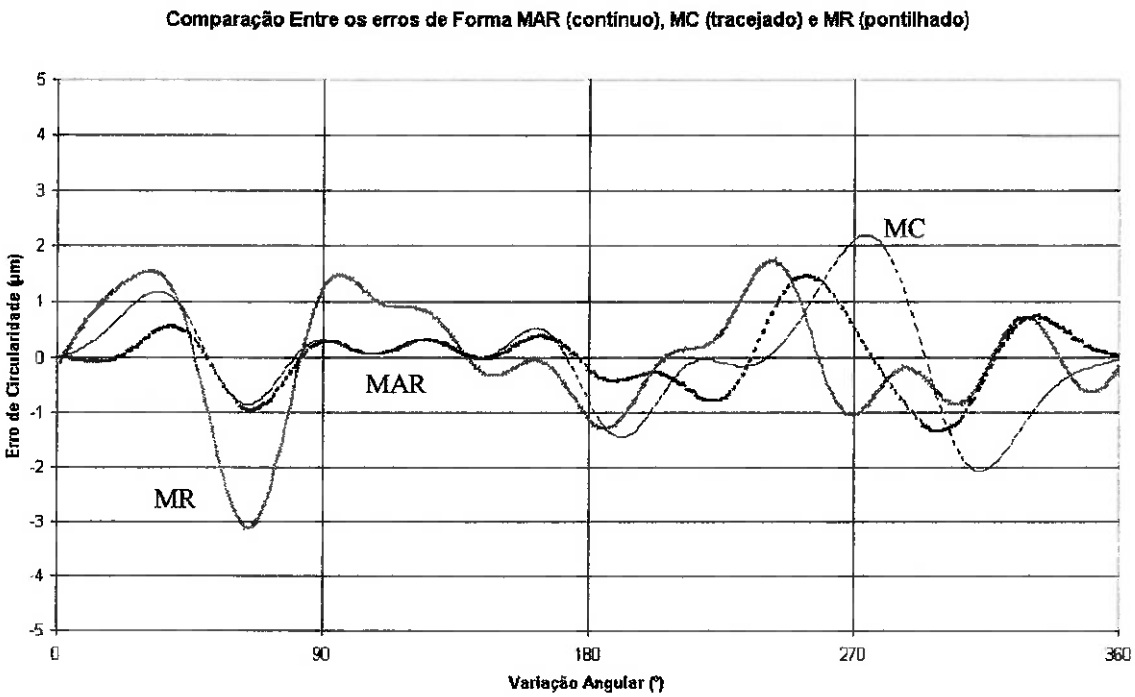


Fig. 10.7.4. Comparação entre os erros de forma obtidos pelo MAR, MR e MC (após filtragem e extração de valor médio e excentricidade).

- Conclusões: Após muita filtragem, pode-se reconstruir um sinal com parte dele parecido com o sinal do MC, utilizando-se o MAR. O resultado do MR está na mesma ordem de grandeza, mas mais diferenças na forma que o resultado do MAR. Isto indica que é possível a reconstrução do erro de forma pelo MAR, após novas adaptações e correções de erros. Um motivo para aceitação desta diferença, é a verificação de diferenças consideráveis ao se medir em faixas próximas a anterior (figura 10.7.5). Isto indica uma diferença de 1µm conforme a faixa que se está medindo com o MC.

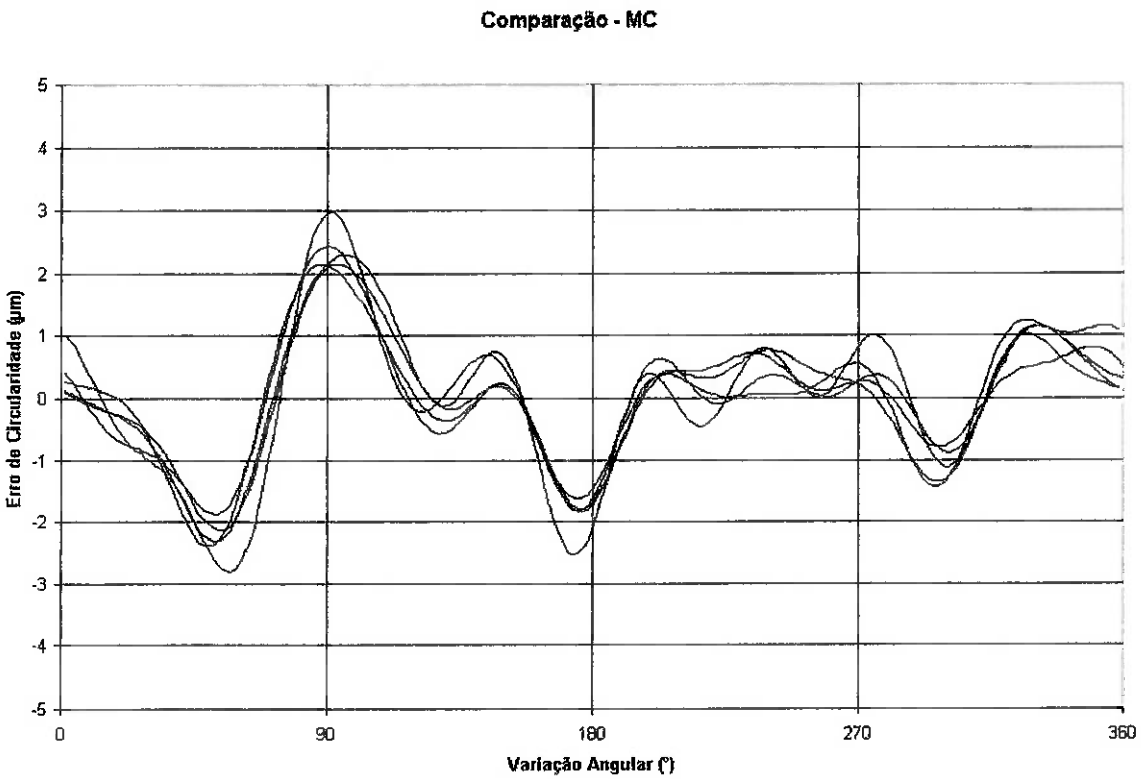


Fig. 10.7.5. Medições em várias faixas do corpo de prova no MC.

11. Discussão

Foi apresentado uma técnica para a obtenção do erro de forma de um corpo de prova, através da separação da medição lida do erro de movimento da bancada de medição.

Os testes aplicados demonstravam que o MAR possui maior robustez comparado ao MR. O erro de movimento da bancada de medição, conforme premissa inicial para o uso do MR, não pode ser considerado como possuidor de alta repetibilidade. Enquanto que o MAR não necessita desta premissa e os resultados indicam que é possível fazer a compensação desta falta de alta repetibilidade.

Os eventuais erros acidentais que possam ocorrer durante a medição podem acarretar a falhas na interpretação dos resultados do MR, enquanto que o MAR pode diminuí-los.

O erro causado pelo não perpendicularidade entre a mesa giratória e o eixo gera uma componente considerável, mas possível de ser reduzido facilmente através de compensação matemática. Assim como a excentricidade e o valor médio.

Devido a diferença entre os tipos de sensores usados (pontual para o MC e por área para o sensor de não contato), pode-se aceitar uma diferença entre os erros de forma obtidos pelos MAR e pelo MC. Porém, numa certa região, a reconstrução do erro de forma teve bons resultados, indicando que a aplicação do MAR está coerente e é viável de uso (conforme a precisão desejada).

Conseguiu-se a obtenção do erro de forma de objeto da ordem de 5-10 μm (figura 10.7.4), utilizando-se um sistema com precisão da ordem de 10 μm (figura 10.1.1 – máxima variação entre as medições), o que era o principal objetivo deste trabalho: provar que é possível obter medições do erro de forma de componentes cilíndricos com precisão superior à dos componentes utilizados no sistema. Pela simples medição do componente do sistema tem-se uma precisão de 10 μm. Após compensação via software e filtragem matemática, a nova precisão é de 2 μm (figura 10.7.4 – maior diferença com valor medido no MC e do MAR).

A Repetibilidade do Sistema de Medição com o uso do M.A.R. da ordem de 3 μm sugere que a precisão de medição do erro de forma também seja da ordem de 3 μm.

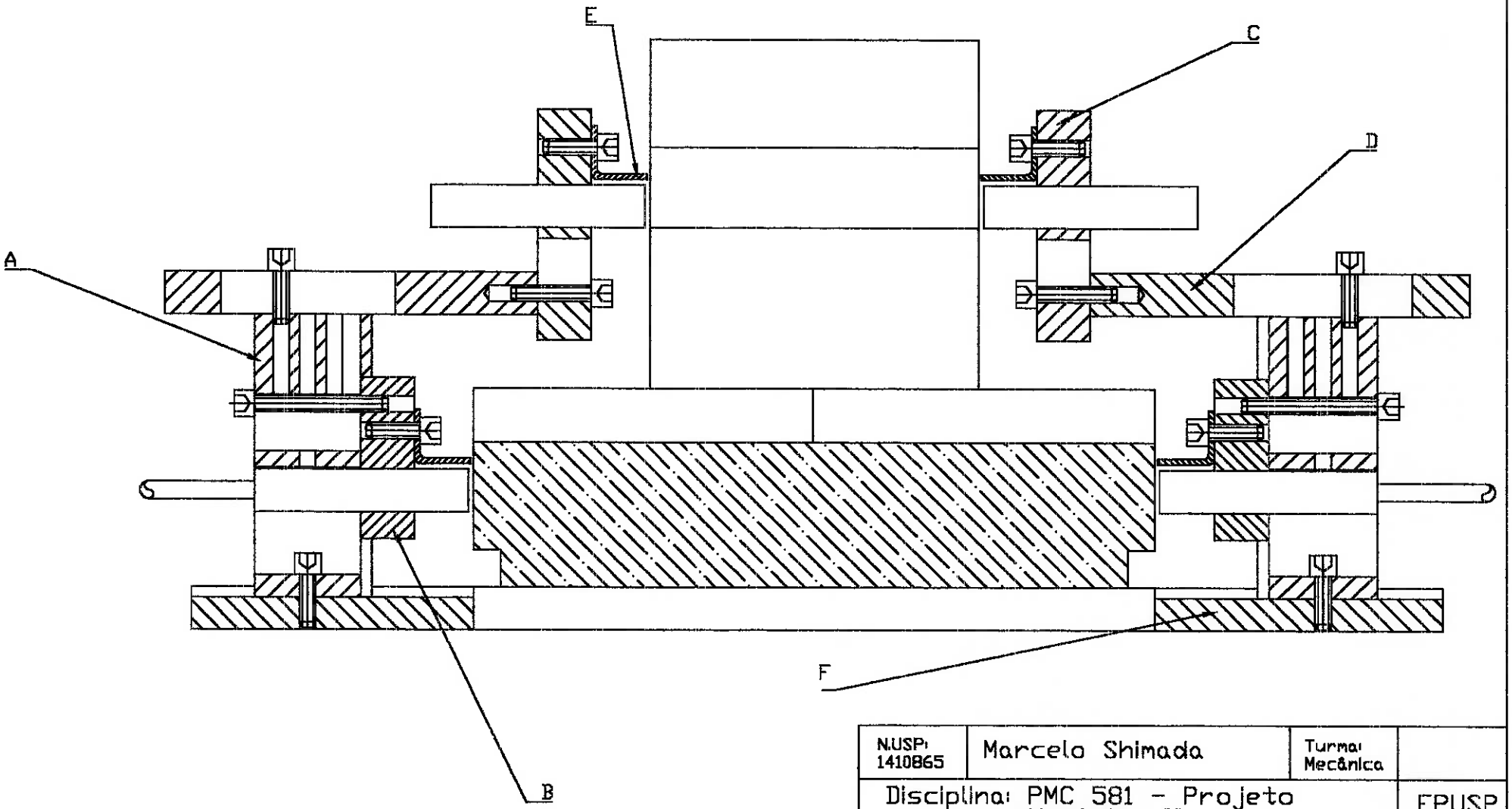
12. Conclusões

Foi desenvolvido um sistema de Medição de Circularidade em que seguiu-se as seguintes etapas para o seu término:

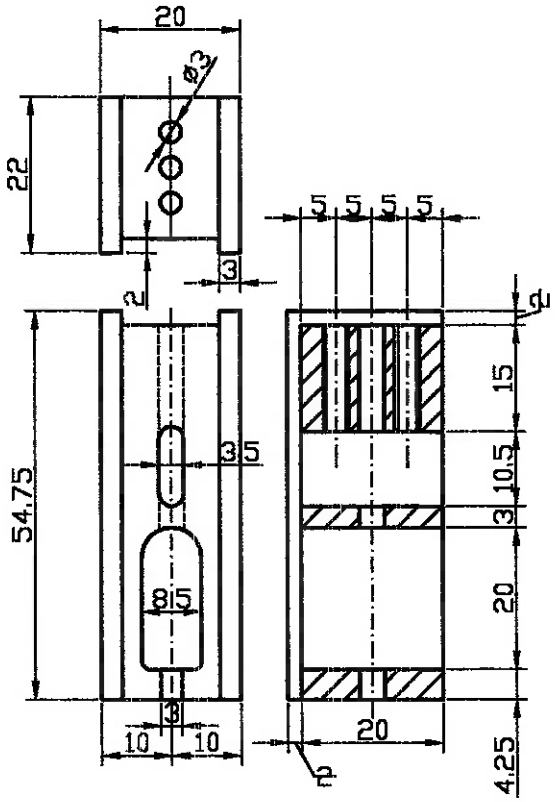
- utilização de uma bancada de medição de um trabalho anterior;
- substituição dos sensores de Medição;
- Montagem de uma nova fixação para os novos sensores de Medição;
- Montagem de Encoders;
- Montagem de Cabos/Conexões;
- Montagem de um circuito eletrônico;
- Desenvolvimento de um software para coleta, processamento e armazenamento de dados;
- Implementação de várias rotinas para compensação de erros (Ex.: M.R., M.A.R., excentricidade);

O trabalho apresentado exigiu conhecimentos de várias áreas técnicas (metrologia, mecânica, eletrônica e programação). O software implementado parece ter a maior parte das funções necessárias para a medição de circularidade.

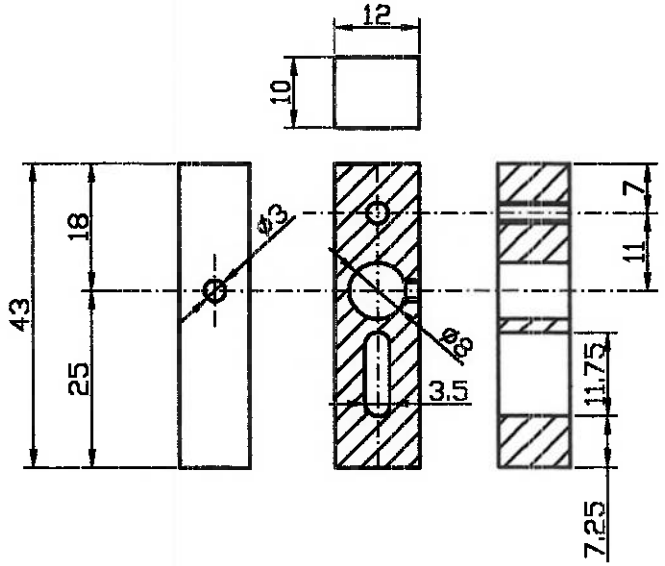
Os resultados encontrados confirmam o objetivo inicial de provar que um sistema de medição pode ter sua precisão/Repetibilidade aumentada com o uso de compensação matemática por software.



N.USP: 1410865	Marcelo Shimada	Turma: Mecânica	
Disciplina: PMC 581 - Projeto Mecânico II			EPUSP
	Anexo A- Desenho de Conjunto da Nova Fixação	Data: 06/06/99	



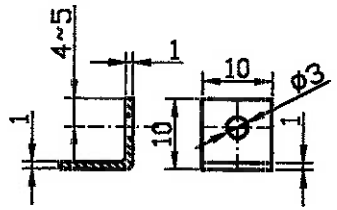
Peça A



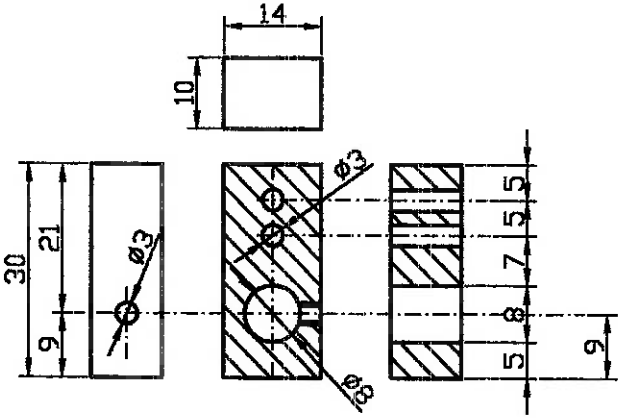
Peça B

Obs.: É necessário duas peças de cada tipo

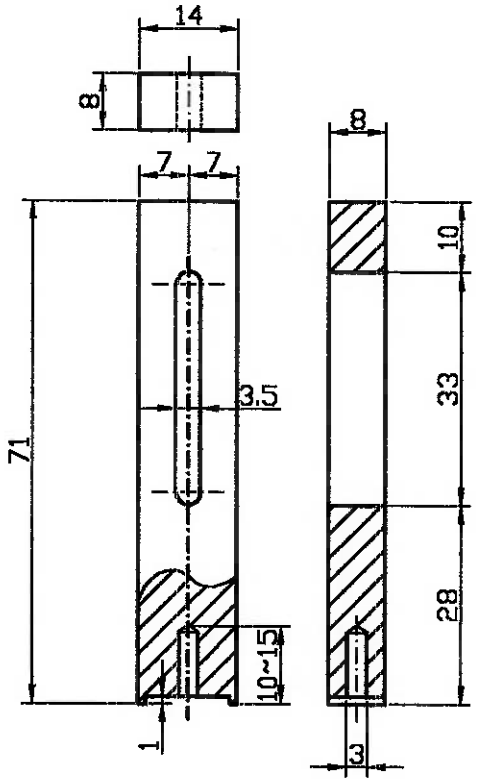
N.USP: 1410865	Marcelo Shimada	Turma: Mecânica	
Disciplina: PMC 581 – Projeto Mecânico II			EPUSP
Escala 1 : 1	Anexo A: Componentes da Nova Fixação	Data: 06/06/99	



Peça E



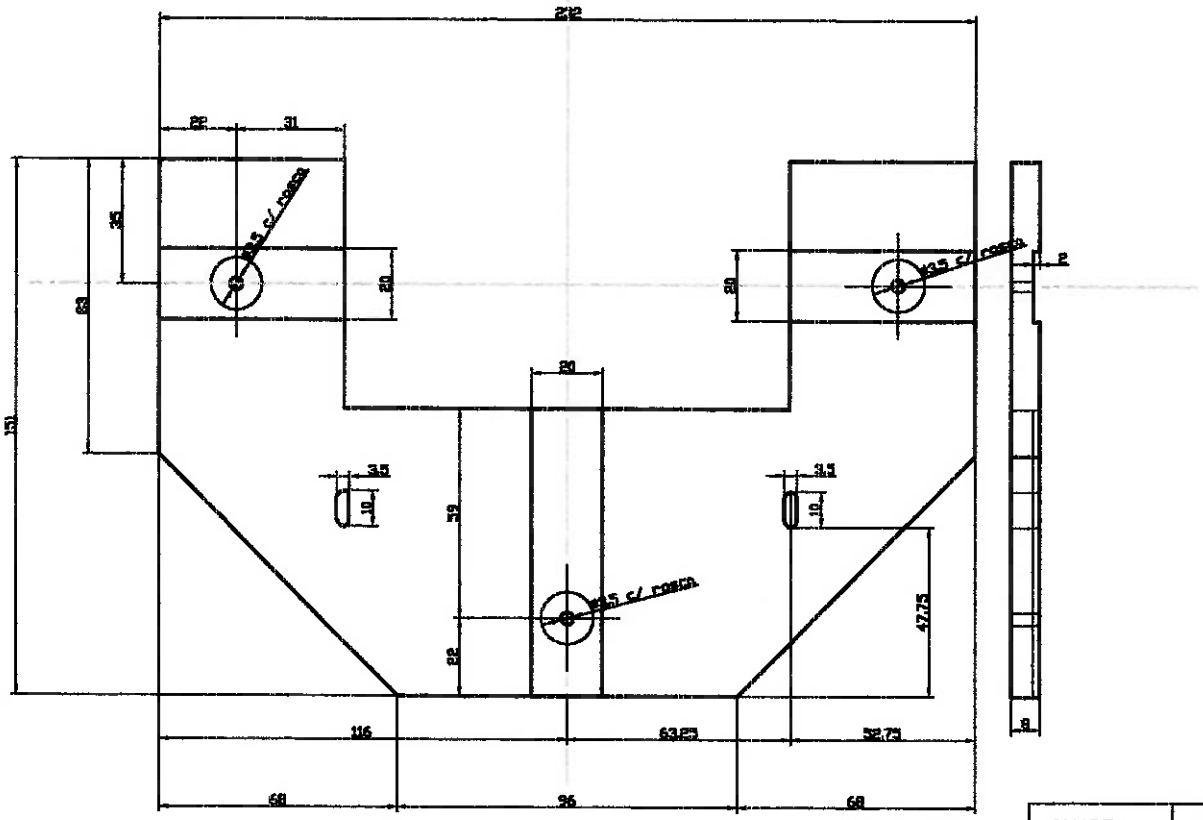
Peça C



Peça D

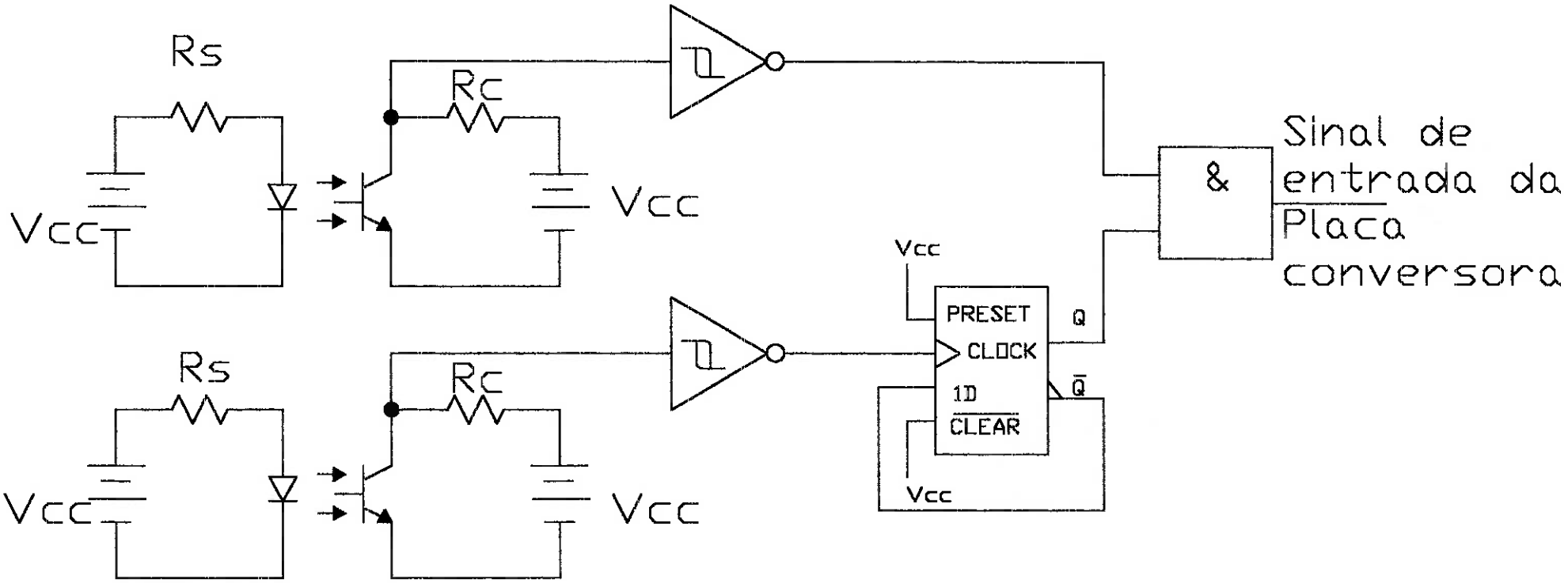
Obs.: É necessário duas peças de cada tipo

N.USP: 1410865	Marcelo Shimada	Turma: Mecânica	
Disciplina: PMC 581 - Projeto Mecânico II			EPUSP
Escala 1 : 1	Anexo A: Componentes da Nova Fixação	Data: 06/06/99	



Peça F

N.USP: 1410865	Marcelo Shimada	Turma: Mecânica	
Disciplina: PMC 581 - Projeto Mecânico II			EPUSP
Escala 1 : 2	Anexo A: Componentes da Nova Fixação	Data: 06/06/99	



N.USP: 1410865	Marcelo Shimada	Turma: Mecânica	
Disciplina: PMC 581 - Projeto Mecânico II			EPUSP
	Anexo B: Descrição da parte elétrica (interface entre sensor rotativo e placa A/D)	Data: 06/06/99	

Anexo C – Software

Linguagem

A linguagem escolhida para o software inicialmente foi o Pascal. O motivo da escolha desta linguagem foi a existência de programas exemplos no manual da placa de aquisição de dados CAD 1026 e sua simplicidade. Em seguida o software foi convertido para Delphi. A escolha por um software para ambiente Windows foi determinada pela facilidade de se converter o software em Pascal para Linguagem Visual Delphi.

Estrutura de Dados

Os dados coletados são armazenados na memória em um vetor cujo índice varia de 1 a n (256). São do tipo single (de 4 bytes) por serem ponto flutuante de menor quantidade de bytes encontrado no Delphi. Não foi escolhida o tipo inteiro pois previa-se o uso de filtros e manipulações matemáticos que iriam geram números que precisariam de casa após a virgula.

Arquivo de Saída/Entrada em disco

O armazenamento dos dados em disco é feito através do seguinte formato de arquivo tipo texto (Ex.: 1.clk):

Nome: nome_do_arquivo -> Primeira linha indica o nome completo do arquivo gerado (incluindo caminho do diretório)
1.00000 <- as 256 linhas seguintes são os valores coletados com uma cinco casas decimais de precisão
?.?????
5.00000 <- A última linha é o valor da escala utilizada. Ex.: 5 indica que o valores acima devem ser multiplicados por 5 -> 1.00000 => 5.00000 µm

Existe ainda o arquivo de configuração 'wclk.ini' que possui o seguinte formato:

// Canal do encoder
4
// Canal 1 de Leitura
12
// Canal 2 de Leitura
12
// Raio do modo gráfico
30
// Amplificação do modo gráfico
10
// Escala do sensor1
5.000
// Escala do sensor2
5.000
Valor do Endereço de base (Este valor não muda)

720

Variáveis e constante principais do software:

- n : quantidade de pontos máximo (256 devido a engrenagem na base inferior da bancada possuir 128 ranhuras) ;
- valores : vetor de 1 a n formado por single. Principal estrutura de dados utilizada para armazenar os valores coletados e processados;
- arquivo : formato de arquivo utilizado no armazenamento de dados em disco (texto);
- name_file : nome do arquivo para armazenamento em disco;
- i,j,k : contadores mais utilizados;
- st : string para uso geral;
- escala_sensor1/2, escala_amp1/2 : valores das escalas utilizadas para os dois sensores a amplificadores para quando fossem necessários correções;
- raio : Raio base para geração de gráficos;
- base: Endereço de base da placa A/D. Neste software decidiu-se torná-lo fixo em 720.
- amplia : fator de ampliação da leitura no modo gráfico.
- deltax/y : usados para armazenar os valores na direção x e y da medição.
- Tabela_repete : matriz de 1 a 14 formado por vetores de 1 a 256. É utilizado somente para armazenar os dados que serão mostrados no modo de comparação de 14 gráficos.
- Valor : usado no modo de comparação de 14 gráficos para indicar quantos gráficos serão mostrados simultaneamente;
- Canal1, canal2 , encoder : canais de leitura da Placa A/D;
- Cores : cores que são usadas no modo de comparação de 14 gráficos;
- H1, H2 : marcadores de tempo. São usados para se verificar o fim do processo de coleta de dados;
- Vcos, Vsin: usados para armazenar os valores de cosseno e seno entre j*360°/256 com j variando de 1 a 256. São muito utilizados e portanto para poupar tempo de cálculo foram armazenados em um vetor;

Principais Funções e Procedimentos Utilizados no Software

- St2byte, byte2st, single2st, ssingle2st, int2st, st2single: convertem valores numéricos em caracteres que podem ser mostrados na tela e vice-versa;
- Botao_carregar : para ajudar o usuário na escolha dos arquivos de armazenamento, este procedimento abre a janela para abertura de arquivo e obtém o nome do arquivo desejado;
- Carregar : realiza a abertura do arquivo e coloca seu conteúdo na memória (mais precisamente no vetor dados);
- Botao_gravar: para ajudar o usuário na escolha dos arquivos de armazenamento, este procedimento abre a janela para gravação de arquivo e obtém o nome do arquivo desejado;
- Gravar(dados, escala) : realiza a gravação do arquivo com os dados do vetor e da escala utilizada passados como parâmetro;

- Raio_medio(dados) : calcula o valor médio do vetor passado como parâmetro;
- Centro(dados) : obtém o centro LSC do vetor passado como parâmetro;
- OutPort, InPort: Para se ter acesso as portas utilizando-se o Delphi, foi necessário o uso de pequenas rotinas em assembler; Estas rotinas realizam a escrita e leitura da porta passada como parâmetro;
- Falso_espera0, falso_espera, falso_diferente : são rotinas para aquisição sincronizada com o sinal externo. Esperam até que leitura do canal do encoder atinja certo nível de voltagem;
- Leitura (dados, canal) : realiza a leitura do canal e coloca o valor lido no vetor dados e incrementa o contador 'i' que é global;
- Timer : contador de tempo em milisegundos;
- Atualiza_tabela : existe uma tabela que aparece no canto direito. Este procedimento altera os dados, atualizando-os;
- Limpar_dados : limpa o vetor principal 'dados';
- Ad : faz a leitura do canal desejado e obtém o valor lido;
- Corrige(dados) : realiza a filtragem do valor médio e da excentricidade do vetor 'dados';

Como está estrutura o software

Como o software foi feito em Delphi, ele possui várias unidades e formulários. Muitos dos objetos utilizados foram deixados com o nome padrão (Ex.: Edit1) pois são facilmente visualizados e compreendidos.

As Unidades

- Alterar_config : permite o acesso a alteração de certos valores (Ex.: canais de leitura) e realiza sua gravação no arquivo texto 'wclk.ini';
- Apresentacao : não possui rotinas;
- Clk : unidade principal, contém várias rotinas, procedimentos e funções;
- Col_repete : possui rotinas para realizar várias coletas seguidas;
- Coleta_mar : realiza a aquisição de dados para o MAR (2 canais de leitura);
- Coleta_unit : realiza a aquisição de dados para o MR (1 canal de leitura);
- Fft : possui os algoritmos para FFT e IFFT;
- Filtro_unit : utiliza um filtro Butterworth de primeira ordem;
- Inicio_tf : não possui rotinas;
- L_teste : realiza a leitura de um canal no instante (não precisa do encoder);
- Linear : realiza gráfico linear;
- Medio_unit : realiza extração de valor médio e excentricidade;
- Polar : visualização de gráfico linear;
- Repete : mostra no máximo 14 gráficos ao mesmo tempo;
- Ver_config : não possui rotinas;

Os Formulários

- Circularidade: estão os principais comandos e funções (Ex.: MR, MAR, repetibilidade);
- Coleta: instruções para a coleta de dados para o MR;
- Coleta_repete: instruções para a coleta de dados para repetibilidade;
- Config: serve para alterar configurações do software (Ex.: canais de leitura);
- Config_atual: mostra a configuração atual do software;
- Filtro: filtragem simples usando Butterworth de primeira ordem;
- Form_fft: filtragem utilizando FFT;
- Form_mar: instruções para a coleta de dados para o MAR;
- Grafico_linear: Visualização do gráfico linear;
- Grafico_polar: Visualização do gráfico polar;
- Grafico_repete: Visualização do gráfico para repetibilidade;
- Inicio: Propaganda do software;
- Leitura_teste: Coleta de dados simples de um canal;
- Medio: instruções para retirar valor médio e excentricidade;
- Sobre: Mais propaganda do software;

Diagramas do funcionamento básico programa de aquisição de dados

Programa Principal

"Aquisição sincronizada de dados"
"Interpretação/Manipulação de dados"
"Gravar/Carregar dados"
"Ver resultados"

Aquisição sincronizada de dados

Esperar borda de subida do sinal de controle externo, que vem dos Encoders ópticos	
Repetir:	
	"aquisição de dados"
	esperar borda de descida do sinal de controle externo
	"aquisição de dados"
	esperar borda de subida do sinal de controle externo

Até que o tempo em que o sinal de controle permanece em 0V for maior que 0.5s

Aquisição de dados

Enviar informação para a placa conversora A/D para fazer leitura e conversão dos dados
Esperar que a conversão esteja completa para ter dado estável
Guardar dado obtido na memória

Interpretação/Manipulação de dados

Pegar dados obtidos na aquisição
Extrair média da curva gerada
Extrair erro de centralização da curva gerada (por fft)
Aplicar método avançado de medição (após ter dados suficientes)
Guardar: circularidade do corpo de prova e erro de movimento (antes e após a reversão)

Aplicação do Método da Reversão

Para i=1 até 256 faça:	
	Resultado[i]<=dados_antes_reversão[i] - dados_depois_reversão[i]; */ Cálculo do erro de movimento;
Gravar resultado;	
Para i=1 até 256 faça:	
	Resultado[i]<=dados_antes_reversão[i] + dados_depois_reversão[i]; */ Cálculo do erro de forma do corpo de prova;
Gravar resultado;	

Aplicação do Método Aprimorado da Reversão

Para i=1 até 256 faça:	
	Resultado[i]<=((dados_corpo_de_prova_antes_reversão[i] - dados_referencia_antes_reversão[i]) + (dados_corpo_de_prova_depois_reversão[i + dados_referencia_depois_reversão[i]))/2; */ Cálculo do erro de forma

Gravar resultado;	
Para i=1 até 256 faça:	
	Resultado[i]<=((dados_corpo_de_prova_antes_reversão[i] + dados_referencia_antes_reversão[i]) - (dados_corpo_de_prova_depois_reversão[i] + dados_referencia_depois_reversão[i])/2; */Erro de movimento antes da reversão;
Gravar resultado;	
Para i=1 até 256 faça:	
	Resultado[i]<=((dados_corpo_de_prova_depois_reversão[i] + dados_referencia_antes_reversão[i]) -(dados_corpo_de_prova_antes_reversão[i] + dados_referencia_depois_reversão[i])/2; */Erro de movimento depois da reversão;
Gravar resultado;	

Gravar/Carregar dados

Obter nome do arquivo
Preparar arquivo para leitura/escrita
Realizar leitura/escrita com os dados da memória

Ver Resultados

Traçar gráfico horizontal com os dados da memória
Traçar gráfico polar com os dados da memória
Obter maior e menor diâmetro possível com os pontos obtidos
Obter a diferença entre o maior e menor raio

```
unit clk;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, Grids, ComCtrls,
ExtCtrls, Buttons;

type
  TCircularidade = class(TForm)
    MainMenu1: TMainMenu;
    Arquivo1: TMenuItem;
    Abrir1: TMenuItem;
    Novo1: TMenuItem;
    N1: TMenuItem;
    Salvar1: TMenuItem;
    N2: TMenuItem;
    Sair1: TMenuItem;
    Editar1: TMenuItem;
    Recortar1: TMenuItem;
    Copiar1: TMenuItem;
    Colar1: TMenuItem;
    SelecionarTudo1: TMenuItem;
    Memo: TMemo;
    Ajuda1: TMenuItem;
    Sobre1: TMenuItem;
    Opes1: TMenuItem;
    Fontes1: TMenuItem;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    tabela: TStringGrid;
    Propriedades1: TMenuItem;
    AlterarConfiguracao1: TMenuItem;
    VerConfiguraoAtual1: TMenuItem;
    PageControl1: TPageControl;
    Repetibilidade: TTabSheet;
    MR: TTabSheet;
    MAR: TTabSheet;
    Grfico1: TMenuItem;
    GrficoPolar1: TMenuItem;
    Aquisio1: TMenuItem;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    Label4: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    edit2: TEdit;
    Label5: TLabel;
    Label6: TLabel;
    Edit3: TEdit;
    Edit4: TEdit;
    GroupBox3: TGroupBox;
    Label2: TLabel;
    Edit6: TEdit;
    Button6: TButton;
    Label7: TLabel;
    Label8: TLabel;
    Edit7: TEdit;
    Edit8: TEdit;
    Button7: TButton;
    Button8: TButton;
    Button4: TButton;
    Button3: TButton;
    Button1: TButton;
    Button2: TButton;
    Button9: TButton;
    GroupBox4: TGroupBox;
    Label9: TLabel;
    Label10: TLabel;
    Edit9: TEdit;
    Edit10: TEdit;
    Button10: TButton;
    Button11: TButton;
    GroupBox5: TGroupBox;
    Label11: TLabel;
    Label12: TLabel;
    Edit11: TEdit;
    Edit12: TEdit;
    Button12: TButton;
    Button13: TButton;
    Button16: TButton;
    AquisioMR1: TMenuItem;
    AquisioMAR1: TMenuItem;
    Panel1: TPanel;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    SpeedButton4: TSpeedButton;
    RetirarExcentricidade1: TMenuItem;
    RetirarValorMdio1: TMenuItem;
    Label1: TLabel;
    Edit31: TEdit;
    Edit21: TEdit;
    Button31: TButton;
    Button21: TButton;
    CheckBox21: TCheckBox;
    Button15: TButton;
    CheckBox31: TCheckBox;
    CheckBox22: TCheckBox;
    CheckBox24: TCheckBox;
    CheckBox25: TCheckBox;
    CheckBox27: TCheckBox;
    CheckBox32: TCheckBox;
    CheckBox33: TCheckBox;
    CheckBox34: TCheckBox;
    CheckBox35: TCheckBox;
    CheckBox36: TCheckBox;
    CheckBox37: TCheckBox;
    CheckBox23: TCheckBox;
    CheckBox26: TCheckBox;
    Edit22: TEdit;
    Edit23: TEdit;
    Edit24: TEdit;
    Edit25: TEdit;
    Edit26: TEdit;
    Edit27: TEdit;
    Button23: TButton;
    Button24: TButton;
    Button25: TButton;
    Button26: TButton;
    Button27: TButton;
    Button22: TButton;
    Edit32: TEdit;
    Edit33: TEdit;
    Edit34: TEdit;
    Edit35: TEdit;
    Edit36: TEdit;
    Edit37: TEdit;
    Button32: TButton;
    Button33: TButton;
    Button34: TButton;
    Button35: TButton;
    Button36: TButton;
    Button37: TButton;
    AquisioRepetibiiidade1: TMenuItem;
    Fontdialog: TFontDialog;
    ColorDialog1: TColorDialog;
    Panel21: TPanel;
    Panel22: TPanel;
    Panel23: TPanel;
    Panel24: TPanel;
    Panel25: TPanel;
    Panel26: TPanel;
    Panel27: TPanel;
    Panel31: TPanel;
    Panel32: TPanel;
    Panel33: TPanel;
    Panel34: TPanel;
    Panel35: TPanel;
    Panel36: TPanel;
    Panel37: TPanel;
    SpeedButton7: TSpeedButton;
    SpeedButton8: TSpeedButton;
    Filtragem1: TMenuItem;
    ExtraodeValorMdioExcentricidade1: TMenuItem;
    SpeedButton9: TSpeedButton;
```

```
    SpeedButton10: TSpeedButton;
    FiltragemusandoFFT1: TMenuItem;
    SpeedButton11: TSpeedButton;
    Procurar1: TMenuItem;
    procedure Novo1Click(Sender:
TObject);
    procedure Abrir1Click(Sender:
TObject);
    procedure Salvar1Click(Sender:
TObject);
    procedure Sair1Click(Sender:
TObject);
    procedure Recortar1Click(Sender:
TObject);
    procedure Copiar1Click(Sender:
TObject);
    procedure Colar1Click(Sender:
TObject);
    procedure
SelecionarTudo1Click(Sender: TObject);
    procedure Fontes1Click(Sender:
TObject);
    procedure Sobre1Click(Sender:
TObject);
    procedure comeco(Sender: TObject);
    procedure
VerConfiguraoAtual1Click(Sender:
TObject);
    procedure
AlterarConfiguraao1Click(Sender: TObject);
    procedure Grfico1Click(Sender:
TObject);
    procedure Aquisio1Click(Sender:
TObject);
    procedure SpeedButton1Click(Sender:
TObject);
    procedure SpeedButton2Click(Sender:
TObject);
    procedure SpeedButton3Click(Sender:
TObject);
    procedure SpeedButton5Click(Sender:
TObject);
    procedure GrficoPolar1Click(Sender:
TObject);
    procedure SpeedButton6Click(Sender:
TObject);
    procedure AquisioMAR1Click(Sender:
TObject);
    procedure SpeedButton4Click(Sender:
TObject);
    procedure Button10Click(Sender:
TObject);
    procedure Button11Click(Sender:
TObject);
    procedure Button12Click(Sender:
TObject);
    procedure Button13Click(Sender:
TObject);
    procedure Button1Click(Sender:
TObject);
    procedure Button2Click(Sender:
TObject);
    procedure Button4Click(Sender:
TObject);
    procedure Button5Click(Sender:
TObject);
    procedure Button6Click(Sender:
TObject);

    procedure Button7Click(Sender:
TObject);
    procedure Button8Click(Sender:
TObject);
    procedure
RetirarExcentricidade1Click(Sender:
TObject);
    procedure
RetirarValorMdio1Click(Sender: TObject);
    procedure Button16Click(Sender:
TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button15Click(Sender: TObject);
    procedure Button21Click(Sender: TObject);
    procedure Button22Click(Sender: TObject);
    procedure Button23Click(Sender: TObject);
    procedure Button24Click(Sender: TObject);
    procedure Button25Click(Sender: TObject);
    procedure Button26Click(Sender: TObject);
    procedure Button27Click(Sender: TObject);
    procedure Button31Click(Sender: TObject);
    procedure Button32Click(Sender: TObject);
    procedure Button33Click(Sender: TObject);
    procedure Button34Click(Sender: TObject);
    procedure Button35Click(Sender: TObject);
    procedure Button36Click(Sender: TObject);
    procedure Button37Click(Sender: TObject);
    procedure Panel21Click(Sender: TObject);
    procedure Panel22Click(Sender: TObject);
    procedure Panel31Click(Sender: TObject);
    procedure Panel23Click(Sender: TObject);
    procedure Panel24Click(Sender: TObject);
    procedure Panel25Click(Sender: TObject);
    procedure Panel26Click(Sender: TObject);
    procedure Panel27Click(Sender: TObject);
    procedure Panel32Click(Sender: TObject);
    procedure Panel33Click(Sender: TObject);
    procedure Panel34Click(Sender: TObject);
    procedure Panel35Click(Sender: TObject);
    procedure Panel36Click(Sender: TObject);
    procedure Panel37Click(Sender: TObject);
    procedure AquisioRepetibilidade1Click(Sender:
TObject);
    procedure SpeedButton7Click(Sender: TObject);
    procedure SpeedButton8Click(Sender: TObject);
    procedure Filtragem1Click(Sender: TObject);
    procedure
ExtraodeValorMdioExcentricidade1Click(Sender: TObject);
    procedure SpeedButton9Click(Sender: TObject);
    procedure SpeedButton10Click(Sender: TObject);
    procedure AquisioMR1Click(Sender: TObject);
    procedure FiltragemusandoFFT1Click(Sender:
TObject);
    procedure SpeedButton11Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

  const
  n =256;

  type
  valores = array [1..n] of single;

  var
  Circularidade: TCircularidade;
  dados: valores; {matriz principal que guarda os
valores da coleta}
  arquivo: text; {arquivo de entrada e saída de dados
em formato .txt}
  name_file : string; {nome do arquivo *.clk}
   i,j,k : integer; {contadores gerais}
  canal1,canal2,encoder: byte; {canais de leitura}
  st:string; {string utilizada na conversão de número
para caracter}

escala,escala_sensor1,escala_sensor2,escala_amp1,escala
_amp2: single; {escalas de conversão}
  raio,amplia,base : integer;
   deltax, deltay : valores; {utilizados no calculo do
centro LSC}
    tabela_repete: array [1..14] of valores; { para
gráfico comparativo}
    valor : byte; {usado para marcar quantidade de
curvas no gráfico comparativo}
    cores : array [1..14] of Tcolor; {cores usadas no
gráfico comparativo}
    H1,H2: word; {contador de tempo}
```

```
    vcos,vsin : valores; {guardam valores
de cos e sin mais usados}

function st2byte(st:string):byte;
function byte2st(num:byte):string;
function single2st(num:single):string;
function ssingle2st(num:single):string;
function int2st(num:integer):string;
function st2single(st:string):single;
procedure botao_carregar(local:Tedit);
procedure carregar;
procedure gravar(var dados: valores; var
escala_sensor:single);
procedure botao_gravar(local:Tedit);
function raio_medio(var dados:array of
single):single;
function centro(valores : array of
single):single;
procedure OutPort(PortAddr: word;
Databyte: byte);
assembler; stdcall;
function InPort(PortAddr: word): byte;
assembler; stdcall;
procedure falso_espera0(var canaL:byte);
procedure falso_espera(var canal:byte);
procedure falso_diferente(var
canal:byte);
function Leitura(var dados : array of
single;var canal:byte):single;
function timer:word;
procedure atualiza_tabela;
procedure limpar_dados(var dados:array of
single);
function ad(var canal:byte):single;
procedure corrige(var dados:valores);

implementation

uses apresentacao, inicio_tf, ver_config,
alterar_config, linear,
  coleta_unit, polar, coleta_mar, repete,
col_repete, filtro_unit,
  medio_unit, l_teste, fft;

{$R *.DFM}
{*********************************}
{InPort recebe o valor da PortAddr}
function InPort(PortAddr: word): byte;
{$IFDEF VER90}
assembler; stdcall;
asm
        mov dx,PortAddr
        in al,dx
end;
{$ELSE}
begin
Result := Port[PortAddr];
end;
{$ENDIF}
{*********************************}
{OutPort escreve o valor Databyte na
Porta PortAddr}
procedure OutPort(PortAddr: word;
Databyte: byte);
{$IFDEF VER90}
assembler; stdcall;
asm
   mov al,Databyte
   mov dx,PortAddr
   out dx,al
end;
{$ELSE}
begin
Port[PortAddr] := DataByte;
end;
{$ENDIF}
{*********************************}
function st2single(st:string):single;
var code:integer; {converte string para
single}
    v:single;
begin
val(st,v,code);
st2single:=v;
end;
{*****************************************}
function timer:word; {calcular diferença de tempo em
ms.}
begin
h2:=getcurrenttime;
timer:=(h2-h1);
end;
{*****************************************}
function int2st(num:integer):string;
begin {converter valor de byte para string}
str(num,st);
int2st:=st;
end;
{*****************************************}
function ssingle2st(num:single):string;
begin {converter valor de byte para string sem nenhuma
casa após a virgula}
str(num:1:0,st);
ssingle2st:=st;
end;
{*****************************************}
function ad(var canal:byte):single;
{faz a leitura do canal - conversão analógica/Digital}
var a,b,s : integer;
valor:single;
begin
outport(727,16*15+canal); {canal do encoder}
outport(723,178); {modo}
outport(722,16);
outport(722,39);
repeat
      b:=inport(725);
      s:=(b and $10);
until (s=0);
a:=inport(724);
valor:=((b and $3)*256+a);
ad:=(valor-512)*0.009766;
end;
{*****************************************}
procedure falso_espera0(var canal:byte);
{Em vez de se utilizar a entrada GATE 2 da placa CAD
1026, utiliza-se o
canal}
var valor : single;
begin
valor:=0;
repeat
begin
valor:=ad(encoder);
{filtrar o sinal}
if (valor<0) or (valor>4) or ((valor>1)and(valor<3))
then valor:=0;
end;
until (valor>3);
valor:=ad(canal); {nova leitura é feita para limpar o
buffer}
end;
{*****************************************}
procedure falso_espera(var canal:byte);
{Em vez de se utilizar a entrada GATE 2 da placa CAD
1026, utiliza-se o
canal}
var valor : single;  {borda de subida}
begin
valor:=0;
repeat
begin
valor:=ad(encoder);
{filtrar o sinal}
if (valor<0) or (valor>4) or ((valor>1)and(valor<3))
then valor:=0;
end;
until (valor>1) or (timer>3000);
valor:=ad(canal);
end;
```

```
{************************************}
procedure falso_diferente(var
canal:byte); {Utiliza o canal do encoder}
var valor : single;    {borda de descida}
begin
valor:=5;
repeat
begin
valor:=ad(encoder);
if (valor<0) or (valor>4) or
((valor>1)and(valor<3)) then valor:=5;
end;
until (valor<1);
valor:=ad(canal);
end;
{*********************************}
function Leitura(var dados : array of
single;var canal:byte):single;
var  {valores tem 10 bits}
    numero: integer;
    valor:single;
begin
     numero:=0;
     valor:=ad(canal); {três medições
seguidas para pegar valor correto}
     valor:=ad(canal);
     valor:=ad(canal);
     i:=i+1;
     dados[i]:=valor;
valor:=ad(canal);
leitura:=dados[i];
end;
{*********************************}
procedure
TCircularidade.Novo1Click(Sender:
TObject);
begin     {limpar campos e dados}
memo.setfocus;
memo.lines.clear;
for k:=1 to 4 do {Limpar células da
tabela}
for j:=1 to n do
    begin
    dados[j]:=0;
    tabela.cells[k-1,j]:='';
    end;
end;
{*********************************}
procedure carrega_config; {carrega
configuracoes do programa}
begin
assignfile(arquivo,'wclk.ini');
reset(arquivo);
readln(arquivo);
readln(arquivo,encoder);
readln(arquivo);
readln(arquivo,canal1);
readln(arquivo);
readln(arquivo,canal2);
readln(arquivo);
readln(arquivo,raio);
readln(arquivo);
readln(arquivo,amplia);
readln(arquivo);
readln(arquivo,escala_sensor1);
readln(arquivo);
readln(arquivo,escala_sensor2);
readln(arquivo);
readln(arquivo,base);
close(arquivo);
end;
{************************************}
procedure atualiza_tabela; {faz aparecer
novos valores na tabela}
begin
with circularidade do
      for j:=1 to n do
          begin
tabela.cells[1,j]:=inttostr(round((dados[j]/0.009766)+5
12));
      {coloca byte mais próximo}
      st:=single2st(dados[j]);
      tabela.cells[2,j]:=st;
          str(dados[j]*escala_sensor1:1:5,st);
          tabela.cells[3,j]:=st;
          end;
end;
{*****************************************}
procedure carregar;
var code:integer;
    v:single;
begin
if fileexists(name_file)=true then
begin
with circularidade do
begin
memo.lines.loadfromfile(opendialog1.filename);
      memo.setfocus;
      assignfile(arquivo,name_file);
      reset(arquivo);
      readln(arquivo);
      i:=0;
      while (not eof(arquivo)) and (i<n) do
      begin
      i:=i+1;
      readln(arquivo,st);
      val(st,v,code);
      dados[i]:=v;
      end;
      readln(arquivo,escala_sensor1); {escala
utilizada}
      if escala_sensor1=0 then escala_sensor1:=1;
{escala padrao}
      closefile(arquivo);
      end;
      atualiza_tabela;
end
else showmessage('Arquivo não existe !');
end;
{*********************************************}
procedure TCircularidade.Abrir1Click(Sender: TObject);
begin
opendialog1.filter:='Arquivos (*.clk)|*.clk';
if opendialog1.execute then
   begin
      name_file:=opendialog1.filename;
      carregar;
   end;
end;
{*********************************************}
procedure gravar(var dados:valores; var
escala_sensor:single);
begin
assignfile(arquivo,name_file);
rewrite(arquivo);
if (i<256) then name_file:=name_file+'c/ problemas';
writeln(arquivo,'Nome: '+name_file);
for j:=1 to n do {gravar informacao até 256}
    writeln(arquivo,dados[j]:1:5);
writeln(arquivo,escala_sensor:1:5); {escala utilizada}
closefile(arquivo);
end;
{*****************************************}
procedure TCircularidade.Salvar1Click(Sender: TObject);
begin
escala:=escala_sensor1*escala_amp1;
savedialog1.filter:='Arquivos (*.clk)|*.clk';
if savedialog1.execute then
memo.lines.savetofile(savedialog1.filename);
name_file:=savedialog1.filename;
if i<n-6 then showmessage('Número insuficiente de
pontos: '+inttostr(i))
else
gravar(dados,escala);
end;
{*********************************************}
procedure TCircularidade.Sair1Click(Sender: TObject);
```

```
begin     {desligar o software}
halt;
end;
{************************************}
procedure
TCircularidade.Recortar1Click(Sender:
TObject);
begin
memo.cuttoclipboard;
end;
{************************************}
procedure
TCircularidade.Copiar1Click(Sender:
TObject);
begin
memo.copytoclipboard;
end;
{************************************}
procedure
TCircularidade.Colar1Click(Sender:
TObject);
begin
memo.pastefromclipboard;
end;
{************************************}
procedure
TCircularidade.SelecionarTudo1Click(Sende
r: TObject);
begin
memo.selectall;
end;
{************************************}
procedure
TCircularidade.Fontes1Click(Sender:
TObject);
begin
if fontdialog.execute then
memo.font:=fontdialog.font;
end;
{************************************}
procedure
TCircularidade.Sobre1Click(Sender:
TObject);
begin
sobre.showmodal;
end;
{************************************}
procedure TCircularidade.comeco(Sender:
TObject);
begin
tabela.cells[0,0]:='Ângulo (°)'; {Colocar
legenda na tabela}
tabela.cells[1,0]:='Binário';
tabela.cells[2,0]:='Valor (V)';
tabela.cells[3,0]:='Valor (µm)';
for j:=1 to 256 do

tabela.cells[0,j]:=single2st(360/n*j);
for j:=1 to n do
    begin
    vcos[j]:=cos(j*2*pi/n);
    vsin[j]:=sin(j*2*pi/n);
    end;
carrega_config;
escala_amp1:=1; {inicializar
amplificadores}
escala_amp2:=1;
inicio.showmodal; {fazer propaganda do
software}
end;
{************************************}
function byte2st(num:byte):string;
begin {converter valor de byte para
string}
str(num,st);
byte2st:=st;
end;
{************************************}
function single2st(num:single):string;
begin {converter valor de single para string}
str(num:1:5,st);
single2st:=st;
end;
{************************************}
procedure
TCircularidade.VerConfiguracaoAtual1Click(Sender:
TObject);
begin {abrir janela que mostra configuração atual do
software}
config_atual.label3.caption:=byte2st(canal1);
config_atual.label4.caption:=byte2st(canal2);
config_atual.label6.caption:=byte2st(encoder);
config_atual.label10.caption:=single2st(escala_sensor1)
;
config_atual.label9.caption:=single2st(escala_sensor2);
config_atual.label15.caption:=single2st(escala_amp1);
config_atual.label16.caption:=single2st(escala_amp2);
config_atual.label18.caption:=single2st(escala_sensor1*
escala_amp1);
config_atual.label17.caption:=single2st(escala_sensor2*
escala_amp2);
config_atual.showmodal;
end;
{************************************}
function st2byte(st:string):byte;
var v,code:integer; {converte string para byte}
begin
val(st,v,code);
st2byte:=v;
end;
{************************************}
procedure
TCircularidade.AlterarConfiguracao1Click(Sender:
TObject);
begin
with config do
begin
combobox1.text:=byte2st(canal1);
combobox2.text:=byte2st(canal2);
combobox3.text:=byte2st(encoder);
edit1.text:=single2st(escala_sensor1);
edit3.text:=single2st(escala_sensor2);
edit2.text:=single2st(escala_amp1);
edit4.text:=single2st(escala_amp2);
end;
config.showmodal;
end;
{************************************}
procedure TCircularidade.Grfico1Click(Sender: TObject);
begin
grafico_linear.showmodal;
end;
{************************************}
procedure limpar_dados(var dados:array of single);
begin
for j:=1 to n do
    dados[j]:=0;
end;
{************************************}
function centro(valores : array of single):single;
{esta funcao obtem o centro atraves da formula: a=2
(somatoria de xi)/n}
var soma : single;
    j: integer;
begin
soma:=0;
for j:=1 to n do
    soma:=soma+valores[j];
centro:=2*soma/n;
end;
{************************************}
function raio_medio(var dados:array of single):single;
{retorna raio medio obtido por somatoria de x dividido
por n}
var soma :single;
    j:integer;
begin
soma:=0;
for j:=1 to n do
```

```
    soma:=soma+dados[j];
raio_medio:=soma/n;
end;
{**********************************}
procedure
TCircularidade.Aquisio1Click(Sender:
TObject);
begin
leitura_teste.showmodal;
end;

procedure
TCircularidade.SpeedButton1Click(Sender:
TObject);
begin
Circularidade.Novo1Click(sender);
end;

procedure
TCircularidade.SpeedButton2Click(Sender:
TObject);
begin
Circularidade.Abrir1Click(sender);
end;

procedure
TCircularidade.SpeedButton3Click(Sender:
TObject);
begin
Circularidade.Salvar1Click(sender);
end;

procedure
TCircularidade.SpeedButton5Click(Sender:
TObject);
begin
Circularidade.Grfico1Click(sender);
end;

procedure
TCircularidade.GrficoPolar1Click(Sender:
TObject);
begin
grafico_polar.showmodal;
end;

procedure
TCircularidade.SpeedButton6Click(Sender:
TObject);
begin
Circularidade.GrficoPolar1Click(sender);
end;

procedure
TCircularidade.AquisioMAR1Click(Sender:
TObject);
begin
form_mar.showmodal;
     end;

procedure
TCircularidade.SpeedButton4Click(Sender:
TObject);
begin
Circularidade.AquisioMR1Click(Sender);
end;
{**********************************}
procedure botao_carregar(local:Tedit);
begin
with Circularidade do
begin
opendialog1.filter:='Arquivos
(*.clk)|*.clk';
if opendialog1.execute then
     local.text:=opendialog1.filename
 else showmessage('Arquivo não existe
!');
end;
end;
{**********************************}

procedure botao_gravar(local:Tedit);
{ cria janela de gravar arquivo e coloca o nome do
arquivo no campo 'local'}
begin
with Circularidade do
begin
savedialog1.filter:='Arquivos (*.clk)|*.clk';
if savedialog1.execute then
     local.text:=savedialog1.filename;
end;
end;

procedure TCircularidade.Button10Click(Sender:
TObject);
begin
botao_carregar(edit9);
end;

procedure TCircularidade.Button11Click(Sender:
TObject);
begin
botao_carregar(edit10);
end;

procedure TCircularidade.Button12Click(Sender:
TObject);
begin
botao_gravar(edit12);
end;

procedure TCircularidade.Button13Click(Sender:
TObject);
begin
botao_gravar(edit11);
end;

procedure TCircularidade.Button1Click(Sender: TObject);
begin
botao_carregar(edit1);
end;

procedure TCircularidade.Button2Click(Sender: TObject);
begin
botao_carregar(edit2);
end;

procedure TCircularidade.Button4Click(Sender: TObject);
begin
botao_carregar(edit4);
end;

procedure TCircularidade.Button3Click(Sender: TObject);
begin
botao_carregar(edit3);
end;

procedure TCircularidade.Button6Click(Sender: TObject);
begin
botao_gravar(edit6);
end;

procedure TCircularidade.Button7Click(Sender: TObject);
begin
botao_gravar(edit8);
end;

procedure TCircularidade.Button8Click(Sender: TObject);
begin
botao_gravar(edit7);
end;

procedure
TCircularidade.RetirarExcentricidade1Click(Sender:
TObject);
begin
retirarexcentricidade1.checked:=not(retirarexcentricida
de1.checked);
end;
```

```
procedure
TCircularidade.RetirarValorMdio1Click(Sen
der: TObject);
begin
retirarvalormdio1.checked:=not(retirarval
ormdio1.checked);
end;
{************************************}
procedure nomear(local: Tedit);
{ Coloca o nome do campo 'local' para ser
carregado }
begin
name_file:=local.text;
carregar;
end;
{************************************}
procedure corrige(var dados:valores);
VAR im0:valores; {faz extração do valor
médio e da excentricidade}
    i,j:integer;
begin
for i:=1 to n do
im0[i]:=0; {valores imaginários são
nulos}
fft_p(dados,im0,8,-1); {realiza fft}
for j:=1 to 2 do
    begin
    im0[j]:=0;
    dados[j]:=0;
    end;
im0[n]:=0;
dados[n]:=0;
fft_p(dados, im0,8, 1); {realiza ifft}
end;
{************************************}
procedure
TCircularidade.Button16Click(Sender:
TObject);
var dados1,dados2:valores; {Executa
método da Reversão (MR)}
begin
nomear(edit9);
dados1:=dados;
corrige(dados1);
nomear(edit10);
dados2:=dados;
corrige(dados2);
for j:=1 to n do  {Cálculo do erro de
movimento}
    dados[j]:=(dados1[j]-dados2[j])/2;
name_file:=edit11.text;
escala:=escala_sensor1*escala_amp1;
gravar(dados,escala);
for j:=1 to n do  {Cálculo do erro de
forma - Circularidade do corpo de prova}
    dados[j]:=(dados1[j]+dados2[j])/2;
name_file:=edit12.text;
gravar(dados,escala);
end;
{************************************}
procedure
TCircularidade.Button9Click(Sender:
TObject);
var     auxiliar : integer;
    s1,s2,s11,s21: valores;
    {s1 - medicao[1]; s2- medicao [2];
s11-medicao[3], s21-medicao[4]}
begin
nomear(edit1); {corpo de prova}
s2:=dados;
corrige(s2);
nomear(edit2); {referencial auxiliar}
s1:=dados;
corrige(s1);
nomear(edit4); {corpo de prova revertido
s2'}
s21:=dados;
corrige(s21);
nomear(edit3); {referencial auxiliar após
a reversão do corpo de prova}
s11:=dados;
corrige(s11);
escala:=escala_sensor1*escala_amp1;
for j:=1 to n do {Calculo Principal - Circularidade}
    begin
   dados[j]:=((s2[j]-s1[j])+(s11[j]+s21[j]))/2;
    end;
name_file:=edit6.text;
gravar(dados,escala);
{Erro antes  da reversao : s1, s2, s11, s21
Erro depois da reversao : s11, s2, s1, s21 }
{calculo principal : somar os dois primeiros arquivos e
os dois ultimos arquivos; em seguida subtrair do
primeiro o segundo valor e
dividir por dois}
for j:=1 to n do
   dados[j]:=((s1[j]+s2[j])-(s11[j]+s21[j]))/2;

name_file:=edit8.text;
gravar(dados,escala);
for j:=1 to n do
   dados[j]:=((s11[j]+s2[j])-(s1[j]+s21[j]))/2;
name_file:=edit7.text;
gravar(dados,escala);
end;
{************************************************}
procedure TCircularidade.Button15Click(Sender:
TObject);
var lista : array [1..14] of Tedit;
    chk : array [1..14] of Tcheckbox;
    num : byte;
    cor_list: array [1..14] of Tcolor;
begin
valor:=0;
lista[1]:=Edit21;lista[8]:=edit31;
lista[2]:=Edit22;lista[9]:=edit32;
lista[3]:=Edit23;lista[10]:=edit33;
lista[4]:=Edit24;lista[11]:=edit34;
lista[5]:=Edit25;lista[12]:=edit35;
lista[6]:=Edit26;lista[13]:=edit36;
lista[7]:=Edit27;lista[14]:=edit37;
chk[1]:=checkbox21;chk[8]:=checkbox31;
chk[2]:=checkbox22;chk[9]:=checkbox32;
chk[3]:=checkbox23;chk[10]:=checkbox33;
chk[4]:=checkbox24;chk[11]:=checkbox34;
chk[5]:=checkbox25;chk[12]:=checkbox35;
chk[6]:=checkbox26;chk[13]:=checkbox36;
chk[7]:=checkbox27;chk[14]:=checkbox37;
cor_list[1]:=panel21.color;cor_list[8]:=panel31.color;
cor_list[2]:=panel22.color;cor_list[9]:=panel32.color;
cor_list[3]:=panel23.color;cor_list[10]:=panel33.color;
cor_list[4]:=panel24.color;cor_list[11]:=panel34.color;
cor_list[5]:=panel25.color;cor_list[12]:=panel35.color;
cor_list[6]:=panel26.color;cor_list[13]:=panel36.color;
cor_list[7]:=panel27.color;cor_list[14]:=panel37.color;
for num:=1 to 14 do
    begin
    if chk[num].checked=true then
       begin
       valor:=valor+1;
       nomear(lista[num]);
       tabela_repete[valor]:=dados;
       cores[valor]:=cor_list[num];
       end;
    end;
    grafico_repete.showmodal;
for num:=1 to 14 do
    begin
    cores[num]:=0;
    for j:=1 to n do
        tabela_repete[num][j]:=0;
    end;
end;

procedure TCircularidade.Button21Click(Sender:
TObject);
begin
botao_carregar(edit21);
end;
```

```
procedure
TCircularidade.Button22Click(Sender:
TObject);
begin
botao_carregar(edit22);
end;

procedure
TCircularidade.Button23Click(Sender:
TObject);
begin
botao_carregar(edit23);
end;

procedure
TCircularidade.Button24Click(Sender:
TObject);
begin
botao_carregar(edit24);
end;

procedure
TCircularidade.Button25Click(Sender:
TObject);
begin
botao_carregar(edit25);
end;

procedure
TCircularidade.Button26Click(Sender:
TObject);
begin
botao_carregar(edit26);
end;

procedure
TCircularidade.Button27Click(Sender:
TObject);
begin
botao_carregar(edit27);
end;

procedure
TCircularidade.Button31Click(Sender:
TObject);
begin
botao_carregar(edit31);
end;

procedure
TCircularidade.Button32Click(Sender:
TObject);
begin
botao_carregar(edit32);
end;

procedure
TCircularidade.Button33Click(Sender:
TObject);
begin
botao_carregar(edit33);
end;

procedure
TCircularidade.Button34Click(Sender:
TObject);
begin
botao_carregar(edit34);
end;

procedure
TCircularidade.Button35Click(Sender:
TObject);
begin
botao_carregar(edit35);
end;

procedure
TCircularidade.Button36Click(Sender:
TObject);
begin
botao_carregar(edit36);
end;

procedure TCircularidade.Button37Click(Sender:
TObject);
begin
botao_carregar(edit37);
end;

procedure TCircularidade.Panel21Click(Sender: TObject);
begin
if colordialog1.execute then
panel21.color:=colordialog1.color;
end;

procedure TCircularidade.Panel22Click(Sender: TObject);
begin
if colordialog1.execute then
panel22.color:=colordialog1.color;
end;

procedure TCircularidade.Panel31Click(Sender: TObject);
begin
if colordialog1.execute then
panel31.color:=colordialog1.color;
end;

procedure TCircularidade.Panel23Click(Sender: TObject);
begin
if colordialog1.execute then
panel23.color:=colordialog1.color;
end;

procedure TCircularidade.Panel24Click(Sender: TObject);
begin
if colordialog1.execute then
panel24.color:=colordialog1.color;
end;

procedure TCircularidade.Panel25Click(Sender: TObject);
begin
if colordialog1.execute then
panel25.color:=colordialog1.color;
end;

procedure TCircularidade.Panel26Click(Sender: TObject);
begin
if colordialog1.execute then
panel26.color:=colordialog1.color;
end;

procedure TCircularidade.Panel27Click(Sender: TObject);
begin
if colordialog1.execute then
panel27.color:=colordialog1.color;
end;

procedure TCircularidade.Panel32Click(Sender: TObject);
begin
if colordialog1.execute then
panel32.color:=colordialog1.color;
end;

procedure TCircularidade.Panel33Click(Sender: TObject);
begin
if colordialog1.execute then
panel33.color:=colordialog1.color;
end;

procedure TCircularidade.Panel34Click(Sender: TObject);
begin
if colordialog1.execute then
panel34.color:=colordialog1.color;
end;

procedure TCircularidade.Panel35Click(Sender: TObject);
begin
if colordialog1.execute then
panel35.color:=colordialog1.color;
```

```
end;

procedure
TCircularidade.Panel36Click(Sender:
TObject);
begin
if colordialog1.execute then
panel36.color:=colordialog1.color;
end;

procedure
TCircularidade.Panel37Click(Sender:
TObject);
begin
if colordialog1.execute then
panel37.color:=colordialog1.color;
end;

procedure
TCircularidade.AquisioRepetibilidade1Clic
k(Sender: TObject);
begin
coleta_repete.showmodal;
end;

procedure
TCircularidade.SpeedButton7Click(Sender:
TObject);
begin
form_mar.showmodal;
end;

procedure
TCircularidade.SpeedButton8Click(Sender:
TObject);
begin
coleta_repete.showmodal;
end;

procedure
TCircularidade.Filtragem1Click(Sender:
TObject);
begin
filtro.showmodal;
end;

procedure
TCircularidade.ExtraodeValorMdioExcentric
idade1Click(
  Sender: TObject);
begin
medio.showmodal;
end;

procedure
TCircularidade.SpeedButton9Click(Sender:
TObject);
begin
filtro.showmodal;
end;

procedure
TCircularidade.SpeedButton10Click(Sender:
TObject);
begin
medio.showmodal;
end;

procedure
TCircularidade.AquisioMR1Click(Sender:
TObject);
begin
coleta.showmodal;
end;

procedure
TCircularidade.FiltragemusandoFFT1Click(S
ender: TObject);
begin
form_fft.showmodal;
end;

procedure TCircularidade.SpeedButton11Click(Sender:
TObject);
begin
form_fft.showmodal;
end;

end.
```

```
unit alterar_config;

interface

uses
  Windows, Messages, SysUtils, unit
col_repete;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  StdCtrls, Gauges;

type
  Tcoleta_repete = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Gauge1: TGauge;
    Gauge2: TGauge;
    procedure Button1Click(Sender:
TObject);
    procedure comeco(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  coleta_repete: Tcoleta_repete;

implementation

uses clk;

{$R *.DFM}
var
quantidade,inicio:byte;
v:integer;


{************************************}
procedure amostragem;
var {Processo de coleta de dados}
    s: array[0..20] of char;
begin
{espera;} falso_espera0(canal1);
i:=0; {no inicio nao temos nenhum ponto}
repeat
      begin
      leitura(dados,canal1);
      falso_diferente(canal1);
      leitura(dados,canal1);
      h1:=getcurrenttime;
      falso_espera(canal1);
      with coleta_repete do
           begin
      gauge1.progress:=(100*i)div(256);
      gauge2.progress:=((100*(i+(v-
1)*256))div(256))div(quantidade);
      end;
      end;
until (i>260) or (timer>3000);
beep;
end;

procedure
Tcoleta_repete.Button1Click(Sender:
TObject);
var num : byte;  {iniciar coleta de
muitos dados seguidamente}
    st:string;
    j :integer;
    raio_m:single;
begin
button1.caption:='Coletando...';
button1.enabled:=false;
quantidade:=st2byte(edit1.text);
inicio:=st2byte(edit2.text);
gauge2.progress:=0;
for v:=1 to quantidade do
begin
limpar_dados(dados);
      gauge1.progress:=0;
         amostragem;
         name_file:=byte2st(inicio);
         name_file:=name_file+'.clk';
         escala:=escala_sensor1*escala_amp1;
         gravar(dados,escala);
         inicio:=inicio+1;
end;
beep;
button2.caption:='Terminado !';
end;

procedure Tcoleta_repete.comeco(Sender: TObject);
begin
button1.caption:='Ok';
button1.enabled:=true;
button2.caption:='Cancelar';
end;

end. Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls,clk;

type
  Tconfig = class(TForm)
    Button1: TButton;
    Button2: TButton;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    ComboBox1: TComboBox;
    ComboBox2: TComboBox;
    Label2: TLabel;
    Label3: TLabel;
    ComboBox3: TComboBox;
    GroupBox2: TGroupBox;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  config: Tconfig;
implementation
{$R *.DFM}
{*******************************************}
procedure salva_config; {carrega configuracoes do
programa}
begin
assignfile(arquivo,'wclk.ini');
rewrite(arquivo);
writeln(arquivo,'// Canal do encoder');
writeln(arquivo,encoder);
writeln(arquivo,'// Canal 1 de Leitura');
writeln(arquivo,canal1);
writeln(arquivo,'// Canal 2 de Leitura');
writeln(arquivo,canal2);
writeln(arquivo,'// Raio do modo gráfico');
writeln(arquivo,raio);
writeln(arquivo,'// Amplificação do modo gráfico');
writeln(arquivo,amplia);
writeln(arquivo,'// Escala do sensor1');
writeln(arquivo,escala_sensor1:2:3);
```

```
writeln(arquivo,'// Escala do sensor2');
writeln(arquivo,escala_sensor2:2:3);
writeln(arquivo,'Valor do Endereço de
base');
writeln(arquivo,base);
closefile(arquivo);
end;

procedure Tconfig.Button1Click(Sender:
TObject);
begin
with config do
begin
canal1:=st2byte(combobox1.text);
canal2:=st2byte(combobox2.text);
encoder:=st2byte(combobox3.text);
escala_sensor1:=st2byte(edit1.text);
escala_sensor2:=st2byte(edit3.text);
escala_amp1:=st2single(edit2.text);
escala_amp2:=st2single(edit4.text);
salva_config;
end;
end;

end.
```

```
unit apresentacao;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Buttons;

type
  Tsobre = class(TForm)
    Image1: TImage;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  sobre: Tsobre;

implementation

{$R *.DFM}

end.
```

```
unit coleta_mar;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  StdCtrls,clk, Gauges;

type
  Tform_mar = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Gauge1: TGauge;
    Edit1: TEdit;
    Edit2: TEdit;
    Label4: TLabel;
    Label5: TLabel;
    procedure Button1Click(Sender:
TObject);
    procedure comeco(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  form_mar: Tform_mar;

implementation


{$R *.DFM}

procedure Tform_mar.Button1Click(Sender:
TObject);
var
    j,k :integer;
    raio_m:single;
    dados1,dados2:valores;
begin
limpar_dados(dados1);
limpar_dados(dados2);
gauge1.progress:=0;
button1.caption:='coletando...';
button1.enabled:=false;
falso_espera0(canal1);
i:=0; {no inicio nao temos nenhum ponto}
repeat
      begin
      dados1[i]:=leitura(dados1,canal1);
      i:=i-1; {sao duas amostragens ao
mesmo tempo}
      dados2[i]:=leitura(dados2,canal2);
      {diferente;}
falso_diferente(canal1);
      dados1[i]:=leitura(dados1,canal1);
      i:=i-1;
      dados2[i]:=leitura(dados2,canal2);
      h1:=getcurrenttime;
      {espera;} falso_espera(canal1);
      gauge1.progress:=(100*i)div(256);
      end;
until (i>n) or (timer>1000);
beep;
button2.caption:='Terminado !';
if
clk.Circularidade.RetirarValorMdio1.check
ed=true then
   begin
   raio_m:=raio_medio(dados1);
for j:=1 to n do
    dados1[j]:=dados1[j]-raio_m;
    raio_m:=raio_medio(dados2);
    for j:=1 to n do
    dados2[j]:=dados2[j]-raio_m;
   end;
if
clk.Circularidade.RetirarExcentricidade1.checked=true
then
   begin
   corrige(dados1);
   corrige(dados2);
   end;
name_file:=edit1.text; {gravar o primeiro canal lido}
raio_m:=escala_sensor1*escala_amp1;
gravar(dados1,raio_m);
name_file:=edit2.text; {gravar o segundo canal lido}
raio_m:=escala_sensor2*escala_amp2;
gravar(dados2,raio_m);
end;
{*****************************************}
procedure Tform_mar.comeco(Sender: TObject);
begin
button1.caption:='Ok';
button1.enabled:=True;
button2.caption:='Cancelar';
end;

end.
```

```
unit coleta_unit;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  StdCtrls,clk,ExtCtrls, Gauges;

type
  Tcoleta = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Panel1: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Gauge1: TGauge;
    procedure Button1Click(Sender:
TObject);
    procedure comeco(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  coleta: Tcoleta;

implementation

{$R *.DFM}
procedure Tcoleta.Button1Click(Sender:
TObject);
var j :integer;
    raio_m:single;
begin
limpar_dados(dados);
button1.caption:='Coletando...';
button1.enabled:=false;
gauge1.progress:=0; {marcar porcentagem
coletada}
falso_espera0(canal1);
i:=0; {no inicio nao temos nenhum ponto}
repeat
      begin
      dados[i]:=leitura(dados,canal1);
      falso_diferente(canal1);
      dados[i]:=leitura(dados,canal1);
      h1:=getcurrenttime;
      falso_espera(canal1);
      gauge1.progress:=(100*i)div(256);
      end;
until (i>260) or (timer>3000);
beep; {fazer barulho}
button2.caption:='Terminado!';
if
clk.Circularidade.RetirarValorMdio1.check
ed=true then
   begin
   raio_m:=raio_medio(dados);
   for j:=1 to n do
       dados[j]:=dados[j]-raio_m;
   end;
if
clk.Circularidade.RetirarExcentricidade1.
checked=true then
   corrige(dados);
atualiza_tabela;
end;

procedure Tcoleta.comeco(Sender:
TObject);
begin
button1.caption:='Ok';
button1.enabled:=true;
button2.caption:='Cancelar';
end;

end.


unit fft;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
  StdCtrls,clk;

type
  Tform_fft = class(TForm)
    GroupBox1: TGroupBox;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Button3: TButton;
    Button4: TButton;
    Label4: TLabel;
    Label5: TLabel;
    CheckBox1: TCheckBox;
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  form_fft: Tform_fft;
procedure fft_p(var fr, fi : valores; lnN : integer;
sign : integer);


implementation

{$R *.DFM}

procedure Tform_fft.Button3Click(Sender: TObject);
begin
botao_carregar(edit1);
end;

function power2(i : integer) : integer;
var n : integer; {potência de base 2}
begin
  n := 1;
  for i := 1 to i do n := n*2;
  power2:= n;
end;

procedure Tform_fft.Button4Click(Sender: TObject);
begin
botao_gravar(edit2);
end;

procedure fft_p(var fr, fi : valores; lnN : integer;
sign : integer);
var t, nd2, nm1 : integer;
    i,j,k,l, le, le1, ip : integer;
    divN : single;
       label 1,2,3;
    var s, ur, ur1,  ui, wr, wi, tr, ti, dv : real;
begin {algoritmo para fft e ifft}
divN := 1/sqrt(n*1.0);
  nd2 := N div 2;
  nm1 := n-1;
  j := 1;
  for i := 1 to N-1 do
```

```
    begin
      if i >= j then goto 1;
      s := fr[i]; fr[i] := fr[j]; fr[j] :=
s;
      s := fi[i]; fi[i] := fi[j]; fi[j] :=
s;
      1: k := nd2;
      2: if k>=j then goto 3;
      j := j - k;
      k := k div 2;
      goto 2;
      3: j := j + k;
    end;
    For l := 1 to lnN do
    begin
      le := Power2(l);
      le1 := le div 2;
      if le1=0 then beep;
      ur := 1;
      ui := 0;
      wr := cos(pi/le1);
      wi := sign*sin(pi/le1);
      for j:=1 to le1 do
      begin
        i := j;
        while (i<=N) do
        begin
          ip := i + le1;
          tr := fr[ip]*ur - fi[ip]*ui;
          ti := fr[ip]*ui + fi[ip]*ur;
          fr[ip] := fr[i] - tr;
          fi[ip] := fi[i] - ti;
          fr[i] := fr[i] + tr;
          fi[i] := fi[i] + ti;
          i := i + le;
        end;
        ur1 := ur*wr - ui*wi;
        ui := ur*wi + ui*wr;
        ur := ur1;
      end;
    end;   {l - l>kke}
    for i := 1 to N do
    begin
      fr[i] := fr[i]*divN;
      fi[i] := fi[i]*divN;
    end;
end;

procedure Tform_fft.Button1Click(Sender:
TObject);
var r0: valores; {valores imaginários}
    i : integer;
    f : integer; {ordem que será
extraído}
begin
name_file:=edit1.text;
carregar;
for i:=1 to n do
r0[i]:=0; {zera valores imaginários}
fft_p(dados,r0,8,-1); {realiza fft}
if checkbox1.checked=false then
   begin
   f:=st2byte(edit3.text);
   for i := 2+f to N-f do {filtra
frequencias}
   begin
        r0[i] := 0;
        dados[i] := 0;
   end;
   fft_p(dados, r0,8, 1); {realiza ifft}
end;
name_file:=edit2.text;
gravar(dados,escala_sensor1);
atualiza_tabela;
end;

end.
```

```
unit filtro_unit;

interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
  StdCtrls,clk;

type
  TFiltro = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Label5: TLabel;
    Label6: TLabel;
    Edit2: TEdit;
    Button3: TButton;
    Button4: TButton;
    GroupBox1: TGroupBox;
    CheckBox1: TCheckBox;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Filtro: TFiltro;
implementation
{$R *.DFM}
procedure TFiltro.Button3Click(Sender: TObject);
begin
botao_carregar(edit1);
end;
procedure TFiltro.Button4Click(Sender: TObject);
begin
botao_carregar(edit2);
end;

procedure TFiltro.Button1Click(Sender: TObject);
var j: integer;
    res: valores;
    a,b,c : single;
    code : integer;
begin
name_file:=edit1.text;
carregar;
res[1]:=dados[1];
if checkbox1.checked=true then
   begin  {usuário possui outras constantes}
   val(edit3.text,a,code);
   val(edit4.text,b,code);
   val(edit5.text,c,code);
   end
else
    begin {adotar constantes}
    a:=0.2934;
    b:=0.3533;
    c:=0.3533;
    end;
for j:=2 to n do {filtro ButterWorth de primeira ordem}
    res[j]:=a*res[j-1]+b*dados[j]+c*dados[j-1];
dados:=res;
name_file:=edit2.text;
a:=escala_sensor1*escala_amp1;
gravar(dados,a);
end;
end.
```

```
unit inicio_tf;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls;

type
  Tinicio = class(TForm)
    Button1: TButton;
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  inicio: Tinicio;

implementation
{$R *.DFM}

end.
```

```
unit l_teste;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
  StdCtrls,clk;

type
  Tleitura_teste = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Label3: TLabel;
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  leitura_teste: Tleitura_teste;

implementation

{$R *.DFM}

procedure Tleitura_teste.Button2Click(Sender: TObject);
var numero:byte;
begin
     numero:=st2byte(edit1.text); {consegue canal de
leitura}
     label2.caption:=single2st(ad(numero)); {faz
leitura do canal}
end;

end.
```

```
unit linear;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls,clk;

type
  Tgrafico_linear = class(TForm)
    Button1: TButton;
    Image1: TImage;
    button2: TButton;
    Edit1: TEdit;
    procedure linha(x1,y1,x2,y2:integer);
    procedure desenho_linear(Sender:
TObject);
    procedure Button1Click(Sender:
TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  grafico_linear: Tgrafico_linear;

implementation

{$R *.DFM}
procedure
Tgrafico_linear.linha(x1,y1,x2,y2:integer
);
begin  {Traça linha do ponto (x1,y1) até
x2,y2}
with image1.canvas do
     begin
     moveto(x1,y1);
     lineto(x2,y2);
     end;
end;

procedure
Tgrafico_linear.desenho_linear(Sender:
TObject);
var
    x:integer;
{posicao x no grafico}
    j,k:integer;
    st:string;
begin
  with image1.canvas do
  begin
 rectangle(0,0,clientwidth,clientheight);
 font.height:=30;
 x:=75;
 j:=(clientheight)div(10);
 for k:=1 to 9 do   {Escrever escala: -5
a 5 V}
     begin
     st:=int2st(5-k)+'V';
     textout(2*256+x,j*k-15,st);
     st:=ssingle2st((5-
K)*escala_sensor1*escala_amp1)+'µm';
     textout(1,j*k-15,st);
     end;
 textout(2*256+x,1,' 5V');
textout(2*256+x,10*j-30,'-5V');
st:=ssingle2st(5*escala_sensor1*escala_am
p1)+'µm';
textout(1,1,st);
st:=ssingle2st(-
5*escala_sensor1*escala_amp1)+'µm';
textout(1,10*j-30,st);

  x:=70;
       {tracar eixos e grades}
       repeat
            begin
            linha(x,j,x+2*256,j);
            j:=j+(clientheight)div(10);
            end;
       until j>clientheight;
       j:=x;
       repeat
            begin
            linha(j,0,j,clientheight);
            j:=j+2*64;
            end;
       until j>x+2*256;
       {x:=10;}
       x:=70;
       for j:=1 to n do
           begin
           k:=(clientheight)div(2)-
trunc(dados[j]*((clientheight)div(10)));
           x:=x+2;
           linha(x-2,(clientheight)div(2)-
trunc(dados[j-1]*((clientheight)div(10))),x,k);
           end;

end;
    end;
procedure Tgrafico_linear.Button1Click(Sender:
TObject);
begin
image1.picture.bitmap.monochrome:=true;
image1.picture.savetofile(edit1.text);
end;

end.
```

```
unit medio_unit;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs,clk,
  StdCtrls,fft;

type
  TMedio = class(TForm)
    GroupBox1: TGroupBox;
    CheckBox1: TCheckBox;
    CheckBox2: TCheckBox;
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    procedure Button3Click(Sender:
TObject);
    procedure Button4Click(Sender:
TObject);
    procedure Button1Click(Sender:
TObject);
  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Medio: TMedio;

implementation

{$R *.DFM}


procedure TMedio.Button3Click(Sender:
TObject);
begin
botao_carregar(edit1);
end;

procedure TMedio.Button4Click(Sender:
TObject);
begin
botao_gravar(edit2);
end;

procedure TMedio.Button1Click(Sender:
TObject);
var i,j:integer;
    raio_m:single;
    centrox,centroy : single;
        alfa : single;
    beta : single;
    x : single;
    raio : single; {distancia do centro
de coordenadas ate' o centro}
    st:string;
    r0:valores;

begin
name_file:=edit1.text;
carregar;
if checkbox1.checked=true then {retirar
valor médio}
  begin
  raio_m:=raio_medio(dados);
  for j:=1 to n do
    dados[j]:=dados[j]-raio_m;
  end;
if checkbox2.checked=true then {retirar
excentricidade e valor médio}
   corrige(dados);
name_file:=edit2.text;
raio_m:=escala_sensor1*escala_amp1;
gravar(dados,raio_m);
end;

end.

unit polar;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,clk,
  StdCtrls, ExtCtrls;

type
  Tgrafico_polar = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Image1: TImage;
    Edit1: TEdit;
    Button2: TButton;
    procedure desenho_polar(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  grafico_polar: Tgrafico_polar;

implementation

{$R *.DFM}

procedure Tgrafico_polar.desenho_polar(Sender:
TObject);
var
    maior_delta,menor_delta : single; {maior e menor
desvio do grafico}
    centrox,centroy : single;          {centro LSC}
    maior_raio,menor_raio:single;      {maior e menor
raio - LSC}
    alfa: single;                      {angulo - grafico
polar}
    dx,dy,rx,ry : single;              {sao usados no
grafico polar}
    xa,ya : single;                    {x,y anteriores}
    x:integer;                         {posicao x no
grafico}

    raio:single;
    amplia:integer;
    j : integer;
    LSC : single;
begin

with image1.Canvas do
begin
rectangle(0,0,image1.width,clientheight);  {limpar
área}
alfa:=0;
```

```
        raio:=100;
        amplia:=10;
        deltax[1]:=dados[1];
        deltay[1]:=0;


xa:=image1.width/2+raio+deltax[1]*amplia;
        ya:=clientheight/2;

        maior_delta:=dados[1];
        menor_delta:=dados[1];

        for j:=2 to n do
            begin
            {variacao angular}

rx:=(image1.width/2+vcos[j]*raio);
            ry:=(clientheight/2-
vsin[j]*raio);

            deltax[j]:=vcos[j]*dados[j];
            deltay[j]:=vsin[j]*dados[j];

            dx:=rx+(deltax[j]*amplia);
            dy:=ry-(deltay[j]*amplia);
            if dados[j]>maior_delta then

maior_delta:=dados[j];

            if dados[j]<menor_delta then

menor_delta:=dados[j];
            moveto(trunc(xa),trunc(ya));
            lineto(trunc(dx),trunc(dy));
            xa:=dx;
            ya:=dy;
            end;
        alfa:=0;

dx:=(image1.width/2+raio+vcos[j]*dados[1]
*amplia);
        dy:=clientheight/2;
        moveto(trunc(xa),trunc(ya));
        lineto(trunc(dx),trunc(dy));


centrox:=(image1.width/2+centro(deltax)*a
mplia); {centro LSC}
        centroy:=(clientheight/2-
centro(deltay)*amplia);


        maior_raio:=
raio+maior_delta*amplia-

sqrt((image1.width/2-
centrox)*(image1.width/2-centrox)
                    +(clientheight/2-
centroy)*(clientheight/2-centroy));
        menor_raio:=
raio+menor_delta*amplia-

sqrt((image1.width/2-
centrox)*(image1.width/2-centrox)
                    +(clientheight/2-
centroy)*(clientheight/2-centroy));


        for j:=1 to n do
            begin
            rx:= raio+dados[j]*amplia-
                sqrt((image1.width/2-
centrox)*(image1.width/2-centrox)
                +(clientheight/2-
centroy)*(clientheight/2-centroy));
            if rx>maior_raio then
               begin
               maior_raio:= rx;
               maior_delta:=dados[j];
               end;
            if rx<menor_raio then
               begin
               menor_raio:= rx;
               menor_delta:=dados[j];
               end;
            end;

        xa:=(centrox);
        ya:=(centroy);
        moveto(trunc(xa-12),trunc(ya));
        lineto(trunc(xa+12),trunc(ya));
        moveto(trunc(xa),trunc(ya-12));
        lineto(trunc(xa),trunc(ya+12));

        {mudar centro da figura e maior e menor delta}
        maior_delta:= maior_delta-
sqrt(centro(deltax)*centro(deltax)
                    +centro(deltay)*centro(deltay));
        menor_delta:= menor_delta-
sqrt(centro(deltax)*centro(deltax)
                    +centro(deltay)*centro(deltay));


ellipse(trunc(xa+maior_raio),trunc(ya+maior_raio),trunc
(xa-maior_raio),trunc(ya-maior_raio));

ellipse(trunc(xa+menor_raio),trunc(ya+menor_raio),trunc
(xa-menor_raio),trunc(ya-menor_raio));

          {******}
            xa:=image1.width/2+raio+deltax[1]*amplia;
        ya:=clientheight/2;
          for j:=2 to n do
            begin
            {variacao angular}
            rx:=(image1.width/2+vcos[j]*raio);
            ry:=(clientheight/2-vsin[j]*raio);

            deltax[j]:=vcos[j]*dados[j];
            deltay[j]:=vsin[j]*dados[j];

            dx:=rx+(deltax[j]*amplia);
            dy:=ry-(deltay[j]*amplia);
            moveto(trunc(xa),trunc(ya));
            lineto(trunc(dx),trunc(dy));
            xa:=dx;
            ya:=dy;
            end;
            alfa:=0;
        dx:=(image1.width/2+raio+dados[1]*amplia);
        dy:=(clientheight/2);
        moveto(trunc(xa),trunc(ya));
        lineto(trunc(dx),trunc(dy));
         xa:=(centrox);
        ya:=(centroy);
        moveto(trunc(xa-12),trunc(ya));
        lineto(trunc(xa+12),trunc(ya));
        moveto(trunc(xa),trunc(ya-12));
        lineto(trunc(xa),trunc(ya+12));
        {*****}
        x:=amplia;
{escala}
        xa:=centrox+menor_raio;
        dx:=xa+10*x;
        ya:=(clientheight/2-centro(deltay)*amplia);
        moveto(trunc(xa),trunc(ya));
        lineto(trunc(dx),trunc(ya));
        for j:=0 to 10 do
            begin
            moveto(trunc(xa),trunc(ya-5));
            lineto(trunc(xa),trunc(ya+5));
            xa:=xa+x;
            end;

        xa:=centrox-menor_raio;
        dx:=xa-10*x;
        moveto(trunc(xa),trunc(ya));
        lineto(trunc(dx),trunc(ya));
        for j:=0 to 10 do
            begin
```

```
            moveto(trunc(xa),trunc(ya-
5));

lineto(trunc(xa),trunc(ya+5));
            xa:=xa-x;
            end;

xa:=(image1.width/2+centro(deltax)*amplia
);
        ya:=centroy-menor_raio;
        dy:=(ya-10*x);
        moveto(trunc(xa),trunc(ya));
        lineto(trunc(xa),trunc(dy));
        for j:=0 to 10 do
            begin
            moveto(trunc(xa-
5),trunc(ya));

lineto(trunc(xa+5),trunc(ya));
            ya:=(ya-x);
            end;
        ya:=centroy+menor_raio;
        dy:=(ya+10*x);
        moveto(trunc(xa),trunc(ya));
        lineto(trunc(xa),trunc(dy));
        ya:=ya-x;
        for j:=0 to 10 do
            begin
            ya:=(ya+x);
            moveto(trunc(xa-
5),trunc(ya));

lineto(trunc(xa+5),trunc(ya));
            end;
         {informacoes}

str((maior_delta*escala_sensor1):1:5,st);
        grafico_polar.label2.caption:=st;

str((menor_delta*escala_sensor1):1:5,st);
        grafico_polar.label4.caption:=st;
        LSC:=(maior_delta-
menor_delta)*escala_sensor1;
        str(LSC:1:5,st);
        grafico_polar.label8.caption:=st;

        str((raio_medio(dados)):1:5,st);

grafico_polar.label14.caption:=st;
        str(escala_sensor1:1:5,st); {10
graduacoes - dividir x por 10}
     grafico_polar.label11.caption:=st;
end;
end;

procedure
Tgrafico_polar.Button2Click(Sender:
TObject);
begin
image1.picture.bitmap.monochrome:=true;
image1.picture.savetofile(edit1.text);
end;

end.
```

```
unit repete;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
  StdCtrls,clk, ExtCtrls;

type
  Tgrafico_repete = class(TForm)
    Button1: TButton;
    Image1: TImage;
    Button2: TButton;
    Edit1: TEdit;
    CheckBox1: TCheckBox;
    procedure linha(x1,y1,x2,y2:integer);
    procedure repete_inicio(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  grafico_repete: Tgrafico_repete;

implementation

{$R *.DFM}
procedure Tgrafico_repete.linha(x1,y1,x2,y2:integer);
begin  {Traça linha do ponto (x1,y1) até x2,y2}
with image1.canvas do
     begin
     moveto(x1,y1);
     lineto(x2,y2);
     end;
end;

procedure Tgrafico_repete.repete_inicio(Sender:
TObject);
var
    x,j,k:integer;                          {posicao x no
grafico}
    num : byte;                             {j,k -
contadores}
begin
   with image1.canvas do
  begin
  rectangle(0,0,clientwidth,clientheight); {limpar
imagem}
  x:=75;
  font.height:=30;
 j:=(clientheight)div(10);
  for k:=1 to 9 do   {Escrever escala: -5 a 5 V}
     begin
     st:=int2st(5-k)+'V';
     textout(2*256+x,j*k-15,st);
     st:=ssingle2st((5-
K)*escala_sensor1*escala_amp1)+'µm';
     textout(1,j*k-15,st);
     end;
 textout(2*256+x,1,' 5V');
textout(2*256+x,10*j-30,'-5V');
st:=ssingle2st(5*escala_sensor1*escala_amp1)+'µm';
textout(1,1,st);
st:=ssingle2st(-5*escala_sensor1*escala_amp1)+'µm';
textout(1,10*j-30,st);
x:=70;
        {tracar eixos e grades}
        repeat
              begin
              linha(x,j,x+2*256,j);
              j:=j+(clientheight)div(10);
              end;
        until j>clientheight;
        j:=x;
        repeat
              begin
```

```
            linha(j,0,j,clientheight);
            j:=j+2*64;
            end;
        until j>x+2*256;
        x:=70;

for num:=1 to valor do
begin
pen.color:=cores[num];
        for j:=1 to n do
            begin
            k:=(clientheight)div(2)-
trunc(tabela_repete[num][j]*((clientheigh
t)div(10)));
            x:=x+2;
            linha(x-
2,((clientheight)div(2)-
trunc(tabela_repete[num][j-
1]*((clientheight)div(10)))),x,k);
            end;
x:=70;
end;
pen.color:=clblack;
end;
end;

procedure
Tgrafico_repete.Button2Click(Sender:
TObject);
begin
if checkbox1.checked then
image1.picture.bitmap.monochrome:=true;
image1.picture.savetofile(edit1.text);
image1.picture.bitmap.monochrome:=false;
end;

end.
```

```
unit sobre;

interface

uses
  Windows, Messages, SysUtils, Classes,
Graphics, Controls, Forms, Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

end.
```

```
unit ver_config;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls;

type
  TConfig_atual = class(TForm)
    Button1: TButton;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    GroupBox2: TGroupBox;
    Label7: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label8: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Config_atual: TConfig_atual;

implementation

{$R *.DFM}

end.
```

Bibliografia Recomendada:

- HORIKAWA, O. **AUTOMAÇÃO DE PROCESSOS DE MEDIÇÃO - Circularidade de Componentes Mecânicos**
- LOPES, L. E. **Notas de aula da disciplina PMC 502 – Automação de Sistemas Mecânicos**.
- SCARR. A.J.T. ,**Metrology and Precision Engineering**. McGraw Hill European Mechanical Engineering Series, 1967
- ANTHONY. D. M. , **Engineering Metrology**, Pergamon Press,1986
- **Notas de aula da disciplina PMC 505 – Metrologia na Fabricação Mecânica**. Cap. 6. p. 6-3 à 6-7.
- LYNX TECNOLOGIA ELETRÔNICA. **Manual do Usuário e Referência Técnica – Conversor Analógico – Digital CAD 10/26**,1987
- BARCZAK, A.L.C. **Comparação entre algoritmos de mínimos quadrados e de zona mínima para desvios de circularidade**, Jun. 1996
- MALVINO, **Eletrônica – vol. I**, 4.ed. São Paulo, Makron Books, 1997.
- TAUB, H., **Circuitos Digitais e Microprocessadores**. São Paulo, McGraw Hill, 1984
- Smith, S.W., **The Scientist and Engineer's Guide to Digital Signal Processing.** California Technical Publishing (http://www.dspguide.com)

Apêndice

Método da Reversão

Neste método, o objeto é medido duas vezes por um medidor de circularidade conforme mostra a Fig.1 (é considerado um medidor de circularidade do tipo mesa giratória). Na primeira medição, um sinal (s(θ), θ: ângulo de rotação do eixo) igual a soma dos erros de circularidade do objeto (h(θ)) com o erro de movimento do medidor de circularidade (t(θ)), é obtido por meio do sensor.

$$s(\theta) = h(\theta) + t(\theta) \qquad (1)$$

Na segunda medição, procede-se a reversão, ou seja, desloca-se a orientação do objeto com relação ao sistema de rotação do medidor de circularidade de 180° e reinstala-se o sensor no lado oposto. Após a reversão, executa-se uma segunda medição, onde o seguinte sinal é obtido,

$$s'(\theta) = h(\theta) - t'(\theta) \qquad (2)$$

Assumindo que o movimento do eixo do medidor de circularidade tem alta repetibilidade, ou seja o erro de movimento antes e depois da reversão (t(θ)≅ t'(θ)) podem ser considerados iguais entre si, obtém-se o erro de circularidade do objeto da seguinte forma :

$$h(\theta) = [s(\theta) + s'(\theta)] / 2 \qquad (3)$$

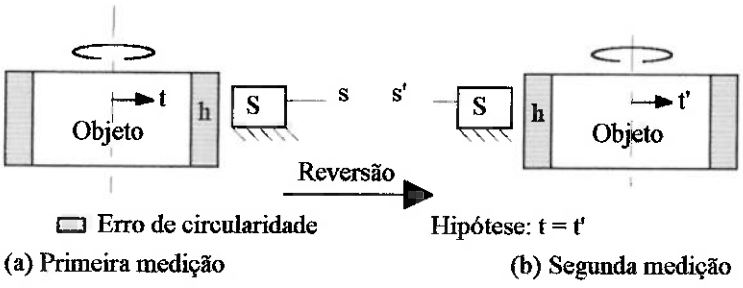


Fig. 1. Método da Reversão convencional, para medição de circularidade

Apesar de simples, o método da reversão convencional, apresenta um problema que é o da necessidade de se ter alta repetibilidade no movimento do eixo do medidor de circularidade. Isto implica que a parte física do sistema deverá ser composta por componentes com alta precisão e pouquíssimas falhas, elevando o custo do sistema de medição. Se não for atendida esta condição de alta repetibilidade, a confiabilidade dos resultados fica comprometida.

Método Aprimorado da Reversão

Uma vez que o método da reversão necessita de alta repetibilidade para sua validade, foi proposto um método chamado "método aprimorado da reversão" (MAR), que elimina tal necessidade. No MAR, um referencial cilíndrico (referencial auxiliar) é colocado sobre o objeto a ser medido, conforme mostra a Fig. 2. A medição é feita em duas fases. Na primeira fase, o objeto e o referencial auxiliar são medidos simultaneamente, e assumindo que o erro de movimento de inclinação do eixo é desprezível, os seguintes sinais (s1(θ) e s2(θ)) são obtidos por meio dos sensores S1 e S2.

$$s_1(\theta) = h(\theta) + t(\theta) \qquad (1)$$

$$s_2(\theta) = q(\theta) + t(\theta) \qquad (2)$$

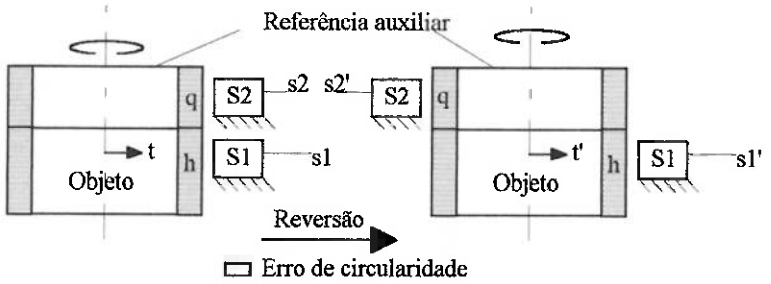
Sendo que t(θ) denota o erro de movimento do eixo de rotação do MC. h(θ) e q(θ) denotam os erros de geometria do objeto e do referencial auxiliar respectivamente.

Na segunda fase da medição, executa-se a reversão somente do referencial auxiliar e do sensor S2. Após a reversão, outra medição semelhante à da primeira fase é repetida obtendo-se os seguintes sinais:

$$s_1'(\theta) = h(\theta) + t'(\theta) \qquad (3)$$

$$s_2'(\theta) = q(\theta) - t'(\theta) \qquad (4)$$

onde t'(θ) denota o erro de movimento do eixo do MC após a reversão, sendo que neste caso, t'(θ) não é necessariamente igual a t(θ).



(a) Primeira medição (b) Segunda medição

Fig. 2. Método Aprimorado da Reversão, para medição de circularidade

Ao final, processando-se os resultados das medições da seguinte forma, obtém-se o erro de geometria do objeto:

$$h(\theta) = [(s_1(\theta) - s_2(\theta)) + (s_1'(\theta) + s_2'(\theta))]/2 \qquad (5)$$

E ainda, caso necessário, os erros de geometria do referencial auxiliar (q(θ)) e os erros de movimento antes e depois da reversão (t(θ) e t'(θ)) também podem ser obtidos por meio de uma aritmética semelhante:

$$t(\theta) = [(s_1(\theta) + s_2(\theta)) - (s_1'(\theta) + s_2'(\theta))]/2 \qquad (6)$$

$$t'(\theta) = [-(s_1(\theta) - s_2(\theta)) + (s_1'(\theta) - s_2'(\theta))]/2 \qquad (7)$$

$$q(\theta) = [-(s_1(\theta) - s_2(\theta)) + (s_1'(\theta) + s_2'(\theta))]/2 \qquad (8)$$

Como mostra a Eq. 5., o erro de movimento é eliminado do resultado da medição da circularidade do objeto, não havendo a necessidade de se ter alta repetibilidade nos movimentos do eixo do MC. Esta é a grande vantagem deste método. Por meio do MAR, uma medição de alta precisão pode ser executada com o uso de um eixo de precisão relativamente baixa. Outra vantagem é que este método requer a reversão somente do referencial auxiliar e do objeto a ser medido. Por esta razão, o método seria útil por exemplo, para se medir a circularidade de um componente torneado sem que haja a necessidade de retirá-lo do torno.